Hybridizing Genetic Algorithms and Particle Swarm Optimization Transplanted into a Hyper-Heuristic System for Solving University Course Timetabling Problem

Morteza Alinia Ahandani*
Department of Electrical Engineering
Langaroud Branch, Islamic Azad University
Langaroud
Iran
* alinia@iaul.ac.ir
Mohammad Taghi Vakil Baghmisheh
Department of Control Engineering
University of Tabriz
Tabriz
Iran
mvakil@tabrizu.ac.ir

Abstract: - In this paper, we use genetic algorithms (GAs), particle swarm optimization (PSO) and hybrid versions of them to solve university course timetabling problem (UCTP). A new crossover method called 2-staged n-point crossover by combining classic n-point crossover method and graph colouring heuristics is introduced which aims to generate free-conflict offspring. The hybrid algorithms are generated by adding a local search (LS), based on hill climbing (HC) method, on three global search algorithms i.e. the GA, the PSO and a combination of them called GAPSO. The proposed algorithms such as hyper-heuristic systems, manage a set of graph colouring heuristics as low-level heuristics in a hyper-heuristic strategy. The proposed algorithms are examined by 11 well-known benchmark problems. Experimental results demonstrate that the GA outperforms the PSO and the GAPSO algorithms, but the hybrid GAPSO algorithm has a better performance than the hybrid GA and hybrid PSO. Also all hybrid algorithms obtain a better performance than their non-hybrid competitors. However the GA has been widely applied to UCTP, to the best our knowledge the obtained results of GA in this paper are the first reported results on these databases which are competitive than results of other approaches. In a later part of the comparative experiments, a comparison of our proposed algorithms and 14 other approaches reported in the literature confirms that by considering the hybrid GAPSO as a hybrid hyper-heuristic, it is one of the best strategies for the hyper-heuristic systems on the UCTP proposed so far. Also results of the hybrid GAPSO in comparison of other hybrid algorithms proposed in the literature are completely comparable.

Key-Words: - Crossover, Genetic algorithm; Hybrid algorithm; Particle swarm optimization; University course timetabling; Hyper-Heuristic

1 Introduction

1.1 Course timetabling

The timetabling problems are a subclass of scheduling problems, which usually are highly constrained, thus difficult to solve. Indeed, due to complexity of the real-world problems it is impossible to satisfy all of the constraints. Therefore, to find a practical solution, it is necessary to relax some of the constraints, which are called soft constraints. Hence constraints are divided into two classes: hard constraints and soft constraints. Satisfaction of all hard constraints is compulsory, otherwise the obtained solution is considered infeasible. On the other hand, satisfaction of soft constraints is desirable but not mandatory. In a university course timetabling problem (UCTP), a number of events (lectures, laboratories, exercises, etc) are assigned into a limited number of resources, i.e. locations (classrooms, laboratories, meeting halls), and timeslots within a week. The distinctive

features of this class of timetabling problems are that lectures have common students and that availability and size of rooms plays an important role.

1.2 Literature review

Timetabling are problems of time-based planning and combinatorial optimization which tend to be solved with a cooperation of stochastic search such as evolutionary algorithms (EAs) and heuristic methods such as sequential graph colouring heuristics. Conventional computer-based automated timetabling methods concern themselves simply to find the shortest timetable that satisfies all the hard constraints, commonly using a sequential graph colouring heuristics, and less navigate toward optimizing over a collection of soft constraints. This research concerns on combining two different approaches. The first approach is a heuristic approach involving graph coloring methods which usually lead to satisfactory and feasible solutions. The second approach is known as the EAs which usually lead to near optimal solutions and can be used as an optimization approach on soft constraints.

Qu [1] divided the artificial intelligence approach applied on educational timetabling problem into six categories: traditional approaches, meta-heuristic methods, constraint logic techniques, knowledge-based techniques, hyper-heuristic methods, and decomposition methods. Also Abdullah [2)] divided the approaches applied on the UCTP into seven categories: constraint-based methods, graph-based approaches, population-based approaches, meta-heuristic methods, case-based reasoning (CBR), knowledge-based and fuzzy-based approaches, multi-criteria approaches, and hyper-heuristic approaches. This research by considering recently applied methods on educational timetabling problem classifies approaches used to solve various components of the UCTP into eight categories:

1.2.1 Clustering or decomposition methods

The clustering methods usually solve timetabling problems in three phases. In the first phase, the set of events are divided into groups which collect events that will be scheduled into the same resources. In each group, events do not conflict with each other. The second phase attempts to reduce second-order conflicts, i.e. number of violations from soft constraints, by finding the optimal sequence of groups. Finally, the third stage is employed with the aim of improving the solution quality further. This is done by moving a particular event between resources such as by employing a HC [2]. [3-5] employed different clustering methods to solve the UCTP.

1.2.2 Constraint-based approaches

In a constraint-based approach, a set of variables with a given domain represents a problem. These approaches insert values to variables in such a way that all constraints of problem are fulfilled. Constraints are relations that are assumed to hold over variables and define the solutions space. Different variations of the logic programming language have been employed in the wide variety of constraint-based methods that have appeared in the literature (for example see [6-8]).

1.2.3 Graph-based approaches

Graph-colouring heuristics are often called sequential heuristics. The main idea is to assign events to resources, one by one, based on a sequencing strategy [9]. Timetabling problems, without considering of soft constraints, can be modeled as graph coloring problems. Graph coloring heuristics were widely used to solve the timetabling problems Burke et al. [10] reviewed the application of graph coloring methods to timetabling. The authors discussed various timetabling problems i.e. class/teacher, course, exam and sports timetabling. Their probe included the role that graph coloring methods have played in the timetabling literature over the last 40 years or so. The reported results in Carter et al. [11] demonstrated that sequential heuristics were very efficient when incorporating a backtracking procedure. Burke et al. [12] employed a heuristic procedure without backtracking but incorporated a random element and Asmuni et al. [13] proposed a fuzzy heuristic ordering. Burke and Newall [14] presented a method for solving examination timetabling problems through adaption of heuristic orderings as an alternative to existing forms of backtracking. Also some of literature used the graph coloring methods as low level heuristics in a hyper-heuristic structure. Burke et al. [15] investigated a tabu search (TS) hyper-heuristic approach upon a set of graph colouring heuristics for university timetabling. Pillay et al. [16] proposed an alternative representation for heuristic combinations, namely, a hierarchical combination of heuristics. Those, meantime introducing of a new low-level heuristic called highest cost, combined the low-level heuristics hierarchically and applied simultaneously rather than sequentially.

The graph colouring technique adapts well to small-scale problems, however they fail to scale up for

larger ones [17]. Normally the real timetabling problem is a large-scale problem, so the timetabling problem solved by the graph colouring approach is still far from the real situations encountered in timetabling.

1.2.4 Metaheuristics and EAs

Compared to other approaches, the EAs, particularly hybrid versions of them, can be very successful in dealing with a variety of soft constraints and thus can generate high quality solutions**.** The EAs can mainly be divided to two distinct groups: point-based or local search (LS) algorithms and population-based or global search algorithms. The LS algorithms explore the solution space by a gradual improvement of the current solution. Classical examples are hill climbing (HC), simulated annealing (SA), TS, variable neighbourhood search (VNS) and great deluge (GD) algorithm. In addition, global search EAs, such as genetic algorithms (GAs), ant colony algorithm (ACS) and memetic algorithms (MAs) perform the search by maintaining a population of candidate solutions.

During the recent years, the various EAs have been intensively applied to solve timetabling problems. Burke et al. [18] described the use of a GA to solve timetabling problems and Ergül [19] implemented a university examination timetabling method based on a GA for the Middle East Technical University. Pillay and Banzhaf [20] presented the results of a study conducted to investigate the use of GAs as a means of inducing solutions to the examination timetabling problem. This method firstly took a two-phased approach to the problem which focused on producing timetables that met the hard constraints during the first phase, while improvements were made to these timetables in the second phase so as to reduce the soft constraint costs. Secondly, domain specific knowledge in the form of heuristics was used to guide the evolutionary process.

One of the most utilized algorithms for solving of the university timetabling problem is the TS algorithm. Hertz [21] and White et al. [22] independently applied the TS algorithm to this problem. In Aladag et al. [23] two new neighborhood structures were proposed by using the moves called simple and swap and the effects of these moves on the operation of TS were examined based on defined neighborhood structures. Also among of other EAs, [24-28] used SA and [29-31] applied ACS to the university timetabling problems. A comparison among five metaheuristic approaches for the same eleven datasets was presented in Rossi-Doria et al. [32]. These approaches include the ACS, the SA, random restart LS, the GA and the TS. A stochastic optimization timetabling tool (SOTT) has been developed for the UCTP in Pongcharoena et al. [33]. The GAs, the SA and random search were embedded in the SOTT. Landa-Silva and Obit [34] proposed a modeled GD algorithm called Nonlinear Great Deluge (NLGD) by using a nonlinear decay of water level. In the original GD, the water level decreases steadily in a linear fashion but they proposed a modified version of the GD algorithm in which the decay rate of the water level was non-linear They successfully improved the performance of the GD algorithm on medium UCTP instances.

Also Lewis [35]presented a survey of metaheuristic-based techniques for university timetabling problems. Those subdivided the metaheuristic algorithms proposed for timetabling into three categories: One-stage optimization algorithms where a satisfaction of both the hard and soft constraints is attempted simultaneously. Two-stage optimization algorithms where a satisfaction of the soft constraints is attempted only once a feasible timetable has been found. Algorithms that allow relaxations where violations of the hard constraints are disallowed from the outset by relaxing some other feature of the problem, and attempts are then made to try and satisfy the soft constraints, whilst also giving consideration to the task of eliminating these relaxations.

1.2.5 Knowledge-Based techniques and CBR

The overall objective of using knowledge-based techniques for timetabling is to model the human knowledge for timetabling. Kong and Kwok [36] implemented a conceptual model of a knowledge-based timetabling system for high school timetabling. Foulds and Johnson [37] developed a database decision support system for a real world course timetabling problem.

All the existing knowledge-based techniques on timetabling use expert system, which models the knowledge of timetabling as rules, to generate course timetables. One possible problem with this is that usually the knowledge within the scheduling is implicit thus difficult to be modeled. This may be resolved by either the careful design of specific problems, or by employing techniques that can use the knowledge and avoid large amounts of work in modeling it. The CBR can be considered one of the solutions for this problem [1]. The CBR is an artificial intelligence technique that is supported by the study of cognitive science. It is motivated by the observation that humans use past experience to solve similar problems and reuse that experience

with some modification to suit different requirements [38]. [39] and [40] used the CBR to solve timetabling problems.

1.2.6 Hyper-Heuristics techniques

Burke et al. [41] defined a hyper-heuristic as '*the process of using* (*meta*-) *heuristics to choose* (*meta*) *heuristics to solve the problem in hand*'. Unlike most implementation of meta-heuristics that modify solutions directly, a hyper-heuristic modifies solutions indirectly by employing the selected low-level heuristics. A hyper-heuristic operates on the search space of heuristics rather than on the search space of candidate solutions (see 15) and 16)). Qu and Burke [42] investigated the effect of employing different high-level search algorithms (i.e. steepest descent, TS, iterated LS and VNS) in the unified graph based hyper-heuristic framework. Experimental results demonstrated that the method of search by different high-level heuristics within the search space of graph heuristics was not crucial. The characteristics of the neighbourhood structures and search space were analyzed. It was shown that the exploration over the large solution space enabled the approach to obtain good results on both the exam and course timetabling problems.

1.2.7 Fuzzy-based approaches

Asmuni et al. [13], [43] and [44] investigated the fuzzy-based approaches on timetabling problems. Asmuni et al. [43] and [44] discussed how fuzzy techniques could be used to combine multiple standard heuristics to construct educational timetables. Petrovic et al. [45] considered fuzzy constraint satisfaction in timetabling problems. Chaudhuri and De [46] presented a fuzzy genetic heuristic algorithm to solve the UCTP.

1.2.8 Artificial neural networks

Artificial neural networks have recently proven to be relatively successful in solving complex combinatorial optimization problems (see [47] and [48]).

In addition to these methods, some approaches have been applied for the educational timetabling problems which rarely utilized in other literature. These include integer programming [49] and [50], VNS [51] and randomized iterative improvement [52]. For more details about applied approaches on timetabling problems see [53-56].

1.3 The hybrid EAs vs timetabling

The reported results of applying the GA on timetabling problem show that the original GA can not find a solution with good quality [57] and [58]. Thus a combination of GA with other algorithms usually has been used to improve the quality of obtained timetables. On the other part, the hybridizing with a LS technique is an efficient approach to improve the quality of original EA. Burke et al. [59] employed a MA that was combined a GA and a HC for university examination timetabling. Merlot et al. [60] implemented the hybridization between constraint programming to obtain a feasible initial timetable and LS to improve those of initial solutions. Azimi [61] presented three hybrid combinations of the TS and the ACS for a classical examination timetabling problem. In each hybrid algorithm, the TS or the ACS was considered as main algorithm and another algorithm was used in the LS part of it. Chiarandini et al. [62] presented a hybrid algorithm for the UCTP by combining various construction heuristics, the TS, variable neighbourhood descent and the SA. The LS and TS procedures were used for solving the hard constraints, while a timetable was improved in terms of soft constraints by means of variable neighbourhood descent and SA.

Yang, and Jat [63] investigated the GAs with a guided search strategy and LS techniques for the UCTP. The guided search strategy was used to create offspring into the population based on a data structure that stored information extracted from good individuals of previous generations. The LS techniques used their exploitive search ability to improve the search efficiency of the proposed GAs and the quality of individuals. The experimental results showed that the proposed GAs were able to produce promising results for the UCTP. Abdullah and Turabieh [64] proposed a GA with sequential LS, called GAWLS. They tested a GA with a repair function and LS on the UCTP. Since combinations of evolutionary based approaches with LS have provided very good results for a variety of scheduling problems, Abdullh et al [65] proposed such an algorithm for the UCTP. Their evolutionary method did not use a crossover operator. After applying the mutation operator on %20 of the courses from each selected individual, the LS component was employed. This hybrid evolutionary approach was tested over established datasets and compared against state-of-the-art techniques from the literature. The results obtained confirmed that the approach was able to produce solutions to the UCTP which exhibited some of the lowest penalty values in the literature on benchmark problems. It was therefore concluded that the hybrid evolutionary approach represented a particularly

effective methodology for producing high quality solutions to the UCTP.

Rossi-Doria et al. [32] proposed a hybrid GA. They used a LS method with the GA to solve the UCTP and also compared several meta-heuristics methods i.e. EAs, ant colony optimization, iterated LS, SA, and TS on the UCTP. To attempt fairness, the implementations of all the algorithms used a common solution representation, and a common neighbourhood structure or LS. The results showed that no meta-heuristic was best on all the timetabling instances considered. Jat and Yang [66] presented a MA that integrated two LS methods into the GA for solving the UCTP. These two LS methods used their exploitive search ability to improve the explorative search ability of GAs. The first LS worked on all events by supposing that each event was involved in soft and hard constraint violations. When the first LS finished, they got a possibly improved and feasible individual. After that, they applied the second LS on the current individual. The second LS could enhance the individuals of the population and increase the quality of the feasible timetable by reducing the number of constraint violations. The experimental results indicated that the proposed MA was efficient for solving the UCTP.

In this research, the GA, particle swarm optimization (PSO) and a combination of them are used as global search optimization algorithms to solve the UCTP. In order to use beneficiary of hybrid schemes, the aforementioned EAs are combined with a LS i.e. HC method. Also a new crossover operator which enhances with graph-based heuristics is proposed. The algorithms proposed in this research, because of employing the EAs to arrange the graph-based heuristics, as well as can be considered as hyper-heuristics systems. The rest of the paper is organized as follows. In section 2, course timetabling problem are briefly explained. In section 3, the graph colouring heuristics are described. In section 4, the GA, the PSO, the utilized LS and the method of hybridizing are explained. The simulation results are presented and analyzed in section 5. Section 6 concludes the paper.

2 The UCTP

UCTP consists of a set of courses to be assigned in a set of timeslots and a set of rooms in which courses can take place within a week. The solution of this problem must satisfy all of hard constraints without any violation, whereas it can necessarily violate from some of soft constraints. Proportion of violation of soft constraints in this problem, measures the solution quality.

Because several university course timetabling papers proposed in the literature applied their approach to the problem instances described in Socha et al. [29], this paper also is focused on this proposed UCTP. [29] proposed the following hard constraints:

I. No student can be assigned to more than one course at the same time.
II. The room should satisfy the features required by the course.
III. The number of students attending the course should be less than or equal to the capacity of the room.
IV. No more than one course is allowed at a timeslot in each room.

Also the following soft constraints were presented:

I. A student has a course scheduled in the last timeslot of the day.
II. A student has more than 2 consecutive courses.
III. A student has a single course on a day.

The problem consists of a set of N courses, $C = \{c_1, c_2, c_3, ..., c_N\}$, T timeslots, $TS = \{t_1, t_2, t_3, ..., t_T\}$ (a given number of work days and a given number of timeslots in every day), a set of R rooms in which events can take place, a set of F room features satisfied by rooms and required by events and a set of M students who attend the events. Thus the objective function of this problem can be considered as (1). This cost function simply counts the number of violations of the obtained solution from hard and soft constraints. It is a penalty function of weighted sum of violations.

$$C.F = \sum_{i=1}^{4} w_i * HC_i + \sum_{j=1}^{3} w_j * SC_j \quad (1)$$

where HC_i and SC_j denote number of violations from i th hard constraint and j th soft constraint, respectively. w_i and w_j are penalty weighting associated with i th hard constraints and j th soft constraints, respectively. Also to satisfy the requests of hard constraints, values of w_i and w_j are set equal to 10 and 1 for all hard and soft constraints, respectively. Lewis [35] mentioned two main advantages for this sort of linear weighted cost function. First, because the aim is to simply search for a candidate solution that minimizes a single cost function, it can, of course, be used with any reasonable optimization technique. Second, this approach is, in general, very flexible and easy to implement, because any sensible constraint can be incorporated into the problem provided that an

appropriate penalty weighting (which indicates its relative importance compared to others) is stipulated. In particular, this second factor is highly convenient for timetabling problems where we can often encounter an abundance of different constraint combinations in practice.

3 Graph Colouring Heuristics

Graph colouring is concerned with colouring the vertices of a given graph using a given number of colors. The relationship of graph colouring problem and timetabling is widely discussed in the literature (see [15], [67] and [68]). In the graph-based structure for the timetabling problem, events, timeslots, and conflicts are modeled by vertices, colors, and edges, respectively. The difficulty of events called as the degree of vertices, is represented by the number of conflicts those have with the others. A conflict between two events in timetabling problem exhibits at least existence of a same student. So these two events must not be scheduled in a same timeslot. The graph colouring heuristics can be used to construct a new timetable or to perfect incomplete timetables.

The principle idea behind using graph colouring heuristics in the timetabling problems is to order the events, one by one, based on their difficulties of scheduling and to assign consecutively them into feasible timeslot and room. It is obvious, in the early stages of scheduling there are more feasible timeslots to assign those of difficult events. Various graph colouring heuristics assign different difficulty degree for a same event in a considered timetable. Some sequential graph colouring heuristics are as follow:

Largest degree (LD): courses with the largest number of conflict with other courses are scheduled first.

Largest enrolment (LE): courses with the largest number of student enrolment are scheduled first.

Largest weighted degree (LWD): in this heuristic, priority is given to the course that has the largest weighted conflict. Each conflict is weighted based on the number of students involved in two conflicting courses.

Random ordering (RO): the courses that are not yet scheduled are selected randomly.

Color degree (CD): in this heuristic that is a dynamic heuristic, the courses are ordered in terms of the number of conflict that they have with those already scheduled in the timetable.

Saturation degree (SD): in this dynamic heuristic, the next selected course to be scheduled is based on the number of available feasible timeslots. The course with the least number of available feasible timeslots will be scheduled first.

4 The Utilized Algorithms

4.1 The GAs

A GA starts by creating a random population of chromosomes, called initial population, and then these chromosomes are evaluated by the cost function and sorted in a decreasing order. Percent of chromosomes which are inferior to others are eliminated. Now two of remaining chromosomes are selected randomly to produce the offspring using crossover operator to replace the eliminated chromosomes. This reproduction (selection and crossover) continues, until the population reaches to its original size. The mutation operator is applied to the whole population of chromosomes with a mutation rate, commonly excluding the elite one. The resultant population is called the first generation. Again the cycle of evaluation, sorting, elimination, reproduction, and mutation continues until fulfilling one of stopping conditions. The different utilized operators in the GA for the course timetabling problem are as follow:

Initialization: coding of a chromosome as a problem solution is the first step of applying the GA to a problem. In the UCTP, each solution must be simultaneously assign the associated timeslot and room of each course. Thus, in a direct representation, with assumption of N courses, each chromosome will own a length equal to $2*N$ genes that N first-genes will assign timeslot of each course and N second-genes will assign room of each course. Fig. 1 shows an example of such a chromosome with 10 courses, 5 timeslots and 4 rooms. For example this chromosome represents that 1st course must occur at 1st timeslot in 3rd room, 2nd course must occur at 2nd timeslot in 2nd room and so on. Also every generated chromosome is evaluated using (1).

Selection: for the selection operator, we use the roulette wheel method with reverse linear rank weighting probability proposed in [69].

Crossover: in the crossover stage, two selected chromosomes in the selection stage are combined together to generate a new offspring. Each offspring has two sets of genes: those of genes are exactly copied from its parents and those of genes are exclusively generated for it. So each crossover method must present a strategy to copy genes from parents and an operator to generate new independent genes. The classic crossover operators, such as uniform and n-point crossover, can easily lead to

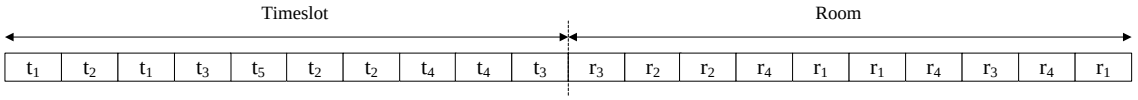

Fig 1. An example of chromosome coding for course timetabling problem.

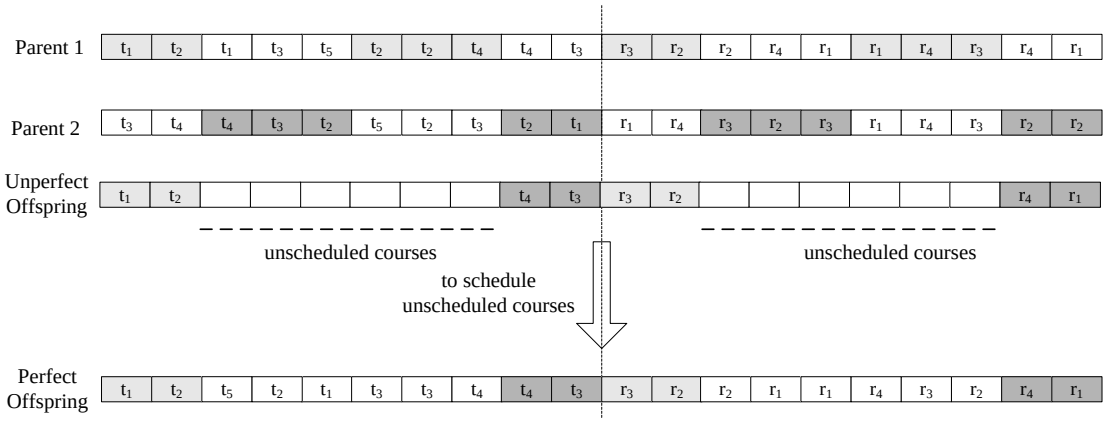

Fig 2. An example of 2 staged n-point crossover.

infeasible timetables. On the other word, the classic crossover methods are applied on feasible solutions but generate infeasible solutions. In order to avoid infeasible solutions, some of literature did not use the crossover and employed only the mutation operator to generate new solutions (see [70] and [71]). Some of other literature to preserve the feasibility of the timetables and to guarantee feasible timetables used a repair mechanism. This mechanism must be applied on infeasible solutions to reintroduce all the necessary courses (see [72] and [73]). This research, as a third approach, proposes a new crossover method called 2-staged n-point crossover (2SNPC) by combination of n-point crossover and graph colouring heuristics. This crossover method by applying to feasible solutions, with a high probability, leads to a feasible offspring.

In the 2SNPC crossover method, the n-point crossover is used to copy genes from parents and graph colouring heuristics are employed to generate new independent genes. Fig. 2 shows an example of 2SNPC. In the first stage of 2SNPC method, the first set of genes (both timeslots and rooms) are copied from first parent to the offspring and to copy second set of genes from second parent to the offspring, feasibility of this assignment is checked. If assignment of a gene (timeslot or room) from every parent generates a conflict, this set of genes in this stage is not assigned any value (no timeslot and no room) and will be scheduled in second stage. These unscheduled genes are denoted with value of zero in Fig. 2. In the second stage of 2SNPC method, to schedule the unscheduled courses, the graph colouring heuristics will be used. First the unscheduled courses will be ordered decreasingly in terms of difficulty degree of scheduling which is defined by graph colouring heuristic. Then scheduling is begun from course with highest priority and a random feasible timeslot and room are assigned to unscheduled courses. For each course, if any feasible timeslot or room is not found, a random value will be assigned to it.

Mutation: the mutation operator also uses graph colouring heuristics. Firstly a chromosome is selected with a probability of $chr - prob$ and then a set of genes from this selected chromosome will be selected with a probability of $gene - prob$. These unscheduled courses will be ordered decreasingly by graph colouring heuristics and the scheduling is begun from course with highest priority. Then a random feasible timeslot and room are assigned to unscheduled courses. Also, if any feasible timeslot or room was not found then a random value will be assigned.

Termination criterion: the GA continues until fulfils a given number of function evaluations.

4.2 The PSO

In contrast to the GAs which used evolutionary operators (selection, crossover and mutation) to generate and improve new candidate solutions, the PSO simply assigns a velocity vector to each member of population called particle, and updates the velocity of the members without generation of a completely new member. In the PSO each particle based on its previous position, its best position

achieved so far and the best position achieved by all particles adjusts its flying.

According to above discussion, the PSO is formulated as follow: denote attributes of each particle i ($i = 1,2,...,Npop$ and $Npop$ is population size) in the D-dimensional search space and iteration t is represented as follow: the current position of particle represented as $X_i^t = (x_{i1}^t, x_{i2}^t, ..., x_{iD}^t)$, the current velocity of particle represented as $V_i^t = (v_{i1}^t, v_{i2}^t, ..., v_{iD}^t)$, the current personal best position of particle represented as $P_i^t = (p_{i1}^t, p_{i2}^t, ..., p_{iD}^t)$ and the global best position represented as $G^t = (g_1^t, g_2^t, ..., g_D^t)$. Thus the particle i at dimension j and iteration t updates its velocity and position based on its cognition part and the social part according to Eqs. (2) and (3) respectively.

$$v_{ij}^t = \omega^{t-1} v_{ij}^{t-1} + c_1 r_1 (p_{ij}^{t-1} - x_{ij}^{t-1}) + c_2 r_2 (g_j^{t-1} - x_{ij}^{t-1}) \quad (2$$

$$x_{ij}^t = x_{ij}^{t-1} + v_{ij}^t \quad (3)$$

where ω is the inertia weight factor that adjusts the weighting of previous velocity in the current velocity; r_1 and r_2 are two random values with uniform distribution in the interval [0,1], c_1 and c_2 are learning factors. These two parameters specify tendency of particle to its own experiences or collective consequences.

The UCTP is a discrete optimization problem but standard PSO equations (Eqs. (2) and (3)) are suited for continuous optimization. To apply the PSO to the course timetabling problem, we used a structure proposed in Pan et al. [74]). They to apply the PSO on no-wait flowshop scheduling problem, proposed a new position update method for particles based on discrete permutations. They generated a new particle in three stages: one mutation and two crossover stages. Thus the position of particle i at iteration t can be updated as follow:

$$X_i^t = c_2 \otimes F_3(c_1 \otimes F_2(\omega \otimes F_1(X_i^{t-1}), P_i^{t-1}), G_i) \quad (4)$$

The update equation consists of three components: the first component is $\lambda_i^t = \omega \otimes F_1(X_i^{t-1})$, F_1 represents the mutation operator with the probability of ω. The second component is $\delta_i^t = c_1 \otimes F_2(\lambda_i^t, P_i^{t-1})$, F_2 represents the crossover operator with the probability of c_1. The third component is $X_i^t = c_3 \otimes F_3(\delta_i^t, G^t)$, F_3 represents the crossover operator with the probability of c_2. So the position of a particle is updated by using a mutation with the probability of ω, a crossover with the probability of c_1 and a another crossover with the probability of c_2. This strategy is used to apply the PSO to UCTP.

4.3 The GAPSO

We combine the GA with the PSO and produce a new global search algorithm called GAPSO. This proposed hybrid algorithm has a simple structure. The GAPSO serially applies two global search methods on a population. Each algorithm will be applied on improved members of another algorithm. The steps of GAPSO are as follow:

Step1: Generate initial population of size N_{pop}.

Step2: Apply the GA to population for (Max1) iterations.

Step3: Apply the PSO to population for (Max2) iterations.

Step4: Check the stopping criteria, if are not met go to **Step2** and repeat the algorithm.

The hybrid algorithms start by generation of an initial random population. Then the GA is applied to them with a predefined number of iterations (Max1). In simple words, in each iteration of algorithm, the GA receives the population, improves them and renders them to the PSO as its initial population. Later, the PSO method is applied to the population with a predefined maximum iteration number (Max2). Finally the termination criteria are checked and the algorithm is repeated until fulfilling one of the termination criteria.

4.4 The Hybridizing Global Search Algorithms and LS Method

In order to improve the performance of global search algorithms, a LS method is applied on obtained solutions of them. The steps of hybrid algorithms are as follow:

Step1: Generate initial population of size N_{pop}.

Step2: Apply the global search algorithm to population for (Max3) iterations.

Step3: Apply the LS to population for (Max4) iterations.

Step4: Check the stopping criteria, if are not met go to **Step2** and repeat the algorithm.

This hybridizing method is applied to the GA, PSO and GAPSO global search algorithms and three hybrid algorithms are obtained which are called HGA, HPSO and HGAPSO, respectively. All hybrid algorithms follow the aforementioned steps. Their single difference is in the **Step2**. The HGA uses the GA as global search algorithm in this step.

1. *Select a gene with a probability of* $local-prob$;
2. *Assign a feasible value for selected gene and generate new chromosome* $(Xnew)$;
3. *if* $f(Xnew) < f(X)$
   replace X *with* $Xnew$;
   else
   do not change X;
4. *Go to step* 1 *and repeat steps* $(1-3)$ *for other genes of chromosome* (X);

Fig. 3. The stages of proposed local search method for each chromosome.

The PSO is used in the HPSO as a global search algorithm in this step. The HGAPSO employs the GAPSO as global search algorithm.

Also for the LS, we use a simple method similar to HC. The HC algorithm iteratively evaluates some of neighbouring solutions and replaces the current solution by the candidate solution which results in the largest increase in the solution quality. The stages of this proposed LS method for each member are shown in Fig. 3.

5 Experimental Results

The proposed algorithms are tested on eleven benchmark course timetabling problems, proposed by the Metaheuristic Network[1]. The problems[2] need to schedule 100–400 courses into a timetable with 45 timeslots (5 work days and 9 timeslots a day), while satisfying room features and capacity constraints. These databases are divided into three groups: *Small*, *Medium* and *Large*. Specifications of these databases are shown in Table 1. Every algorithm was run 10 times and 1000000 function evolutions for *Small* databases, 1200000 function evolutions for *Medium* and *Large* databases was considered as stopping criterion, respectively. The performance of different algorithms was compared using two criteria: (i) the average value of the solution obtained in all trial runs (mean), (ii) the minimum value of the solutions obtained in all trial runs (min). The considered values for different parameters of algorithms are as follow:

- size of initial population equal to 60;
- number of crossover points equal to $\frac{N}{3}$ that N is number of courses;
- the GA parameters (ω, $chr-prob$, $gene-prob$) equal to 0.1and 0.2, respectively;
- the PSO parameters (ω, c_1, c_2) equal to 0.3, 0.8 and 0.8, respectively;
- the GAPSO parameters (Max1, Max2) equal to 20 and 20, respectively;
- the LS parameters ($local-prob$, Max4) equal to 0.4 and 3, respectively;
- the value of Max3 in the HGA, HPSO and HGAPSO equal to 5, 5 and 1, respectively.

To generate the initial population, we used a feasible assignment by starting from an empty timetable. In this initializing method, for each course, a random timeslot and room is selected and if theses assigned values were feasible, those will be accepted. Also if any feasible timeslot and room were not found, a random assignment will be considered. The experiments on 1000 random members generated using this initializing method show that this method on *Small* databases generates completely feasible solutions. But on *Medium1*, *Medium2*, *Medium3*, *Medium4*, *Medium5* and *Large* databases averagely leads to 43, 46, 109, 35, 262 and 491 infeasible assignments, respectively. Also the term ''x% Inf'' in some of this section tables indicates the percentage of runs which associated algorithm failed to obtain feasible solutions.

5.1 Decision on proposed algorithms

The proposed evolutionary operators, i.e. crossover and mutation, use the graph colouring heuristics to order unscheduled courses. Table 2 shows a comparison among performance of different heuristics in the GA on 4 databases. From results of Table 2 we can observe that on *Small1*, the LWD and LD heuristics obtain the best minimum and mean cost, respectively. On this problem, the RO has the worst performance. On *Small2*, the LE and CD heuristics have the best minimum and mean cost, respectively. Also the RO obtains the worst performance. On *Medium1*, the SD gives a better minimum cost and the LD has a better mean cost. For problem *Medium2*, the LE and SD heuristics give the best minimum and mean cost, respectively. The obtained results of Table 2 demonstrate that for all of the problems tested, the GA with different heuristics finds feasible solutions. Also the RO heuristic, that is random ordering of unscheduled courses, obtains the worst performance on all problems except on *Medium1* and *Medium2* in terms of mean cost. But it is evident that, among other heuristics, no heuristic obtained significantly better

[1] http://www.metaheuristics.net/.

[2] http://iridia.ulb.ac.be/~msampels/ttmn.data/.

Table 1. Specifications of the utilized database

category	*Small*	*Medium*	*Large*
Number of courses	100	400	400
Number of rooms	5	10	10
Number of features	5	5	10
Number of students	80	200	400
Maximum courses per student	20	20	20
Maximum student per courses	20	50	100
Approximate feature per room	3	3	5
Percent feature use	70	80	90

Table 2. A comparison among performance of different graph colouring heuristics in the GA on 4 databases

Graph heuristic	*Small1*		*Small2*		*Medium1*		*Medium2*	
	min	mean	min	mean	min	mean	min	mean
RO	14	17.9	15	21.3	215	244.7	227	250.2
LE	12	17.6	10	18.7	212	248.1	217	241.6
LD	9	14.5	11	19.3	205	236.5	227	254.6
LWD	7	15.3	12	17.2	203	249	224	248.3
CD	9	14.7	12	16.9	211	241.1	223	243.2
SD	8	14.8	11	17.2	197	239.5	219	240.6

Table 3. A comparison among non-hybrid global search algorithms.

Database	GA		PSO		GAPSO	
	min	mean	min	mean	min	mean
Small1	**6**	**12.8**	11	20	8	15.7
Small2	**8**	**15.2**	11	24.8	9	16.7
Small3	**7**	**15.3**	9	16.4	**7**	16.1
Small4	**8**	**15**	10	16.3	**8**	17.6
Small5	**3**	7.5	6	13	4	**7.3**
Medium1	**187**	**221.1**	225	289.5	198	240.9
Medium2	**202**	**240.6**	284	314.2	242	302.2
Medium3	**252**	**317.2**	321	370.4	286	338.4
Medium4	**224**	**274.5**	280	337	240	285.5
Medium5	268	**308.5**	-	100% Inf	**260**	311.2
Large	-	100% Inf	-	100% Inf	-	100% Inf

performance.

Table 3 shows a comparison of the PSO, the GA and the GAPSO results based on 11 university course timetabling databases. The best results were highlighted. The obtained results demonstrate that the GA has a better performance than two other global search algorithms in terms of both considered aspects, except on *Small5* that the GAPSO has a better average result. Also the PSO can not find any feasible timetable on *Medium5* and all algorithms obtain infeasible timetables on *Large* database.

The experimental results after applying LS method on global search algorithms are shown in Table 4. In comparison with results of Table 3, hybridizing can improve the quality of solutions. All hybrid algorithms outperform their non-hybrid competitors in Table 3. However, the GA had the best results among non-hybrid global search algorithms in Table 3, the HGAPSO obtains the best results among hybrid algorithms on all databases, except on *Medium5*. Also the HPSO algorithm, such as PSO in Table 3, has the worst performance among hybrid algorithms.

It is clear from Tables 3 and 4 that the GA and the HGAPSO are two superior non-hybrid and hybrid algorithms which give the best results than other algorithms.

5.2 Comparison with previous studies

Table 5 compares results of the GA and HGAPSO which were the best non-hybrid and hybrid algorithms, and three other hyper-heuristic approaches proposed in literature. The utilized studies include:

- The GHH upon six heuristics [15].
- The TS hyper-heuristic (TSHH) [75].
- The fuzzy multiple heuristic (FMH) [76].

Results of Table 5 clearly evident the promising results of our proposed methods. The GA has a better performance than the GHH, the TSHH and the FMH on 5, 2 and 7 problems, respectively. Also the HGAPSO has a better performance than the GHH, the TSHH and the FMH on 9, 4 and 9 problems, respectively. Based on this comparison, The HGAPSO obtains the best results on 6 problems and the second-best results on 3 other problems. Thus by considering the HGAPSO as a hybrid hyper-heuristic, it is one of the best strategies for

Table 4. A comparison among hybrid global search algorithms.

Database	HGA		HPSO		HGAPSO	
	min	mean	min	mean	min	mean
Small1	2	5	4	6.4	**0**	**1.2**
Small2	3	5.6	5	7.8	**1**	**2.4**
Small3	2	6.6	3	6.2	**0**	**1.9**
Small4	3	5.6	3	6.8	**1**	**3.8**
Small5	0	1.4	1	1.8	**0**	**0.8**
Medium1	178	196.2	184	204.2	**175**	**184.4**
Medium2	191	205.3	198	210.7	**184**	**196.1**
Medium3	212	224.7	221	238.2	**205**	**218.2**
Medium4	**174**	**191**.4	190	208.3	176	192.5
Medium5	201	214.8	312	50% Inf	**180**	**194.8**
Large	-	100% Inf	-	100% Inf	-	100% Inf

Table 5. The best results obtained by our proposed algorithms, i.e. the GA and HGAPSO, and other hyper-heuristic methods.

Database	GA	HGAPSO	GHH	TSHH	FMH
Small1	6	**0**	6	1	10
Small2	8	**1**	7	2	9
Small3	7	**0**	3	**0**	7
Small4	8	**1**	3	**1**	17
Small5	3	**0**	4	**0**	7
Medium1	187	175	372	**146**	243
Medium2	202	184	419	**173**	325
Medium3	252	**205**	359	267	249
Medium4	224	176	348	**169**	285
Medium5	268	180	171	303	**132**
Large	100% Inf	100% Inf	**1068**	80% Inf 1166	1138

hyper-heuristic systems on the UCTP. Also the GA obtains a competitive performance with GHH, LS and fuzzy multiple heuristic approaches. It outperforms the GHH and fuzzy multiple heuristic approaches on all *Medium* databases, except *Medium5*.

Table 6 compares results of the GA and HGAPSO and eleven other EAs proposed in literature. The utilized studies include:

- The LS [29].
- The ant algorithm (Ant) [29].
- The DCABA [76].
- The randomized iterative (RI) [52].
- The EGSGA [63].
- The NLGD [34].
- The GAWLS [64].
- The HEA Abdullah et al [65].
- The HGA [32].
- The VNS-Tabu [51]
- The MA [66].

(Note: the reported results for LS and Ant algorithm are average of obtained results, but the reported results for other approaches are the best results).

The GA obtains a better performance than the Ant, DCABA, RI, LS, EGSGA, NLGD, GAWLS, HEA, HGA, VNS-Tabu and MA on 1, 0, 2, 9, 0, 0, 4, 1, 2, 5 and 1 problems, respectively. Also the HGAPSO obtains a better performance than the Ant, DCABA, RI, LS, EGSGA, NLGD, GAWLS, HEA, HGA, VNS-Tabu and MA on 6, 5, 3, 10, 0, 4, 9, 2, 6, 5 and 3 problems, respectively. The performance of the HGAPSO on *Small1*, *Small3* and *Small5* is in the range of the best algorithms. It also has the second-best results on two other *Small* databases. There are 5 hybrid algorithms in this comparison which are a combination of GA and another LS, i.e. EGSGA, GAWLS, HEA, HGA and MA. Among these algorithms, the HGAPSO obtains the best results on *Small1*, *Small3* and *Small5* problems and the second-best results on *Small2*, *Small4*, *Medium1* and *Medium3* problems. Also the EGSGA has a considerably better performance than other algorithms.

So in an overall view, the HGAPSO method obtains a competitive performance on *Small* and *Medium* databases, however it leads to an infeasible solution on *Large* database.

6 Conclusion and future works

The overall goals and the obtained results of this paper were as follow:

1. There was not any efficient crossover method for the UCTP, thus more of literatures which used the GA for this problem, have withdrawn from crossover or have used a repair mechanism to modify infeasible solutions generated by classic crossover methods. This paper proposed a new crossover method called 2SNPC, by combination of classic n-point crossover and graph colouring

Table 6. The best results obtained by our proposed algorithms, i.e. the GA and HGAPSO, and other EAs.

Database	GA	HGAPSO	Ant (average)	DCABA	RI	LS (average)	EGSGA
Small1	6	**0**	1	5	**0**	8	**0**
Small2	8	1	3	5	**0**	11	**0**
Small3	7	**0**	1	3	**0**	8	**0**
Small4	8	1	1	3	**0**	7	**0**
Small5	3	**0**	**0**	**0**	**0**	5	**0**
Medium1	187	175	195	176	242	199	**139**
Medium2	202	184	184	154	161	202.5	**92**
Medium3	252	205	248	191	265	77.5% Inf	**122**
Medium4	224	176	164.5	148	181	177.5	**98**
Medium5	268	180	219.5	166	151	100% Inf	**116**
Large	100% Inf	100% Inf	851.5	798	100% Inf	100% Inf	615

Table 6 continued

Database	NLGD	GAWLS	HEA	HGA Rossi-Doria	VNS-Tabu	MA
Small1	3	2	**0**	**0**	**0**	**0**
Small2	4	4	**0**	3	**0**	**0**
Small3	6	2	**0**	**0**	**0**	**0**
Small4	6	**0**	**0**	**0**	**0**	**0**
Small5	**0**	4	**0**	**0**	**0**	**0**
Medium1	140	254	221	280	317	227
Medium2	130	258	147	188	313	180
Medium3	189	251	246	249	357	235
Medium4	112	321	165	247	247	142
Medium5	141	276	135	232	292	200
Large	876	1027	**529**	100% Inf	100% Inf	100% Inf

Note: the reported results for local search and ant algorithm are average of obtained results, but the reported results for other approaches are the best results.

heuristics. The GA using this crossover method obtained some competitive results than other proposed methods.

2. In this paper, to improve the obtained solutions by non-hybrid algorithms, a LS based on HC method was applied to three global search algorithms i.e. the GA, the PSO and the GAPSO and three hybrid algorithms were obtained i.e. the HGA, the HPSO and the HGAPSO.

3. Experimental results on 11 well-known benchmark problems demonstrated that among compared non-hybrid algorithms, the GA obtained the best results and outperformed the PSO and the GAPSO algorithms. Also to the best of our knowledge, the obtained results of GA were the first reported results on these databases in the literatures which were competitive with results of other approaches. Also among hybrid algorithms, the HGAPSO gives the best results.

4. In final part of comparison study, a comparison of our proposed algorithms with some approaches reported in the literature was carried out. The obtained results demonstrated that by considering the HGAPSO as a hybrid hyper-heuristic, it was one of the best strategies for hyper-heuristic systems on the UCTP proposed so far. Also results of the HGAPSO in comparison of other hybrid algorithms proposed in the literature were completely comparable.

However our proposed algorithms obtained a comparable performance than other proposed approaches, but those did not find any feasible solution on *Large* database. This infeasible solution is due to initialization stage that leads to a huge number of infeasible assignments on this database. For the future work, it might be interesting to employ and examine some efficient initialization methods that generate less infeasible assignments. Also in crossover and mutation stage, we used a single and same graph colouring heuristic. It might also be interesting to employ more than one and different heuristics when assigning the unscheduled courses.

References:

[1] R. Qu, Case-based reasoning for course timetabling problems, PH.D. thesis, University of Nottingham, UK, 2002.

[2] S. Abdullah, Heuristic approaches for university timetabling problem, PH.D. thesis, University of Nottingham, UK, 2006.

[3] M.W. Carter, A comprehensive course timetabling and student scheduling system at the university of waterloo, In: Burke, E. K., Erben, W., eds.: *The Practice and Theory of Automated Timetabling: Selected papers from the third International Conference*, Lecture Notes in Computer Science, Springer-Verlag, Berlin, Vol.2079, 2000, pp. 64–82.

[4] M. Amintoosi,J. Haddadnia, Feature selection in a fuzzy student sectioning algorithm, In: Burke, E. K., Trick, M.,, eds.: *The Practice and Theory of Automated Timetabling V*: Selected Papers from 5th International Conference on the Practice and Theory of Automated Timetabling, Lecture Notes in Computer Science, Springer-Verlag, Pittsburg, USA, Vol.3616, 2005, pp. 147-160.

[5] P. De Causmaecker,P. Demeester, G.V. Berghe, A decomposed metaheuristic approach for a real-world university timetabling problem, *Eur. J. Oper. Res*, Vo.195, 2009, pp. 307–318.

[6] G. Lajos, Complete university modular timetabling using constraint logic programming, In: Burke, E. K., Ross, P., eds.: *The Practice and Theory of Automated Timetabling I*: Selected Papers from 1st International Conference on the Practice and Theory of Automated Timetabling, Lecture Notes in Computer Science, Springer-Verlag, Edinburgh, UK, Vol.1153, 1996, pp. 146–161.

[7] M. Henz, J. Würtz, Using Oz for college timetabling, In: Burke, E. K., Ross, P., eds.: *The Practice and Theory of Automated Timetabling I*: Selected Papers from 1st International Conference on the Practice and Theory of Automated Timetabling, Lecture Notes in Computer Science, Springer-Verlag, Edinburgh, UK, Vol.1153, 1996, pp. 162-177.

[8] H. Cambazard, F. Demazeau, N. Jussien, P. David, Interactively solving school timetabling problems using extensions of constraint programming, In: Burke, E. K., Trick, M., eds.: *The Practice and Theory of Automated Timetabling: Selected papers from the third International Conference*, Lecture Notes in Computer Science, Springer-Verlag, Vol.124, 2004, pp. 107–124.

[9] M.W. Carter, G. Laporte, Recent developments in practical examination timetabling, In: Burke E. K., and Ross, P., eds.: *The Practice and Theory of Automated Timetabling I:* Selected Papers from 1st International Conference on the Practice and Theory of Automated Timetabling, Lecture Notes in Computer Science, Springer-Verlag, Edinburgh, UK, Vol.1153, 1996, pp. 3-21.

[10] E.K. Burke, J.H. Kingston, D. De Werra, *Applications to timetabling*, In: Gross, J., Yellen, J., eds.: The Handbook of Graph Theory, Chapman Hall/CRC Press, 2004, pp. 445-474.

[11] M.W. Carter, G. Laporte, J.W. Chinneck, A general examination scheduling system, *Interfaces*, Vol.24, 1994, pp. 109-120.

[12] E.K.Burke,J.P. Newall, R.F. Weare, A simple heuristically guided search for the timetable problem, *In: proceedings of the International ICSC Symposium on Engineering of Intelligent System (EIS'98)*, Canada/Switzerland, 1998, pp. 574-579.

[13] H. Asmuni, E.K. Burke, J. Garibaldi, B. McCollum, Fuzzy multiple ordering criteria for examination timetabling, In: Burke, E. K., Trick, M., eds: *The practice and theory of automated timetabling (PATAT)*, Springer, Berlin, Vol.3616, 2005, pp. 334-353.

[14] E.K. Burke, J.P. Newall, Solving examination timetabling problems through adaption of heuristic orderings, *Ann. Oper. Res*. Vol.129, 2004, pp. 107-134.

[15] E.K. Burke, B. McCollum, A. Meisels, S. Petrovic,R. Qu, A graph-based hyper-heuristic for educational timetabling problem, *Eur. J. Oper. Res*, Vol. 176, 2007, pp. 177-192.

[16] N. Pillay, W. Banzhaf, G.V. Berghe, A study of heuristic combinations for hyper-heuristic systems for the uncapacitated examination timetabling problem, *Eur. J. Oper. Res*, Vol.197, 2009, pp. 482-491.

[17] A. Tripathy, School timetabling-a case in large binary integer linear programming, Manage. *Sci*, Vol.30, 1984, pp. 1473–1489.

[18] E.K. Burke, D.G. Elliman, R.F. Weare, A genetic algorithm for university timetabling, *In: proceedings of the Artificial Intelligence and Simulation of Behaviour (AISB)*, University of Leeds, Pittsburg, UK, 1994, pp. 334-353.

[19] A. Ergül, GA-based examination scheduling experience at Middle East Technical University, In: Burke, E. K., Ross, P., eds.*: The Practice and Theory of Automated Timetabling (PATAT) I*, Springer, Berlin, Vol. 1153, 1996, pp. 212- 226.

[20] N. Pillay, W. Banzhaf, An informed genetic algorithm for the examination timetabling problem, *Appl. Soft. Comput*, Vol.10, 2010, pp. 457-467.

[21] A. Hertz, Finding a Feasible Course Schedule Using Tabu Search, *Discrete Appl Math*, Vol.35, 1992, pp. 255-270.

[22] G.M. White, B.S. Xie, S. Zonjic, Using tabu search with longer-term memory and relaxation to create examination timetables, *Eur. J. Oper. Res*., Vol.153, 2004, pp. 80-91.

[23] C.H. Aladag, G. Hocaoglu, M.A. Basaran, The effect of neighborhood structures on tabu search algorithm in solving course timetabling problem, *Expert. Syst. Appl*. Vol.36, 2009, pp. 12349-12356.

[24] J.M. Thompson., K.A. Dowsland, Variants of simulated annealing for the examination

timetabling problem, *Ann. Oper. Res*, Vol.63, 1996, pp. 105-128.

[25] J.M. Thompson, K.A. Dowsland, A robust simulated annealing based examination timetabling system, *Comput. Oper. Res*, Vol.25,1996, pp. 637-648.

[26] D. Abramson, M. Krishnamoorthy,H. Dang, Simulated annealing cooling schedules for the school timetabling problem, *Asia. Pacc. J. Oper. Res*, Vol.16, 1999, pp. 1-22.

[27] J. Frausto-Solís, F. Alonso-Pecina, J. Mora-Vargas, An Efficient Simulated Annealing Algorithm for Feasible Solutions of Course Timetabling, In: Gelbukh, A., Morales, E. F., eds.: *MICAI 2008*, Springer, Berlin, Vol.5317, 2008, pp. 675–685.

[28] D. Zhang, L. Yongkai, R. M'Hallah, S.C.H. Leung, , A simulated annealing with a new neighborhood structure based algorithm for high school timetabling problems, *Eur. J. Oper. Res*, Vol.203, 2010, pp. 550-558.

[29] K. Socha, J. Knowles, M. Samples, A max-min ant system for the university course timetabling problem, *In: proceedings of the 3rd international workshop on ant algorithms (ANTS 2002)*, Springer, Vol.2463, 2002, pp. 1-13.

[30] K. Socha, M. Samples,. M. Manfrin, Ant algorithm for the university course timetabling problem with regard to the state-of-the art, *In: proceedings of the 3 European workshop on evolutionary computation in combinatorial optimisation*, Essex, UK, Springer, Vol.2611,2003, pp. 334-345.

[31] N. Ejaz, M. Javed, An Approach for Course Scheduling Inspired by Die-Hard Co-Operative Ant Behavior, *In: proceedings of the IEEE international conference on automation and logistics*, Jinan, china, 2007, pp. 3095-3100.

[32] O. Rossi-Doria, M. Sampels, M. Birattari, M. Chiarandini, M. Dorigo, L. Gambardella, J. Knowles, M. Manfrin, M. Mastrolilli, B. Paechter, L. Paquete, T. St¨utzle, A comparison of the performance of different metaheuristics on the timetabling problem, *In: Proc. 4th Int. Conf. Pract. Theory Automated Timetabling*, Lecture Notes in Computer Science, Vol.2740, 2003, pp. 329–351.

[33] P. Pongcharoena, W. Promtetb, P. Yenradeec, C. Hicks, Stochastic Optimisation Timetabling Tool for university course scheduling, *Int. J. Prod. Econ.*, Vol.112, 2008, pp. 903-918.

[34] D. Landa-Silva, J.H. Obit, Great deluge with non-linear decay rate for solving course timetabling problems, *In: Proc. 4th IEEE Int. Conf. Intell. Syst*., 2008, pp. 811–818.

[35] R. Lewis, A survey of metaheuristic-based techniques for university timetabling problems, *OR. Spectrum*., Vol.30, 2008, pp. 167-190.

[36] S.C. Kong, L.F. Kwok., A conceptual model of knowledge-based timetabling system, *Knowl-Based. Syst*., Vol.12, 1999, pp. 81-93.

[37] L.R. Foulds, D.G. Johinson, SlotManager: a microcomputer-based decision support system for university timetabling, *Decis. Support. Syst*., Vol.27, 2000, pp. 307–381.

[38] J.L. Kolodner, Case-based reasoning. Morgan Kaufman Publishers, Inc. San Mateo 1993.

[39] E.K. Burke, B. MacCarthy, S. Petrovic, R. Qu, Structured Cases in CBR – Re-using and Adapting Cases for Timetabling Problems, *Knowl-Based. Syst*. Vol.13, 2000, pp. 159-165.

[40] E.K. Burke, S. Petrovic, R. Qu, Case based heuristic selection for timetabling problems, *J. Scheduling*., Vol.9, 2006, pp. 115–132.

[41] E.K. Burke, E. Hart, G. Kendall, J.P. Newall, P. Ross, S. Schulenburg, Hyper-heuristics: An emerging direction in modern search technology, In: Glover, F., Kochenberger, G., eds.: Chapter 16 in Handbook of Meta-Heuristics, Kluwer, 2003, pp. 457-474.

[42] R. Qu, E.K. Burke, Hybridisations within a Graph Based Hyperheuristic Framework for University Timetabling Problems, Technical Report NOTTCS-TR-2006-*1*, School of CSiT, University of Nottingham, UK, 2006.

43] H. Asmuni, E.K. Burke, J.M. Garibaldi, Fuzzy multiple heuristic ordering for course timetabling, *In: proceedings of the 5th united kingdom workshop on computational intelligence (UKCI05)*, London, UK, 2003, pp. 302-309.

[44] H. Asmuni, E. K. Burke, J.M. Garibaldi, A comparison of fuzzy and non-fuzzy ordering heuristics for examination timetabling, *In: Proceedings of the 5th International Conference on Recent Advances in Soft Computing*, Nottingham, UK, 2004, pp. 288-293.

[45] S. Petrovic, V. Pate, Y. Yang, University timetabling with fuzzy constraints, In: Burke, E. K., Trick, M.,, eds.: *The Practice and Theory of Automated Timetabling I*: Selected Papers from 5th International Conference on the Practice and Theory of Automated Timetabling, Lecture Notes in Computer Science, Springer-Verlag, Pittsburg, USA, Vol.3616, 2005, pp. 313-333.

[46] A. Chaudhuri, K. De, Fuzzy genetic heuristic for university course timetable problem, Intern.

J. Adv. Soft. Comput. Appl., Vol.2, 2010, pp. 100-123.

[47] K.A. Smith, D. Abramson, D. Duke, Hopfield neural networks for timetabling: formulations, methods, and comparative results, *Comput. Ind. Eng*., Vol.44, 2003, pp. 283-305.

[48] M.P. Carrasco, M.V. Pato, A comparison of discrete and continuous neural network approaches to solve the class/teacher timetabling problem, *Eur. J. Oper. Res*., Vol.153, 2004, pp. 65-79.

[49] S. Daskalaki, T. Birbas, Efficient solutions for a university timetabling problem through integer programming, *Eur. J. Oper. Res*., Vol.160, 2005, pp. 106-120.

[50] K. Schimmelpfeng, S. Helber, Application of a real-world university-course timetabling model solved by integer programming, *OR. Spectrum*, Vol.29, 2007, pp. 783-803.

[51] S. Abdullah., E.K. Burke, B. McCollum, An investigation of a variable neighbourhood search approach for course timetabling, *In: proceedings of the 2nd multidisciplinary international conference on scheduling: theory and applications (MISTA 2005)*, New York, USA, 2005, pp. 413-427.

[52] S. Abdullah, E.K. Burke, B. McCollum, Hybrid evolutionary approach to the university course timetabling problem. *In proceedings of the IEEE congress on evolutionary computation (CEC 2007)*, 2007, pp. 1764-1768.

[53] A. Schaerf, A survey of automated timetabling. *Artif. Intell. Rev*., Vol.13, 1999, pp. 87-127.

[54] R. Qu, E.K. Burke, B. McCollum, L.T.G. Merlot, S.Y. Lee, A survey of search methodologies and automated approaches for examination timetabling, Computer Science Technical Report No. NOTTCS-TR-2006-4, School of Computer Science and Information Technology, University of Nottingham, UK, 2006.

[55] S. Petrovic, E.K. Burke, University timetabling, In: Leung J ed.: Chapter 45 in Handbook of Scheduling: Algorithms, Models, and Performance Analysis, CRC Press, 2004.

[56] E.K. Burke, S. Petrovic, Recent research directions in automated timetabling, *Eur. J. Oper. Res*., Vol.140, 2002, pp. 266-280.

[57] S.C. Chu, H.L. Fang, Genetic algorithms vs tabu search in timetable scheduling, In: proceedings of the 3rd international conference on knowledge-based intelligent information engineering *system*, Australia, Adelaide, 1999, pp. 492-495.

[58] P. Ross, E. Hart, D. Corne, Some observations about GA based timetabling, In: Burke, E. K., Carter M eds.*: The Practice and Theory of Automated Timetabling (PATAT) II*, Springer, Berlin, Vol.1408, 1998, pp. 115-129.

[59] E.K. Burke, J.P. Newall, R.F. Weare, A memetic algorithm for university exam timetabling, In: Burke, E. K., Ross, P. eds.: *Practice and theory of automated timetabling (PATAT) I*, Springer, Berlin, Vol.1153, 1996, pp. 241-250.

[60] L.T.G. Merlot, N. Boland, B.D. Hughes, O.J. Stuckey, A hybrid algorithm for the examination timetabling problem, In: Burke, E. K., De Causmaecker P eds.: *Practice and theory of automated timetabling (PATAT) IV*, Springer, Berlin, Vol.2740, 2003, pp. 207-231.

[61] Z.N. Azimi, Hybrid heuristics for examination timetabling problem, *Appl. Math. Comput*., Vol.163, 2005, pp. 705-733.

[62] M. Chiarandini, M. Birattari, K. Socha, O. Rossi-Doria, An effective hybrid algorithm for university course timetabling, *J. Scheduling*., Vol.9, 2006, pp. 403-432.

[63] S. Yang, S.N. Jat, Genetic algorithms with guided and local search strategies for university course timetabling, *IEEE. T. Sys. Man. CY*. C, Vol.41, 2011, pp. 93-106.

[64] S. Abdullah, H. Turabieh, Generating university course timetable using genetic algorithm and local search, *In: Proc 3rd Int Conf Hybrid Inform Tech*, 2008, pp. 254–260.

[65] S. Abdullah, E.K. Burke, B. McCollum, H. Turabieh, A hybrid evolutionary approach to the university course timetabling problem, *In Proc 2007 Congr Evoll Comput*, 2007, pp. 1764–1768.

[66] S.N. Jat, S. Yang, A Memetic algorithm for the university course timetabling problem, In*: 2008 20th IEEE International Conference on Tools with Artificial Intelligence* DOI 10.1109/ICTAI.2008.126, 2008.

[67] E.K. Burke, D.G. Elliman, R.F. Weare, A university timetabling system based on graph colouring and constraint manipulation, J. Res. Comput. Educ., 27, pp. 1-18, 1994.

[68] Burke, E. K., Kingston, J., De Werra, D., Applications to timetabling, In Gross J, Yellen J eds.: Handbook of Graph, Chapman Hall/CRC Press, Vol.24, 2004, pp. 445-474.

[69] R.L. Haupt, S.E. Haupt, Practical genetic algorithms. John Wiley & Sons, 2004, pp. 38-41.

[70] P. Ross, D. Corne, H.L. Fang, Improving evolutionary timetabling with delta evolution and directed mutation, *In: proceedings of the*

Conference on parallel problem solving from nature III, Springer, Berlin, Vol.866, 1994, pp. 556-565.

[71] S. Abdullah, E.K. Burke, B. McCollum, Using a randomised iterative improvement algorithm with composite neighbourhood structures for the university course timetabling problem, In: Doerner, F. K., Gendreau, M., Greistorfer, P., Gutjahr, W. J., Hartl, R. F., and Reimann, M., eds. Metaheuristics - Progress in Complex Systems Optimization, Computer Science Interfaces Book Series, Springer Operations Research, ISBN-13, pp. 978-0-387-71919-1, Vol.39, 2007, pp. 153-169.

[72] D. Corne, H.L. Fang, C. Mellish, Solving the modular exam scheduling problem with genetic algorithms, *In: proceedings of the 6th international conference of industrial and engineering applications of AI and expert system*, 1993, pp. 370-373.

[73] V. Bhatt, R. Sahajpal, Lecture timetabling using hybrid genetic algorithms, *In: proceedings of 2004 international conference on intelligent sensing and information processing*, 2004, pp. 29-34.

[74] Q.K. Pan, M.F. Tasgetiren, Y.C. Liang, A discrete particle swarm optimization algorithm for the no-wait flowshop scheduling problem, *Comput. Oper. Res*., Vol.35, 2008, pp. 2807-2839.

[75] E.K. Burke, G. Kendall, E. Soubeiga, A tabu-search hyper-heuristic for timetabling and rostering, *J. Heuristics*., Vol.9, 2003, pp. 451-470.

[76] H. Asmuni, E.K. Burke., J.M. Garibaldi, A hybrid approach for course scheduling inspired by die-hard co-operative ant behavior, *In proceedings of the IEEE international conference on automation and logistics*, Jinan, China, 2007, pp. 3095-3100.