Computers & Industrial Engineering 86 (2015) 43–59

Contents lists available at ScienceDirect

Computers & Industrial Engineering

journal homepage: www.elsevier.com/locate/caie

Review

A survey of approaches for university course timetabling problem

Hamed Babaei [a,*], Jaber Karimpour [b], Amin Hadidi [c]

[a] Department of Computer Engineering, College of Engineering, Ahar Branch, Islamic Azad University, Ahar, Iran
[b] Department of Computer Sciences, University of Tabriz, Tabriz, Iran
[c] Department of Mechanical Engineering, College of Engineering, Ahar Branch, Islamic Azad University, Ahar, Iran

ARTICLE INFO

Article history:
Available online 21 November 2014

Keywords:
University course timetabling problem (UCTTP)
Multi-agent based approach (Cooperative Search)
Operational research methods
Metaheuristic methods

ABSTRACT

Scheduling is one of the problems which so many researches have been conducted on it over the years. The university course timetabling problem which is an NP-hard problem is a type of scheduling problem. Timetabling process must be done for each semester frequently, which is an exhausting and time consuming task. The allocation of whole of events in timeslots and rooms performs by the university course timetabling process considering the list of hard and soft constraints presented in one semester, so that no conflict is created in such allocations. In the university course timetabling problem (UCTTP), the hard constraints should not be violated under any conditions; soft constraints also should not be violated as much as possible. The aim of the present paper is to analyze available approaches in the study of university course timetabling problems, including operational researches, metaheuristic methods and intelligent novel methods; also the distributed multi agent systems based approach (Cooperative Search method) is investigated due to its scalability which enables the timetabling of common events between departments. In addition, in this work a complete introduction of reliable datasets has been given to test and evaluation of the structure of considered algorithms.

1. Introduction

The goal of the university course timetabling problem (UCTTP) is to find a method to allocate whole events to fix predefined timeslots and rooms, where all constraints within the problem must be satisfied. Events include students, teachers and courses where resources encompass the facilities and equipment's of classrooms such as theoretical and practical rooms. Also timeslots include two main components, namely daily and weekly timeslots which it varies from one institution to another. However, each classroom also has its own components including audio-visual equipment's (video projector), number of chairs necessary for courses allocated to those classrooms (the capacity of theory and practical rooms), number of blackboards and whiteboards related to each theory and practice classroom and etc.

1.1. Scope and purpose

Object and method of this research in review of University Course Timetabling Problem is presented in Fig. 1.

1.2. Description of the problem

UCTTP is a hybrid optimization problem in the class of NP-hard problems occur at the beginning of each semester of universities and includes the allocation of events (courses, teachers and students) to a number of fixed timeslots and rooms. This problem must satisfy both hard and soft constraints during allocation of events to resources, so that the possible timetables are obtained after full satisfaction of whole hard constraints and also soft constraints to increase and promote the quality of possible generated timetables as necessary (Asmuni, 2008; Obit, 2010; Redl, 2004).

There are some problems and complexities in UCTTP process; firstly, the scheduling process is an NP-complete problem, then it could not be solved in the polynomial time classes because of the exponential growth of this problem and the existence of some variations in the fast growth of students' numbers in this problem, so we must seek heuristic approaches. Secondly, the number of constraints (hard and soft) in this problem differs from one institution to another. Therefore, the main aim of all of the mentioned algorithms is to maximize the number of soft constraints satisfied in the final timetables (Feizi-Derakhshi, Babaei, & Heidarzadeh, 2012; Obit, 2010).

* Corresponding author. Tel.: +98 937 056 4179.
E-mail addresses: hamedbabaei63@gmail.com, h-babaei@iau-ahar.ac.ir (H. Babaei), karimpour@tabrizu.ac.ir (J. Karimpour), amin.hadidi@yahoo.com, a-hadidi@iau-ahar.ac.ir (A. Hadidi).

http://dx.doi.org/10.1016/j.cie.2014.11.010
0360-8352/© 2014 Elsevier Ltd. All rights reserved.

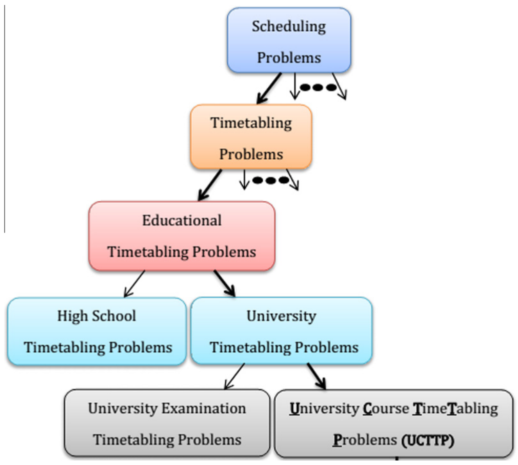
Scheduling Problems
Timetabling Problems
Educational Timetabling Problems
High School Timetabling Problems
University Timetabling Problems
University Examination Timetabling Problems
University Course TimeTabling Problems (UCTTP)

Fig. 1. Diagram of university course timetabling problem.

1.3. The basic definitions of the problem

- *Event:* a scheduled activity, like: teacher, course, and student.
- *Timeslot:* a time interval in which each event is scheduled, like: weekly timeslot such as Tuesday and daily timeslot such as 8–9 a.m. and etc.
- *Resource:* resources are used by events, like: equipment's, rooms, timeslots and etc.
- *Constraint:* a constraint is a restriction in scheduling of events, categorized into two types of hard and soft constraints, like the capacity of classrooms, given timeslot and etc.
- *People:* people include lecturers, students and are a part of events.
- *Conflict:* the confliction of two events with each other, like: scheduling of more than one teacher for one classroom at the same time.

1.4. Different types of constraints in the problem

Constraints in UCTTP problem are classified into two classes of hard and soft constraints. Hard constraints must be satisfied in the problem completely so that the generated solution would be possible and without conflict; no violation is allowed in these constraints. Soft constraints are related to objective function; objective function is to maximize the number of satisfied soft constraints. Unlike hard constraints, soft constraints are not necessarily required to satisfy; but as the number of these satisfied constraints increases, the quality of solutions of objective function increases. In the following, a list of hard and soft constraints presented which are taken from literature (Asmuni, 2008; Feizi-Derakhshi et al., 2012; Gotlib, 1963; Lewis, 2006; Obit, 2010; Redl, 2004).

1.4.1. Hard constraints

- A teacher could not attend two classes at the same time.
- A course could not be taught in two different classes at the same time.
- A teacher teaches only one course in one room at each timeslot.
- At each daily timeslot in one room only one group of students and one teacher could attend.
- A teacher teaches for only one group of students at each daily timeslot.
- There are some predefined courses which are scheduled in a given timeslots.
- The capacity of the classrooms should be proportional to the number of students of the given course.

1.4.2. Soft constraints

- The teacher can have the choice to suggest priority certain timeslots for her/his courses either public or private times.
- A teacher may request a special classroom for a given course.
- The courses should be scheduled in a way that the empty timeslots of both teacher and student to be minimized.
- Timetabling of the courses should be conducted in a way that the courses not scheduled at evening timeslots, as it is possible; unless an evening timeslot has been requested by a particular teacher.
- The lunch break is either 12–13 p.m. or 13–14 p.m., usually.
- The start time of classes may be 8 a.m. and the ending time may be 20:30 p.m. (evening), usually.
- The maximum teaching hours for teachers in a classroom are 4 h.
- The maximum learning hours for students is 4 h.
- Scheduling should be conducted in a way that one or a group of students not attend university for one timeslot in a day.

1.5. Mathematical formulation of the problem

Formal definition of UCTTP problem includes n: the number of events $E=\{e_1, e_2, \ldots, e_n\}$, k: the number of timeslots $T=\{t_1, t_2, \ldots, t_k\}$, m: the number of rooms $R=\{r_1, r_2, \ldots, r_m\}$, L: the number of rooms' features $F=\{f_1, f_2, \ldots, f_l\}$ and s: the set of students $S=\{s_1, s_2, \ldots, s_s\}$. For example, if the number of daily timeslots is 9 and the number of weekly timeslots is 5, then the total timeslots will be $T = 9 \times 5 = 45$ (Asmuni, 2008; Obit, 2010; Redl, 2004; Wangmaeteekul, 2011).

The input data for each sample problem (data sets) include the size and features of each room, the number of students in an event and information about conflicting events. So, we should know the procedure of measuring violation and non-violation of hard and soft constraints in order to have the ability to replace events within matrixes. At first the penalty function per violation from soft constraint must be calculated for each solution which is corresponding to a timetable, as bellow (Asmuni, 2008; Obit, 2010; Redl, 2004; Wangmaeteekul, 2011):

$$PF(S) = \sum_{j=1}^{SC} W_j \times (-1) \tag{1}$$

In Eq. (1), S is the solution, W_j is the weight of each soft constraint (value 0 means non-violation, value 1 means violation and -1 shows the cost of each violation per soft constraint) and SC is the number of soft constraints. However, PF represents the penalty function. Value of objective function per solution considering hard constraints can be calculated as:

$$\mathrm{OF}(S) = \sum_{i=1}^{HC} W_i \times (-1) + \mathrm{PF}(S) \tag{2}$$

In Eq. (2), W_i is the weight of each hard constraint where value 0 means non-violation, value 1 means violation and -1 shows the cost of each violation per hard constraint. Also HC and OF are the number of hard constraints, and the objective function, respectively. Always the value of first term of right hand side of the Eq. (2) is equal to zero ($\sum_{i=1}^{HC} W_i \times (-1) = 0$), this means that the violation of hard constraints is not feasible. So $OF(S) = 0 + PF(S)$, consequently $OF(S) = PF(S)$.

In order to determine the violation of solutions, from hard and soft constraints, results of sample problems are stored in five matrixes namely STUDENT-EVENT, EVENT-CONFLICT, ROOM-FEATURES, EVENT-FEATURES and EVENT-ROOM which is introduced in the following.

Each event is met by each student which is stored in the matrix STUDENT-EVENT. This matrix called matrix A is a $k \times n$ matrix. If the value of $U_{i,j}$ in the matrix $A_{k,n}$ be 1, then student $i \in S$ must attend event $j \in S$, otherwise, its value will be 0. The matrix size is $k \times n = |S| \times n$. The EVENT-CONFLICT matrix is an $n \times n$ matrix with two arbitrary events which could be scheduled in the same timeslots. This matrix called matrix B is used to quickly identify events which potentially allocated to same timeslots. ROOM-FEATURES matrix is a $m \times l$ matrix which shows the features of each room; this matrix called matrix C. If the value of $C_{i,j}$ be 1, then each $i \in R$ has a feature of $j \in F$, and otherwise its value will be 0. The matrix size is $m \times l = m \times |F|$. The EVENT-FEATURE matrix also called matrix D is a $n \times l$ matrix and represents the features required by each event. Namely, event $i \in E$ requires features of $j \in F$, if and only if $d_{i,j} = 1$. The matrix size is $n \times l = n \times |F|$. Finally the EVENT-ROOM matrix called G matrix is an $n \times m$ matrix which represents the list of possible rooms so that each event could be allocated in those rooms. This matrix represents the quick identification of all rooms in terms of their size and features for each appropriate event. The matrix size is $n \times m$ (Asmuni, 2008; Feizi-Derakhshi et al., 2012; Lewis, 2006; Obit, 2010; Redl, 2004).

1.6. The approaches used in the study of UCTTP

The first definition of timetabling has been presented as three sets of: (1) teachers, (2) classrooms and (3) timeslots (Gotlib, 1963). Approaches used to solving the UCTTP problem up to now are as follows: (1) Operational Researches (OR) based techniques including Graph Coloring (GC) theory based technique, Integer/Linear programming (IP/LP) method and Constraint Satisfaction(s) Programming (CSPs); (2) Metaheuristic approaches also including Case Base Reasoning method (CBR), population based approaches and single solution based approaches where the population based approaches includes Genetic Algorithms (GAs), Ant Colony Optimization (ACO), Memetic Algorithm (MA), Harmonic Search Algorithm (HAS), Partial Swarm Optimization (PSO), Artificial Bee colony Optimization (ABC) and single solution algorithms also includes Tabu Search Algorithm (TS), Variable Neighborhood Search (VNS), Randomized Iterative Improvement with Composite Neighboring algorithm (RIICN), Simulated Annealing (SA) and Great Deluge Algorithm (GD); (3) multi criteria and multi objective approaches; (4) intelligent novel approaches such as hybrid approaches, artificial intelligence based approaches, fuzzy theory based approaches, Clustering Algorithm based approaches and (5) distributed multi agent systems approach (Asmuni, 2008; Feizi-Derakhshi et al., 2012; Lewis, 2006; Redl, 2004).

1.7. Motivation and historical perspective of the problem

Agents are technologies inspired from global environment to develop initial instances of systems. Whenever a distributed multi agent system is considered, it means that there is a network of agents collaborates with each other to solve problems which are out of capability of each single agent (Srinivasan, Singh, & Kumar, 2011). Recently, using distributed multi agent systems based approach to solve UCTTP problem has been applied by Obit, Landa-Silva, Ouelhadj, Khan Vun, and Alfred (2011) where in the this method, a solution is used to deal with UCTTP problem using distributed environment and an interface agent -which is responsible to cooperate different timetabling agents- collaborate with each other to improve the solution of common goal. The initial timetables are generated for multi agent systems by using multiple hybrid metaheuristics which are a combination of graph coloring metaheuristics and local search in different methods. The hybrid metaheuristics provide the capability to generate possible solutions for all samples of both Socha, Knowles, and Samples (2002) and International Timetabling Competitions 2002 (ITC-2002) datasets. However, recently, Wangmaeteekul (2011) has used distributed agents to create UCTTP by considering hard (necessary) and soft (desirable) constraints. Also, he presented fairly meeting of distribution in allocating resources in his Ph.D. thesis. There are two types of agents in that model which are year-programmer agent and rooms' agent. However, there are four principles to efficiently organize agents, including: (1) queue and the sequential queue algorithm, (2) queue and interleaved queue algorithm, (3) round robin and sequential round robin algorithm and (4) round robin and interleaved round robin algorithm. The problem formulation and dataset have been adopted from the third section of International Timetabling Competitions 2007 (ITC-2007) datasets. The obtained result ensures the consistency of interleaved round robin principle for year-programmer agents in the system and the fairest chance in obtaining the required resources.

1.8. Aim of the paper

In this paper the aim is to first review approaches in solving UCTTP problem in detail and then to introduce distributed multi agent systems based approach to solve UCTTP problem. Also researches on application of distributed multi agent approach to create infrastructure of scheduling common events among multiple departments have been reviewed. In addition, in this paper approaches used in solving UCTTP is classified in four categories as:

- Operational Research (OR) methods,
- metaheuristic methods,
- multi-objective and multi-criteria approaches,
- intelligent novel methods,
- the approach based on distributed multi agent systems (Cooperative Search).

Also, in these research relevant data sets for examination of algorithms' structure is investigated.

1.9. Paper outline

The remainder of this paper is organized as follows: First, in Section 2, we review the related works in UCTTP problem. However, Section 3 will describe multi agent systems based approach to solve UCTTP problems in detail. Section 4 includes the study of experiments and results of other works using available algorithms, Section 5 includes the discussions and final section encompasses the conclusion.

2. Related works

Approaches which are used to study the UCTTP problem up to now surveyed in this section.

2.1. Operational research methods

This approach includes the technique based on graph coloring theory, IP/LP method and constraint based technique (CSPs) explained in the following.

2.2. A method based on graph coloring theory

The first definition of timetabling problem has been introduced by Gotlib (1963) as three sets of lecturers, classrooms and timeslots. The first timetabling problem has been solved using graph coloring problem (Welsh & Powell, 1967). However, this presented method was not able to solve the problems when there were pre-assigned sessions. De Werra (1985) had described the graph coloring method to modeling a timetabling problem by using a non-directional graph. Here the aim is to color the graph by using a given number of colors where no adjacent vertices (nodes) have the same color. The resulted timetable should not have a conflict and it should be colored with the least number of colors. In the graph coloring theory based timetabling problem, the events, constraints (preferably hard constraints) and timeslots are considered as nodes, edges and colors, respectively. Nodes, edges and colors in this theory are shown in Fig. 2. The chromatic number is the least number of colors which is necessary to color the nodes and edges of the graph. Colored nodes are equivalent to the number of timeslots. So we can schedule the events in a timeslot in one day using this coloring; in the other words, if two adjacent nodes have the same color, then time conflict will be occurs.

However, Selim (1988) has introduced the idea of separated graph vertices to reduce the chromatic number of graphs and has applied it to separate students. Here, the separation of one vertex is similar to separation of students in one course. The method of coloring the edges of two part graph also has been conducted by Hafizah and Zaidah (2010) in order to reduce the number of penalties and create high quality timetables compared with manual timetables. The scheduling of classrooms also has been performed using the graph coloring method by Dandashi and Al-Mouhamed (2010) where vertices and edges represent common courses and students, respectively, and the aim is to present a VC* heuristic approach in order to: (1) promote the uniform distribution of courses over colors and (2) balance the number of courses for each timeslot over the existing rooms. Another hybrid approach to solve UCTTP problem using genetic coloring has been proposed by Asham, Soliman, and Ramadan (2011) where this method reduces the cost of finding the least number of required colors to color a graph.

We can conclude that graph coloring theory could perform a straightforward implementation of the problem and generate conflict free timetables in the output; but since finding the required number of colors to color the nodes of the graph has NP-hard time, so this method does not have efficient performance, unless it is combined with metaheuristic approaches to increase the performance.

2.2.1. IP/LP method (Integer programming/Linear programming)

Feiring, had defined the Linear Programming (LP) as a subset of mathematical programming where this aim has been followed by efficient assigning of limited resources to the specified activities in order to maximize the interest and minimize the cost. Bunday, had also defined the Integer Programming (IP), here all or some of the variables could be defined just as non-negative integer values (Lewis, 2006; Obit, 2010).

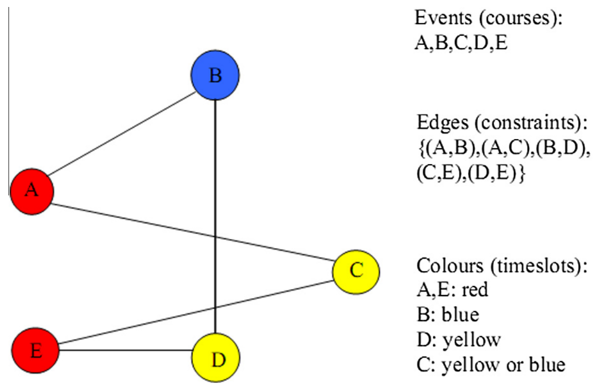

Fig. 2. The timetabling problem represents on graph coloring method (De Werra, 1985).

IP/LP method is a complete mathematical method and has been applied based on the structure and type of faculty. Operator of IP/LP method interviews with the director and dean of faculty. Solutions of the UCTTP problem using this method are different for education institutes; the size of the institute has great influence on solving this problem so that it is difficult to obtain the optimal solutions at the end of performing an IP/LP method for UCTTP problem. This method is mostly used separately without combination with other approaches; however we could use some constructive heuristics within this method which facilitates the analysis of constraints. Therefore, the general structure of IP/LP method in UCTTP problem is presented in following.

The formulation in IP/LP is performed based on different groupings in two classes on courses grouping and time intervals grouping. At first, the course groups should be defined and then courses are selected in subject groups by the corresponding students and these subject groups should be scheduled at different timeslots. Time group is also in four-hour groups and divided into two two-hour timesolts. The courses allocated to the mentioned time groups are defined as a cluster, i.e. a group of courses is scheduled over a given timeslots (Dimopoulou & Miliotis, 2001). Aubin and Ferland (1989) have developed a more general procedure to deal with large scale timetabling problems. They have separated the timetabling problem into two sub problem which are called timetabling sub problem and grouping sub problem. They have proposed a heuristic approach to solve this problem through simultaneous applying of sub problems of timetabling and grouping, until a final solution is reached and improved as much as possible.

Daskalaki, Birbas, and Housos (2004) have presented the IP (Integer Programming) method for solving the UCTTP problems where the aim is to allocate a set of courses among teachers and groups of students and also a set of weekly and daily timeslots pairs. Again, Daskalaki and Birbas (2005) have presented an IP-based two-step relaxation method to generate efficient solutions of timetabling in two steps. During step 1, the sessions of courses which should be held consecutively have been scheduled through allocation of courses to days and special times. During step 2, it should be ensured that the courses (presented to the same group of students) require more than one timeslots have been scheduled consecutively. An integer programming (IP 0/1) approach has been presented by Bakir and Aksop (2008) to organize courses and teachers, according to available timeslots and classrooms which results in reduction of dissatisfaction of students and teachers during the implementation of finite rules by a set of constraints simultaneously.

2.2.2. Constraint satisfaction programming (CSP) based method

The constraint satisfaction programming (CSP) based method is a computing based system where a constraint could be defined as a limitation over a space of facilities. The aim of this method is to find a set of consistent values where each of these values could be assigned to the values of variables and satisfy the predefined constraints. This problem is a function of three variables as $CSP = (X, D, C)$ where X is a finite set of variables as $X = x_1, x_2, \ldots, x_n$ and D is a finite set of domain values as, $D = d_1, d_2, \ldots, d_n$ so that each of these variables could be selected from this domain and C is a finite set of constraints as $C = c_1, c_2, \ldots, c_m$. Constraints also depend on a subset of variables. The final solution is to assign values to each variable where these assignments could satisfy the whole of given constraints (Lewis, 2006; Obit, 2010). Deris, Omatu, Ohta, and Saada (1999) had combined a genetic algorithm with constraint based reasoning and they presented a possible and near to optimum solution to the course timetabling problem. Deris, Omatu, and Ohta (2000) have used a timetabling planning problem using the constraint based reasoning technique in an object oriented approach.

However, ILOC software has been applied to implement the CSP approach by Zhang and Lau (2005) performed to build timetabling in university. The results of this software are to reach an objective function due to satisfaction of events' constraints in allocating to resources.

2.3. Metaheuristic methods

Metaheuristic approaches include two main methods of population based and single-solution discussed in this section.

2.3.1. Population based metaheuristic approaches

In population based method, at first we have a number of people or initial solutions where this set of people is called initial population. At each iteration of population based metaheuristic approaches, a selection mechanism is used to select the best solution(s) from the present population, then, according to the selected metaheuristic method, some changes are applied over the selected solutions so that the improvement is obtained in the solution, now these improved solutions are replaced with damaged people. This procedure continues until reaching a desirable solution. This method includes the following algorithms: (1) evolutionary and genetic algorithms, (2) ant colony algorithm, (3) memetic algorithm, (4) harmony search algorithm, (5) Partial Swarm Optimization and (6) Artificial Bee colony Optimization (Lewis, 2006; Obit, 2010).

2.3.1.1. Evolutionary and genetic algorithms. Evolutionary algorithms (EAs) are based on computing model and inspired of natural evolutionary mechanism. EAs act on a population of possible solutions and include three steps: (1) selection, (2) regeneration and (3) replacement. In the selection phase: people with high fitness are selected to be parents for the next generation. In the regeneration phase: two crossover and mutation operators are performed on parents who were selected in the first phase and in the replacement phase: the people of original (initial) population are replaced by the newly created people.

Genetic algorithms which are a subset of EAs are based on the following steps: (1) to generate the initial population, (2) to evaluate the generated population by using the evaluation function, (3) to select some people as parents to crossover based on the obtained information from the evaluation functions, (4) to apply crossover operator to produce children, (5) to apply mutation operator for children, (6) to select parents and children to form the new population for the future generation and (7) if the termination condition is satisfied, the algorithm stops else it goes to second step and continues (Obit, 2010).

Khonggamnerd and Innet (2009) have been used a genetic algorithm to sort a university timetables where the crossover rate was 70% and however, no hard constraint has been violated in timetabling. Used constraints result in more occupation of rooms and capacity of the rooms. However, Alsmadi, Abo-Hammour, Abu-Al-Nadi, and Algsoon (2011) have proposed a novel GA to solve the UCTTP problem which uses a learner machine. The results of this technique are to minimize the number of violated soft constraints, maximize the use of available rooms and reduce of teachers' workload.

2.3.2. Ant colony optimization algorithm

This method has inspired by the ants' behavior to find a route between the place of formicary and food. Ants always move randomly to find food and then they place a track of pheromone. Other ants when find this route, follow it and if they reach the food, they return to their home and place a track besides the previous track. The pheromone evaporates slightly where results in the route would become less attractive to the next ant; therefore the random search is limited to that food. The aim of this approach is the movement of artificial ants to find the shortest route (Obit, 2010).

Using Max–Min ant system to generate university courses timetabling by Socha et al. (2002) has led to building an optimal path where each path could generate a constructive graph to allocate courses to timeslots affected by the amount of pheromone within a range. However, applying ant colony optimization algorithm by Mayer, Nothegger, Chwatal, and Raidl (2008) for the UCTTP problem has been post enrolled according to ITC-2007 dataset where ants allocate events to rooms and timeslots based on two types of pheromone T_{ij}^{s} and T_{ij}^{y}. This algorithm has good performance and leads to good results with high run time. Applying a hybrid ant colony system has been proposed by Ayob and Jaradat (2009) to solve the UCTTP problem. Here, two types of hybrid ant system, including a combination of simulated annealing (SA) with ant colony (AC) and tabu search (TS) with AC have been presented. A number of ants perform the complete allocation of courses to timeslots based on a predefined list. The selection of timeslots probabilities by ants to allocate courses has been done by using heuristic information and information of an indirect coordinator mechanism among agents and activities within an environment.

2.3.2.1. Memetic algorithm. We can consider a meme as a unit of information which generates itself when the beliefs of people change. A meme is different from a gene due to this fact that when a meme is transmitted among people, each person adapts the meme if it seems good, while the gene is transmitted without any changes. In memetic algorithm, the space of possible solutions is reduced to the local optimization of sub spaces which are one of the best advantages of this method. Memetic algorithm is a combination of genetic and hill climbing algorithm (Obit, 2010).

The memetic algorithm has been performed by Jat and Shengxiang (2008) to solve the UCTTP problem through a combination of local search method in genetic algorithms. One of the local searches has been run over the events and another one over timeslots.

2.3.3. Single solution based metaheuristic approaches

This method is also one of the metaheuristic methods which uses a single solution to analysis the problem instead of using an initial population to solve an optimization problem, so that at first a single solution is selected based on some criteria and until the termination of the problem solving process, this single solution is manipulated and relocated until the final improvement of final solution is obtained. The termination step occurs when the final condition and criterion is satisfied. This approach includes the methods surveyed in the following.

2.3.3.1. Tabu Search. Tabu search algorithm is one of the metaheuristic optimization algorithms. At first, this algorithm starts to move from an initial result, and then it selects the best neighbor result among the neighbors of current result. If this result was not in the Tabu list, the algorithm moves toward the neighbor result, otherwise the algorithm will check a criterion called aspiration criterion. Based on aspiration criterion, if the neighbor result is better than the available best result found by now, then the algorithm will move toward that result, even if that result is in the Tabu list. After the movement of algorithm toward neighbor result, the Tabu list is updated; it means that the previous movement by which we have moved toward the neighbor result is placed in the Tabu list in order to avoid return back to that result and cycle creation. The duration in which the movements are within the Tabu list is determined by a parameter called Tabu Tenure. Movement from the current result to the neighbor result continues until reaching the termination condition.

The Tabu search algorithm has been applied by Alvarez, Crespo, and Tamarit (2002) for the first time to assign students to the course groups in order to generate timetables with high quality and balance the number of students who registered in whole group and good results have been obtained. The allocation process in Alvarez et al. (2002) method has two phases: (1) the first phase, to generate a set of solutions for one student and (2) the second phase, a combination of a set of solutions and applying TS with local strategies to obtain the high quality timetables without considering the worst solution(s) for each student and (3) the third phase, to allocate room and improve the allocation, of course without changing the initial assignment of courses to the timeslots. In the second phase, TS is used to improve the quality of initial timetable, also some moves are also used which are: (1) simple move, (2) exchange and (3) multiple exchanges where the influence of these moves are highlighted in this method.

Aladag, Hocaoglu, and Basaran (2009) have presented the influence of neighboring structures on a TS algorithm to solve the UCTTP problem where the influence of simple and swap moves is tested on TS operations which are based on neighboring structures. Also, two new neighboring structures have been proposed by using simple and swap moves. Here, four applied neighboring structures and the comparison of obtained results from these structures has been specified. TS algorithm includes using advanced strategies and common components like Tabu list, various memories, neighboring structures and etc. One of the main factors is the influence of algorithm efficiency based on the defined neighboring structures which depends on the nature of problems. Here, two neighboring structures based on different types of moves like simple, swap and two other neighboring structures have been used which have been applied by Aladag and Hocaoglu (2007) and Alvarez et al. (2002), respectively. The aim is to combine the various effects in the applied TS algorithm. Due to the obtained results, multiple comparisons have been presented among whole neighboring structure statistically.

2.3.3.1.1. Simulated annealing. This method is a local search inspired of heating solids in physic science. This approach avoids trapping in the local optimal and uses local search methods more easily. The solutions obtained by local search replace the current solution frequently and this repeats until some of the termination criteria are satisfied. The starting process of the algorithm is through a creation of a random initial solution and in any iteration of SA algorithm, the current solution is replaced with a random solution which can be probabilistically the optimal solution of the problem. Search process starts from a high temperature; temperature decreases by progression formula as: $T_{i+1} = T_i \times \beta$. However, the cooling rate β and initial temperature value T are usually different and mostly determined based on experience and the nature of the problem (Obit, 2010).

To solve the UCTTP problem, the combination of Kempe neighboring chain in the simulated annealing algorithm has been presented by Tuga, Berretta, and Mendes (2007). In this method, the hard constraints are reformulated by relaxation; so these constraints are created in the form a relaxed soft constraint. However, the relaxation problem is analyzed in two steps: (1) to create a feasible solution based on a heuristic based graph and (2) a simulated annealing algorithm has been used to minimize the violations of soft constraints (in the second phase, a Kempe neighboring chain based heuristic has been used). However, the simulated annealing approach is presented by Aycan and Ayav (2008) to solve the UCTTP problem. They compared the efficiency of different neighboring search algorithms based on simple search, swap search, simple-swap search and computed the run time cost for each method. The highest satisfaction of timetabling is obtained by a combination of three mentioned algorithms.

2.3.3.2. Local search. Local search is introduced to find a solution in order to maximize a criterion among a number of ongoing solutions. These algorithms move from one solution to another one in a space of ongoing solutions (search space) by using limited changes, until a desirable solution is found or a period of time passes. The local search algorithm starts from an ongoing solution and then moves toward neighbor solutions frequently and this is possible only when the neighboring relations are defined in the adjacency of the search space of problem and selection of each solution to move is performed only by using information about neighbor solutions in a way that it maximizes the criterion locally; the given metaheuristic method is hill climbing algorithm. The termination of the local search algorithm could be based on a timeslot or when the best solution has not been optimized by the algorithm in the definite steps (Lewis, 2006; Obit, 2010).

Joudaki, Imani, and Mazhari (2010) have used local search algorithm and MA method to solve the UCTTP problem. These researchers presented a method based on an improved combination of SA and MA. In their research, the SA algorithm is applied as a local search routine which increases the ability of extracting from MA. Albeit, the modification of the crossover operator in MA and creation of initial population through a heuristic based method have been done in this algorithm. The optimization operator is performed by optimizing the generated chromosomes and minimizing the number of violations from the constraints and it is added as a new operator to MA. The success factors of MA are to use the heuristic capability of evolutionary algorithms and combine this advantage with the extraction ability of local search procedures.

Shengxiang and Jat (2011) have used local and guided search strategy within the GA process to solve the UCTTP problem. Here, the guided search strategy uses a data structure to create children, where this structure stores the extracted information from the good people of previous generations. Local search separation is an extractive technique which has the capability of improving the search efficiency of GAs. The results of consolidation of this local search in GA are satisfactory. The aim is to maximize the allocations and to minimize the violation of soft constraints. GSGA (Guided Search Genetic Algorithm) includes a guided search strategy and a local search technique. However, in GSGA a local search technique is applied to improve the quality of people by searching in three neighboring structures. Here, the aim is to study the efficiency of GSGA with LS strategies for the UCTTP problem and a uniform framework of combining standard GA and LS strategies has been used and using LS within GSGA results in the development of GSGA approaches to solve the UCTTP problem called EGSGA.

2.3.3.3. VNS and RIICN. The variable neighborhood search algorithm (VNS) has been presented by Abdullah, Burke, and McColloum (2005) to solve the UCTTP problem which proposed a basic VNS and then to use an exponential Monte Carlo acceptance criterion by each solution. The main idea is to apply a Monte Carlo acceptance criterion for improvement of the explorations through acceptance of the best solution with the given probability in order to find the number of promised neighbors. Again, Abdullah, Burke, & McColloum, 2007a, 2007b have presented a randomized iterative improvement with a composite neighboring algorithm (RIICN) to improve its previously presented algorithm which is in fact the combination of VNS and local search. The Tabu list has been applied to the penalty of inefficient and un-promised neighboring structures after a given number of iterations.

2.4. Intelligent novel methods

These methods include schemes like hybrid methods, fuzzy approach, hyper heuristic approaches, knowledge based methods, artificial intelligence and clustering Algorithms based approaches, and here we introduce two methods of them.

2.4.1. Hybrid approaches

This group of methods to solve the hybrid optimization problems has become more attractive in solving NP-complete problems recently and represents better efficiency and performance in solving this type of problems. In these methods, useful and efficient features of other techniques and algorithms are applied together to solve problems such as timetabling problem. Since the mentioned methods have some weaknesses, so combination of these methods can eliminate the weakness of individual methods and leads to a superior hybrid approach; therefore, this hybrid method can be led in generation of solutions more efficient in comparison to the results of each of these methods, individually. For example, a hybrid approach, in the first phase the possible initial solution is created by using the constraint programming technique and in the second phase the simulated annealing method is used to improve the initial generated solution and in the third phases the hill climbing solution is applied to improve and modify more.

A hybrid algorithm by Kostuch (2005) used sequential heuristic and simulated annealing to solve the UCTTP problem over ITC-2002 dataset has been presented. This method includes three phases as: phase one; using a sequential heuristic to generate the feasible initial timetables, phase two; applying simulated annealing to minimize the number of violations of soft constraints and phase three also uses simulated annealing algorithm to increase the improvement of generated timetables. A hybrid evolutionary approach has been presented to solve the UCTTP problem by Abdullah et al. (2007a, 2007b) where the combination of local search algorithm with evolutionary approaches give better results. The aim of problem is to completely meet of hard constraints and minimize the violations of soft constraints. The obtained results of a hybrid evolutionary approach show the minimization of penalty values from soft constraints.

However, Abdullah and Hamdan (2008) also presented another hybrid approach to solve UCTTP problem implemented in three phases. The first phase is to generate the initial solution by using constructive heuristics, the second phase uses an improved technique applying the randomized iterative improvement algorithm and the third phase also uses simulated annealing as an acceptance criterion. However, the hill climbing has been used in the third phase to promote the quality of timetables. Integration of two Tabu search and memetic algorithms to solve UCTTP problem has been done by Turabieh and Abdullah (2009) where the crossover and mutation operators have been used to select a solution from a population and then the random neighboring structures have been reused for each solution whose quality of solutions have not been promoted within Tabu list. The Tabu list is applied to penalty neighboring structures which do not have the capability to generate better solutions. Recently, a multi population hybrid genetic algorithm has been proposed by Shahvali Kohshori and Saniee Abadeh (2012) to solve UCTTP problem based on three genetic algorithms, FGARI, FGASA and FGATS. In this algorithm, fuzzy logic is used to evaluate the number of violations of soft constraints in the fitness function to deal with real world data which are ambiguous and non-deterministic and random, local search, simulated annealing and Tabu search methods would also be useful accompanied with fuzzy method to improve inductive search to satisfy the search ability.

2.4.2. Fuzzy approach

Aristotle logic forms the basis of classic mathematics and based on this logic, all things are included in a fixed rule where due to this fact either that thing exists or not. The lack of decisiveness in objects, things and etc. represent the lack of accuracy, ambiguous and generally fuzziness. Now in fuzzy science the intermediary which does not exist within mathematic is released. In fact, fuzzy logic is a continuous logic inspired of the approximate argumentation of human being (Asmuni, 2008).

To optimal fuzzy classification of students, Amintoosi and Haddadnia (2005) have used a fuzzy function to solve UCTTP genetic programming problem. The aim was to separate the students of populous classes. This separation has led to reduce the amount of conflict of students' courses in weekly program. Here, at first the fuzzy c-mean clustering algorithm divides students into c classes and then, according to the criteria of distance of clusters' centers, density of each cluster, co-entrance of students of each cluster and dimension ratio of clusters by using a fuzzy function, the value of clustering is determined so that by selecting the appropriate features (courses), the best classification of students is obtained. A hybrid fuzzy evolutionary algorithm has been presented by Rachmawati and Srinivasan (2005) to multi objective resource allocation problem which was a student's project allocation problem. Here, the student project allocation must satisfy a number of soft purposes in a sequence of some points. This algorithm uses a fuzzy inductive system to model and collect purposes. Fuzzy system considers some priorities to decide on an agreement among the different purposes by which the direction of the search path toward attractive regions within purpose space is performed. To solve UCTTP problem, Asmuni, Burke, and Garibaldi (2005) have presented a fuzzy multiple heuristic sorting method where the sorting of events has been done through a simultaneous considering of three distinct heuristics by using fuzzy methods. The sequential combination of three heuristics is sorted as (1) the highest degree, (2) saturation degree and (3) submission degree and fuzzy weight of an event is also used to represent that event has what problem to be scheduled. The descending sorted events are allocated to the last timeslot with the least penalty cost sequentially while the feasibility is maintained during the whole process. A fuzzy solution based on memetic approach has been presented by Golabpour, Mozdorani Shirazi, Farahi, kootiani, and beige (2008) to solve university timetabling problem where a timetable has been compared with both genetic and memetic algorithms and its results may satisfy the existing constraints simultaneously during a shorter time interval. The aim is to use fuzzy logic as a means for local search in memetic algorithm. Chaudhuri and Kajal (2010) have presented a fuzzy genetic heuristic idea to solve UCTTP problem where the genetic algorithm has been applied by using an indirect representation based on integration events, features and the fuzzy set model is also to evaluate the violation of soft constraints in objective function according to uncertainties of real world data. Here, a degree of uncertainty within objective function is considered for each soft constraint and this uncertainty is evaluated by formulating violation parameter from soft constraint in objective function using fuzzy membership functions. However, a fuzzy genetic algorithm has been presented by Shahvali Kohshori, Saniee Abadeh, and Sajedi (2011) accompanied with local search to solve UCTTP problem where the fuzzy genetic algorithm with a local search algorithm uses inductive search to solve the combined problem and applied local search which has the ability of improving efficiency within genetic algorithm. The applied fuzzy logic within this approach is also used to evaluate the violation of soft constraints in objective function due to facing with uncertainty in real world data.

However, recently Shatnawi, Al-Rababah, and Bani-Ismail (2010) have used a novel clustering technique based on FP-Tree to solve UCTTP where the given technique is done to classify students based on their selective courses who submitted for the next semester. The aim of this clustering is to solve scheduling of courses where in the previous semesters the submission of students in some courses due to simultaneous scheduling has been prevented, while in this technique no conflict would happen over scheduling of exams since no two exams at the same time would be taken for courses by two identical groups of students.

3. Studying the approach based on distributed multi agent systems in UCTTP

In Srinivasan et al. (2011), an agent could observe and receive anything through sensors from its environment and then performs over environment through a driver. Agents are classified into different classes based on their application, including (1) autonomous, (2) intelligent, (3) reactionable, (4) proactive, (5) learner, (6) mobile, (7) cooperative/communicative agents.

3.1. Multi agent systems

Therefore, agents must have a common language and a communication media to cooperate with each other where these two components are essential among two agents. Multi agent systems have a more general concept and for all types of current systems, including multiple autonomous components are applied to the following features and include: (1) each agent has the ability of solving a problem incompletely, (2) in multi agent systems there is no general control system, (3) data are as distributed and (4) computations are asynchronous. However, after stating the autonomous features, we can explain the multi-capacity features in agents as: (1) dividing tasks among a large number of agents which are modular making, flexibility, modifiable and extensible, (2) the knowledge released over different resources (agents) has the capability of being integrated to completion, (3) applications require distributed computations by distributed multi agent systems for better support and (4) the agents technology provides the summary and result of distributed components technology (Srinivasan et al., 2011; Wangmaeteekul, 2011).

3.2. Studying the related works to solve UCTTP by distributed multi agent systems based approach

To generate course timetables, Gaspero, Missaro, and Schaerf (2004) have used distributed multi agent architecture. Here, UCTTP problem includes a set of courses in fixed timeslots in a circulating week. UCTTP problem refers to only a set of university departments. Each department has an education program corresponding to particular rules, constraints and purposes based on their resources and resources do not have shared feature, unless the resource exchange among departments would be useful which is done through negotiation. To solve the problem, a multi agent scheduling system based on one market with artificial money has been considered and each department includes three colleague agents: (1) to search a local solution, (2) to negotiate for resources with other departments and (3) to manage the related information. In Kaplansky and Meisels (2004), a distributed timetabling system is in three software layers as: (1) the first layer, scheduling agents, (2) the second layer, using a negotiation protocol to generate a pervasive university scheduling and (3) the third layer, presenting network infrastructures. Protocol implementation is to negotiate among scheduling agents and classroom agent. The main attempt is focused on studying the various places in the protocol of negotiation among scheduling agents.

The studying of distributed timetabling problem based on scheduling agents by Meisels and Kaplansky (2003), almost in real timetabling problems includes organized parts which requires creating timetables for people included in an independent way, while some global constraints are considered. Recently, department timetabling is combined with each other as a result of integration and consistent solution and this combination itself would require negotiation of various agents. Here, only one model including a reduplicated agent called CA (central agent) is studied and the duty of this agent is to cooperate of search process among all SA(s) (scheduling agents). However, this idea is in respect of creating feasible solutions for a network of SA(s). Presenting a multi agent system to create automatic timetabling with shared resources has been done by Pedroso (2003) where an automatic timetabling system has been proposed for a normal state of the universities. Agents (departments) compete for a set of classrooms over a number of given timeslots. Each agent applies its own algorithm and this algorithm may be unknown for other departments. A central system is assumed to determine whether some agents are allowed to allocate resources or not, which is also based on a list of requests received from each agent. The initial priority is evaluated by a number of attendances (expectations) and some of the requirements are valued for particular features over resources. Presenting a distributed technique for UETTP has been done by Kaplansky, Kendall, Meisels, and Hussin (2004) based on a case study on a real university where the timetabling of distributed exams is performed by beginning of a distributed search for a global solution through computation of agents and this is as the best local scheduling where after that the negotiation among agents is done to find a global solution. In the presented method, there are two solutions to model multi agent systems in timetabling of exams where (1) applying SA(s) system (scheduling agents) and (2) applying Tabu search would be as a hyper heuristic, where the focus is on negotiation of procedures which would be able to detect and avoid collisions among SA(s) with common exams. The negotiation protocol consists of three steps where the first phase is on search for a local solution in timetabling problem at each agent, the second phase is a global timetabling without collision and the third phase is also the negotiation of SA(s) with each other to improve the local solution a creating timetabling.

To solve UCTTP problem, a global solution model based on agent has been presented by Yanga, Paranjape, and Benedicenti (2006) where in this model each agent within the system is a constraint in UCTTP problem. The agent based solution includes generality, flexibility, dynamism and scalability which act better than other modes. Each constraint agent is an independent module in the proposed model and new constraint agents have the ability to join and leave simply the timetabling system which would be the dynamism result of constraint in resolving through system technique and design. To evaluate a system with mobile agents in UCTTP problem one test has been presented by Yanga, Paranjape, and Benedicenti (2004) which have been done as a mechanism to solve UCTTP problem. Most focus was on the application of new technology of mobile agents to implement the solution of a general UCTTP problem. In modeling an agent, each mobile agent represents a course called course agent (CA) and course agents perform negotiation with other agents with a mechanism defined as a Signboard agent. A Signboard agent is considered for per day of the week and each platform represents the usual days of the week. The mobile agents' technology has been established based on TEEMA (TR Labs Execution Environment for Mobile) platform. The study of UCTTP problem as distributed DisTTP (Distributed TimeTabling Problem) has been done by Xiang and Zhang (2008) where previously this problem has been introduced as a distributed constraint satisfaction problem (DisCSP). To solve DisTTP problem based on multiple segmented constraint networks

(MSCNs), an alternative method has been presented here. The network topology is based on sparse matrix and unlike DisTTP algorithm, a central agent is required proposed to the presented solution. The proposed method maintains a part of timetabling from all particular agents. The distributed timetabling avoids translation and interpretation of concentrated local constraints and their communication where this process keeps the shorter scheduling and privatization of departments. The obtained solution of a multi agent system consists of an SA (Scheduling Agent) at each department and a CA (Central Agent). SA(s) has been proposed for department timetabling to meet local constraints and CA collects department timetables and performs studies in respect of global constraints to modify and revisit department timetables. The implementation of class timetabling has also been done by Nandhini and Kanmani (2009) based on multi agent systems where the implementation process has been presented through applying hill climbing algorithm with the sharpest upward slope (until reaching the ancestor). CombinationGenerator and MinFinders agents are applied to generate maximum input combinations and create a combination with the minimum evaluation function to consecutive exams, respectively. Using this proposed method would continue the initial random solution until reaching the given optimal solution.

A multi agent system has been proposed by Oprea (2007) for UCTTP problem scheduling where two basic features have been defined by the distribution and the dynamism of environment. An efficient solution to solve this problem could be provided by an agent based approach. The focus is on architecture of multi agent systems which is presented for UCTTP timetabling called MAS-UP-UCT. The advantage of this method includes a large number of communications, collaboration and negotiation among agents.

Presenting a system model to UCTTP problem has been proposed by Yanga, Paranjape, Benedicenti, and Reedc (2006) using mobile agents where a multi agent system has been applied to generate the solutions of UCTTP problem. Four types of agents collaborate sequentially with each other to perform courses scheduling process as: (1) course (mobile), (2) signboard, (3) publisher and (4) mediator. The powerful key of this approach is to use the independency feature of agents and this independency is embedded clearly in the performance of course agent. Each course agent in the system is responsible to negotiate with other course's agents to find the satisfactory resource class (timeslots and rooms) in order to present and represent courses. Solving UCTTP problems by using multi agent systems requires the development of an intelligent decision making system proposed by Yanga and Paranjape (2011) UCTTP problem is a dynamic distribution problem which requires a system of decision making, where agent are independent and a flexible communication methodology has been used to create the backbone of decision making system. The course agents represent each course in the problem and the negotiation of course agents with each other has also been applied by a signboard agent to find the collision prevented acceptable timetabling. The signboard agent uses a mechanism to identify course agents where each requires a negotiation with other course agents and in other words it performs resolving collisions. However, that mechanism is also built through completion of each timetable available for user. Here also the powerful feature of autonomous has been used for an agent to present all aspects of basic units in problem (course). Strnad and Guid (2007) have presented a new architecture for multi agent systems in solving UCTTP problem where agents allocate the required technical and human resources through negotiation as agencies from each course. The negotiation and description protocol of independency decisions of agents has been defined in the given framework format. The advantages of the multi agent approach are to resolve collisions directly, variability of strategic negotiation and present some real problems and events.

3.3. Summary of conducted research in this paper on Educational Timetabling Problems

A list of information which gives details about solutions of educational timetabling problems and techniques which have been deployed in each solution is presented in Table 1.

4. Experiments and results

The Existence of different types of approaches and algorithms to solve UCTTP problem has resulted in the vast domain of papers in this field where selecting the best approach or following the sequences of that approach could provide relatively better successes for future approaches. In this respect, we can firstly find a valid dataset over which most algorithms be run on studying experiments and then, experiences and analyze them over several datasets. Therefore, at first we have studied the domain of other's experiments with valid datasets and then evaluated the comparison of these experiments and their results over these datasets. In the coming sections, Socha dataset, Ben Paechter dataset, ITC-2002 and ITC-2007 datasets are presented. In these sections A_i ($i = 1, \ldots ,66$) are algorithms which have been used in mentioned datasets and presented in detail in Appendix 2.

4.1. The study and comparison of algorithms on Socha et al. (2002) dataset

The dataset of Table 2 consists of four columns where the first column is a list of a department's events and resource classification and three other columns represent the number of events and resources in terms of their small, medium and large sizes. The row number of this dataset corresponds with each available event and resource of department with ascending order from small to medium and large for each row (Obit, 2010; Socha et al., 2002).

Detailed description and classification of used dataset in Table 2 namely small, medium and large datasets are presented in Table 3.

We can represent algorithms run based on that dataset in terms of values of fitness function of each algorithm in Table 4 (in appendix) over a given dataset in Tables 2 and 3. In this Table, the amount of violation from soft constraints has been represented for each small (1–5), medium (1–5) and large datasets per applied algorithm. According to Table 4, the best performance of approaches is presented with numerical value of 0–529. In general, A_4, A_5, A_7, A_{11} and A_{13} encounter inefficiency for small datasets and in turn other algorithms also have descending performance over small size datasets from left to right as A_3, A_{14}, A_6, A_9, A_{11} and A_{10}. A_2–A_{14}. Algorithms have appropriate efficiency over medium and large size problem datasets.

4.2. The study and comparison of algorithms on Ben Paechter Dataset

The dataset of Table 5 consists of four columns where the first column is a list of department's events and resources classification and three other columns represent the number of events and resources in terms of their small, medium and large sizes. The row number of this dataset corresponds with each available event and resource of the department with ascending order from small to medium and large for each row. The details of Table 5 are as Table 3 in Section 4.1 (Rossi et al., 2003).

In the Table 5 at first the performance of algorithms A_{30}, A_{57}, A_{64}, A_{65} and A_{66} are evaluated on small (1–5) dataset and according to Tables 3 and 5 where the ranking of algorithms would have bet-

Table 1
Summary of conducted research on Educational Timetabling Problems.

Approaches	Techniques		Article Name
Operational Research (OR)	Graph Coloring (GC)		Graph Coloring for Class Scheduling (Dandashi & Al-Mouhamed, 2010), An Introduction to TimeTabling (De Werra, 1985), Bipartite Graph Edge Coloring Approach to Course Timetabling (Hafizah & Zaidah, 2010), A Study of University Timetabling that Blends Graph Coloring with the Satisfaction of Various Essential and Preferential Conditions (Redl, 2004), Split Vertices in Vertex Coloring and Their Application in Developing a Solution to the Faculty TimeTable Problem (Selim, 1988), An upper bound for the chromatic number of a graph and its application to timetabling problems (Welsh & Powell, 1967)
	Integer/Linear Programming (IP/LP)		A 0-1 integer programming approach to a university timetabling problem (Bakir & Aksop, 2008), Efficient solutions for a university timetabling problem through integer programming (Daskalaki & Birbas, 2005), An integer programming formulation for a case study in university timetabling (Daskalaki, Birbas and Housos, 2004)
	Constraint Satisfaction Programming (CSP)		Timetable planning using the constraint-based reasoning (Deris et al., 2000), Incorporating Constraint Propagation in Genetic Algorithm for University Timetable Planning (Deris et al., 1999), Constructing university TimeTable using constraint satisfaction programming approach (Zhang & Lau, 2005).
Metaheuristic and Swarm Intelligent	Multi-population	Genetic Algorithm (GA)	A Novel Genetic Algorithm Technique for Solving University Course Timetabling Problems (Alsmadi et al., 2011), Trans Genetic Coloring Approach for Timetabling Problem (Asham et al., 2011), On Improvement of Effectiveness in Automatic University Timetabling Arrangement with Applied Genetic Algorithm (Khonggamnerd & Innet, 2009)
		Ant Colony Optimization (ACO)	Hybrid Ant Colony Systems For Course Timetabling Problems (Ayob and Jaradat, 2009), Solving the Post Enrolment Course Timetabling Problem by Ant Colony Optimization (Mayer et al., 2008), A Max–Min Ant System for the University Course Timetabling Problem (Socha et al., 2002)
		Meme tic Algorithm (MA)	Using improved Memetic algorithm and local search to solve University Course Timetabling Problem (Joudaki et al., 2010), A Memetic Algorithm for the University Course Timetabling Problem (Jat & Shengxiang, 2008)
		Harmony Search Algorithm (HSA)	University Course Timetabling Using a Hybrid Harmony Search Metaheuristic Algorithm (Al-Betar, Tajudin Khader and Zaman, 2012)
	Single-population	Local Search (LS)	A Fuzzy Genetic Algorithm with Local Search for University Course Timetabling (Shahvali Kohshori et al., 2011), Genetic Algorithms with Guided and Local Search Strategies for University Course Timetabling (Shengxiang & Jat, 2011)
		Variable Neighborhood Search (VNS)	An Investigation of Variable Neighborhood Search for University Course Timetabling (Abdullah et al., 2005)
		Randomized Iterative Improvement with Composite Neighborhood (RIICN)	Using a Randomized Iterative Improvement Algorithm with Composite Neighborhood Structures for University Course Timetabling (Abdullah et al., 2007a, 2007b)
		Simulated Annealing (SA)	Solving the Course Scheduling Problem Using Simulated Annealing (Aycan & Ayav, 2008), A Hybrid Simulated Annealing with Kempe Chain Neighborhood for the University Timetabling Problem (Tuga et al., 2007)
		Tabu Search (TS)	The effect of neighborhood structures on tabu search algorithm in solving course timetabling problem (Aladag et al., 2009), The effect of neighborhood structure and of move types in the problem of course timetabling with the tabu search algorithm (Aladag & Hocaoglu, 2007), Design and Implementation of a Course Scheduling System Using Tabu Search (Alvarez et al., 2002)
Novel Intelligent	Hybrid Algorithms (Hybrid Meta heuristic)		A Hybrid Evolutionary Approach to the University Course Timetabling Problem (Abdullah et al., 2007a, 2007b), A Hybrid Approach for University Course Timetabling (Abdullah & Hamdan, 2008), The University Course Timetabling Problem with a Three-Phase Approach (Kostuch, 2005), A Hybrid Fuzzy Evolutionary Algorithm for A Multi-Objective Resource Allocation Problem (Rachmawati & Srinivasan, 2005), Hybrid Genetic Algorithms for University Course Timetabling (Shahvali Kohshori & Saniee Abadeh, 2012), Incorporating Tabu Search into Memetic Approach for Enrolment-based Course Timetabling Problems (Turabieh & Abdullah, 2009)
	Fuzzy method		Fuzzy multiple heuristic ordering for course timetabling (Asmuni et al., 2005), Fuzzy Genetic Heuristic for University Course TIMETABLE Problem (Chaudhuri & Kajal, 2010), A fuzzy solution based on Memetic algorithms for timetabling (Golabpour et al., 2008)
	Clustering Algorithms		Fuzzy C-means Clustering Algorithm to Group Students in A Course into Smaller Sections (Amintoosi & Haddadnia, 2005), Applying a Novel Clustering Technique Based on FP- Tree to University Timetabling Problem: A Case Study (Shatnawi et al., 2010)
Multi-Agent Systems	Review papers Multi-Agent Systems		A Multi-agent Architecture for Distributed Course Timetabling (Gaspero et al., 2004), Distributed Examination Timetabling (Kaplansky et al., 2004), Negotiation among Scheduling Agents for Distributed TimeTabling (Kaplansky & Meisels, 2004), Scheduling Agents-Distributed TimeTabling Problems (Meisels & Kaplansky, 2003), Implementation of Class Timetabling Using Multi Agents (Nandhini & Kanmani, 2009), Designing a Multi-Agent Approach System for Distributed Course TimeTabling (Obit et al., 2011), A Multi-Agent System for University Course TimeTable Scheduling (Oprea, 2007), A Multi-Agent System for Automated TimeTabling with Shared Resources (Pedroso, 2003), A Multi-Agent System for University Course TimeTabling (Strnad & Guid, 2007), Distributed University Timetabling with Multiply Sectioned Constraint Networks (Xiang & Zhang, 2008), A multi-agent system for course timetabling, Intelligent Decision Technologies (Yanga & Paranjape, 2011), An Agent Based General Solution Model For the Course TimeTabling Problem (Yanga et al., 2006), An Examination of Mobile Agents System Evolution in the Course Scheduling Problem (Yanga et al., 2004), A System Model for University Course TimeTabling Using Mobile Agents (Yanga et al., 2006), Using Distributed Agents to Create University Course TimeTables Addressing Essential Desirable Constraints and Fair Allocation of Resources (Wangmaeteekul, 2011)

Table 1 (*continued*)

Approaches	Techniques	Article Name
Literatures review papers	Review papers	Fuzzy Methodologies for Automated University Timetabling Solution Construction and Evaluation (Asmuni, 2008), A large scale timetabling problem (Aubin & Ferland, 1989), A New Heuristic Optimization Algorithm: Harmony Search (Geem, Kim, & Loganthan, 2001), A Survey of Approaches for University Course TimeTabling Problem (Feizi-Derakhshi, Babaei and Heidarzadeh, 2012), The Construction of Class-Teacher TimeTables (Gotlib, 1963), Metaheuristics for University Course Timetabling (Lewis, 2006), Developing Novel Meta-heuristic, Hyper-heuristic and Cooperative Search for Course Timetabling Problems (Obit, 2010), A Comparison of the Performance of Different Metaheuristics on the Timetabling Problem (Rossi, Sampels, Birattari, Chiarandini, Dorigo, Gambardella, Knowles, Manfrin, Mastrolilli, Paechter, Paquete, and Stutzle, 2003), Multi-Agent based Decision Support System Using Data Mining and Case Based Reasoning (Srinivasan, Singh and Kumar, 2011)

Table 2
Socha et al. (2002) dataset.

Features	Categories		
	Medium	Small	Large
Number of courses	400	100	400
Number of rooms	10	5	10
Number of features	5	5	10
Number of students	200	80	400
Max. courses per student	20	20	20
Max. students per course	50	20	100
Approx. features per room	3	3	5
Percent feature use	80	70	90

Table 3
Different sizes of datasets along with number of events and resources.

Categories	Features			
	#Students	#Rooms	#Events	#Features
Small 1	80	5	100	5
Small 2	80	5	100	5
Small 3	80	5	100	5
Small 4	80	5	100	5
Small 5	80	5	100	5
Medium 1	200	10	400	5
Medium 2	200	10	400	5
Medium 3	200	10	400	5
Medium 4	200	10	400	5
Medium 5	200	10	400	5
Large	400	10	400	10

Table 5
Ben Paechter dataset.

Features	Categories		
	Medium	Small	Large
Number of events	400	100	400
Number of rooms	10	5	10
Number of room features	5	5	10
Approx. features per room	3	3	5
Percent feature use	80	70	90
Number of students	200	80	400
Max. events per student	20	20	20
Max. students per course	50	20	100

ter performance in terms of efficiency over this dataset so that here the efficiency of the algorithm A_{57} is the best and A_{64} and A_{30} algorithms have well performance. However, the worst performance belongs to both A_{66} and A_{65} algorithms. We could obtain the performance of the above algorithms on Medium (1–5) size dataset where the ranking process of algorithms would have better performance in terms of efficiency over this data A_{57} so that here the efficiency of algorithm A_{64} is the best and A_{57} and A_{65} algorithms have good performance. However, the worst performance belongs to both A_{66} and A_{30} algorithms. Finally, we could obtain the performance of mentioned algorithms on large and very large dataset where the ranking process of algorithms would have better performance in terms of efficiency over these two datasets so that in the

Table 4
The performance of algorithms to solve UCTTP problem over dataset, Socha et al. (2002).

Algorithms	Categories										
	Small1	Small2	Small3	Small4	Small5	Medium1	Medium2	Medium3	Medium4	Medium5	Large
A_1[a]	1	3	1	1	0	195	184	248	164.5	219.5	851.5
A_2	1	2	0	1	0	146	173	267	169	303	1166
A_3	10	9	7	17	7	243	325	249	285	132	1138
A_4	0	0	0	0	0	317	313	357	247	292	932
A_5	0	0	0	0	0	242	161	265	181	151	757
A_6	6	7	3	3	4	372	419	359	348	171	1068
A_7	0	0	0	0	0	221	147	246	165	130	529
A_8	17	15	24	21	5	201	190	229	154	222	1066
A_9	3	4	6	6	0	140	130	189	112	141	876
A_{10}	1	3	1	1	0	195	184	248	164.5	219.5	851.5
A_{11}	2	4	2	0	4	254	258	251	321	276	1027
A_{12}	0	0	0	0	0	80	105	139	88	88	730
A_{13}	0	0	0	0	0	175	197	216	149	190	912
A_{14}	5	5	3	3	0	176	154	191	148	166	798
A_{15}	0	0	0	0	0	77	73	133	69	101	627
A_{16}	0	0	0	0	0	50	70	102	32	61	653
A_{17}	0	0	0	0	0	93	98	149	103	98	980
A_{18}	0	0	0	0	0	124	117	190	132	73	424
A_{19}	1	2	0	1	0	146	173	248	164.5	171	851.5
A_{20}	0	0	0	0	0	99	102	158	86	79	768
A_{21}	0	0	0	0	0	240	160	242	158	124	801
A_{22}	0	0	0	0	0	70	77	115	67	64	555
A_{23}	1	1	1	1	0	136	138	165	143	135	786
A_{24}	0	0	0	0	0	117	108	135	75	160	589

[a] Full name of algorithms are presented in Appendix 2.

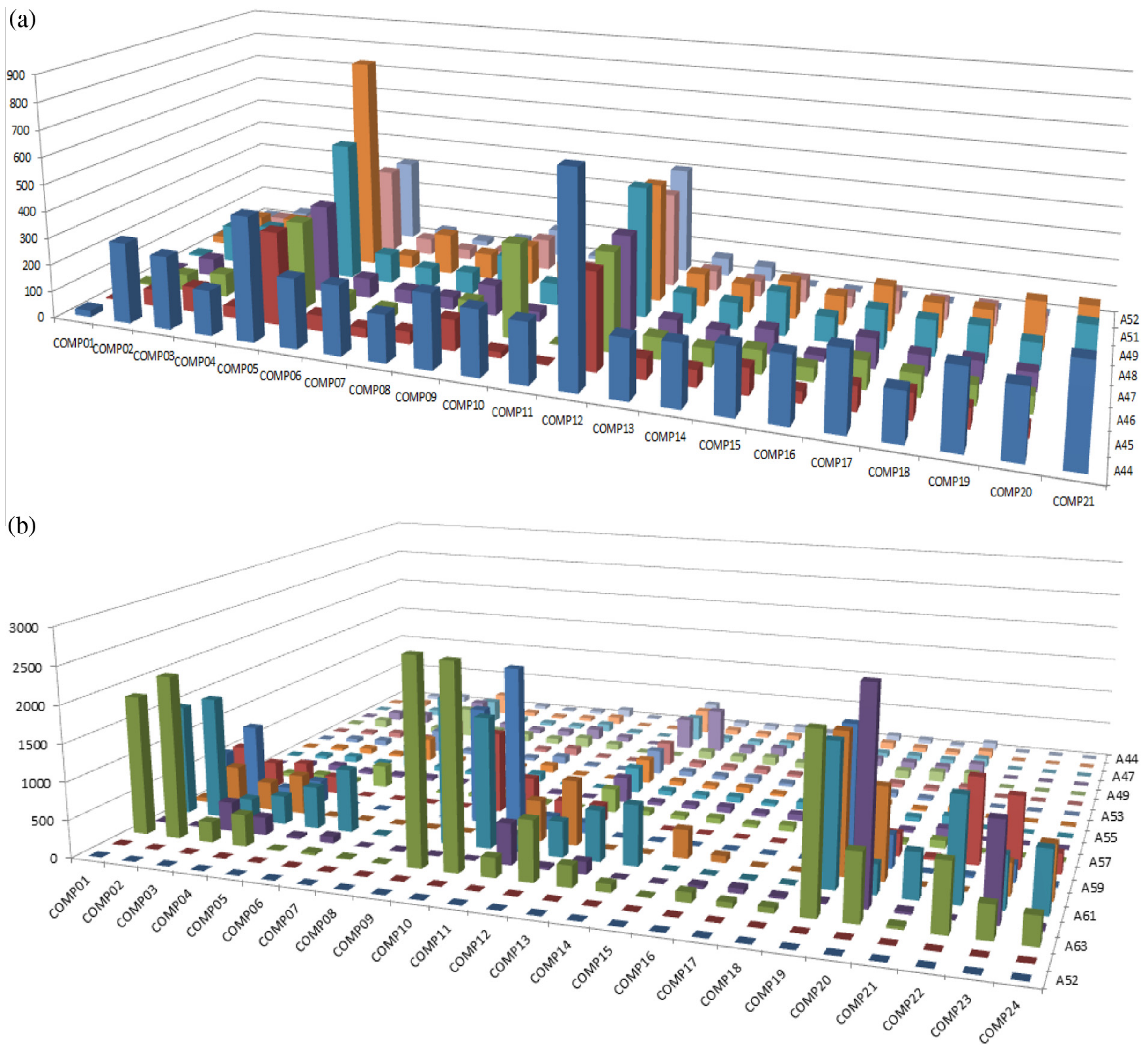

Fig. 3. Comparison of studied algorithms performances'.

large dataset, the efficiency of algorithm A_{64} is the best and A_{57} and A_{30} algorithms have well performance. However, the worst performance belongs to both A_{66} and A_{64} algorithms and the ranking process of algorithms would have better performance in terms of efficiency over very large datasets so that the efficiency of algorithm A_{57} is the best and A_{65} and A_{30} algorithms have well performance. However, the worst performance belongs to both A_{66} and A_{64} algorithms. In general, we find that the performance of the algorithm A_{30} in all datasets is better than other algorithms, while in the turn GA algorithm also has the worst performance on the same dataset. In Table 6 (in appendix) we could represent the performance of some other algorithms on Ben Paechter dataset to solve UCTTP problem. In this Table, the terms of Best, MED (Median) and In% show the best result in a number of runs, the medium state and the running percent's to reach the failed possible solutions, respectively.

4.3. The study and comparison of algorithms on ITC-2002 dataset

The dataset of Table 7 (in appendix) consists of seven columns where the first column is a list of a department's events and resource classification and the six other columns represent the number of events and resources and also the proportion of allocating rooms to events, events to students and students to events (Obit, 2010; Paechter, Gambardella, & Rossi-Doria, 2002). However, after reviewing Table 7 (in appendix) to introduce ITC-2002 dataset, it is the turn of studying algorithms which were run on this dataset. In Table 8 (in appendix) is also the performance of algorithms evaluated on datasets. In the Tables of 7 and 8 COMP01 to COMP20 show the competition between groups of solutions.

4.4. The Study and Comparison of Algorithms on ITC-2007 and ITC-2007 CBCT Datasets

The datasets in both Tables 9 and 10 (in appendix) include the entire timetabling ITC-2007 dataset as (1) general and (2) based on CBCT[1] problem. At first, Table 9 (in appendix) consists of nine columns where the first column is a list of a department's events and resources classification and the remained columns represent the number of events and resources according to the proportion of placing rooms, features of each room, events, the number of students. However, Table 10 (in appendix) represents the full feature to solve UCTTP problem based on each course and according to an

[1] Curriculum-Based Course TimeTabling

Table 6
The performance of algorithms on Ben Paechter dataset.

Algorithms	Categories																				
	Small1		Small2		Small3		Small4		Small5		Medium1		Medium2		Medium3		Medium4		Medium5		Large	
Best/Med	B	M	B	M	B	M	B	M	B	M	B	M	B	M	B	M	B	M	B	M	B	M
A_5	0	0	0	0	0	0	0	0	0	0	242	245	161	262.8	265	267.8	181	183.6	151	152.6	100%In	-
A_6	6	–	7	–	3	–	3	–	372	–	419	–	359	–	348	–	171	–	1068	–	–	–
A_7	0	–	0	–	0	–	0	–	0	–	221	–	147	–	246	–	165	–	135	–	529	–
A_9	3	–	4	–	6	–	6	–	0	–	140	–	130	–	189	–	112	–	141	–	876	–
A_{25}	0	0	0	0	0	0	0	0	0	0	227	229.5	180	185	235	238.5	142	155	200	203	–	–
A_{26}	0	–	0	–	0	–	0	–	0	–	317	–	313	–	357	–	247	–	292	–	–	–
A_{27}	1	–	2	–	0	–	1	–	0	–	146	–	173	–	267	–	169	–	303	–	80%In	–
A_{28}	–	8	–	11	–	8	–	7	–	5	–	199	–	202.5	–	77.5%In	–	177.5	–	100%In	–	100%In
A_{29}	0	–	3	–	0	–	0	–	0	–	280	–	188	–	249	–	247	–	232	–	–	–
A_{30}	–	1	–	3	–	1	–	1	–	0	–	195	–	184	–	248	–	164.5	–	219.5	–	851.5
A_{31}	10	–	9	–	7	–	17	–	7	–	243	–	325	–	249	–	285	–	112	–	1138	–
A_{32}	0	0	0	0	0	0	0	0	0	0	139	143	92	69.5	122	124	98	101	116	119.5	615	622.5
A_{33}	2	–	4	–	2	–	0	–	4	–	254	–	258	–	251	–	321	–	276	–	1027	–
A_{34}	0	–	3	–	0	–	0	–	0	–	280	–	188	–	249	–	247	–	232	–	100%In	–

Table 7
Complete specification of ITC-2002 dataset.

Competitions	Categories					
	#Events	#students	#Rooms	Rooms/event	Events/Student	Students/event
COMP01	400	200	10	1.96	17.75	8.88
COMP02	400	200	10	1.92	17.23	8.62
COMP03	400	200	10	3.42	17.70	8.85
COMP04	400	300	10	2.45	17.43	13.07
COMP05	350	300	10	1.78	17.78	15.24
COMP06	350	300	10	3.59	17.77	15.23
COMP07	350	350	10	2.87	17.48	17.48
COMP08	400	250	10	2.93	17.58	10.99
COMP09	400	220	11	2.58	17.36	8.68
COMP10	400	200	10	3.49	17.78	8.89
COMP11	400	220	10	2.06	17.41	9.58
COMP12	400	200	10	1.96	17.57	8.79
COMP13	400	250	10	2.43	17.69	11.05
COMP14	350	350	10	3.08	17.12	17.12
COMP15	350	300	10	2.19	17.58	15.07
COMP16	440	220	11	3.17	17.75	8.88
COMP17	350	300	10	1.11	17.67	15.45
COMP18	400	200	10	1.75	17.56	8.78
COMP19	400	300	10	1.94	17.71	13.28
COMP20	350	300	10	3.43	17.49	14.99

Table 8
The performance of algorithms on ITC-2002 dataset.

Competitions	Categories											
	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8	A_9	A_{10}	A_{11}	A_{12}
COMP01	54	45	61	85	63	132	148	61	178	211	257	153
COMP02	25	25	39	42	46	92	101	39	103	128	112	118
COMP03	44	65	77	84	96	170	162	77	156	213	226	120
COMP04	132	115	160	119	166	265	350	160	399	408	441	385
COMP05	97	102	161	77	203	257	412	161	336	312	299	398
COMP06	3	13	42	6	92	133	246	42	246	169	209	129
COMP07	12	44	52	12	118	177	228	52	225	281	99	99
COMP08	23	29	84	32	66	134	125	54	210	214	194	111
COMP09	21	17	50	184	51	139	126	50	154	164	175	119
COMP10	53	61	72	90	81	148	147	72	153	222	308	153
COMP11	46	44	53	73	65	135	144	53	169	196	273	149
COMP12	96	107	110	79	119	290	182	110	219	282	242	229
COMP13	69	78	109	91	160	251	192	109	248	315	364	240
COMP14	13	52	93	36	197	230	316	93	267	345	156	282
COMP15	35	24	62	27	114	140	209	62	235	185	95	172
COMP16	12	22	34	300	38	114	121	34	132	185	171	91
COMP17	104	86	114	79	212	186	327	114	313	409	148	356
COMP18	39	31	38	39	40	87	98	38	107	153	117	190
COMP19	63	44	128	86	185	256	325	128	309	334	414	228
COMP20	2	7	26	0	17	94	185	26	185	149	113	72

Table 9
Complete specification of ITC-2007 dataset.

Categories	Features							
	Number of events	Number of rooms	Number of features	Number of students	Max. students per event	Max. events per students	Mean features per room	Mean features per event
1	400	10	10	500	33	25	3	1
2	400	10	10	500	32	24	4	2
3	200	20	10	1000	98	15	3	2
4	200	20	10	1000	82	15	3	2
5	400	20	20	300	19	23	2	1
6	400	20	20	300	20	24	3	2
7	200	20	20	500	43	15	5	3
8	200	20	20	500	39	15	4	3
9	400	10	20	500	34	24	3	1
10	400	10	20	500	32	23	3	2
11	200	10	10	1000	88	15	3	1
12	200	10	10	1000	81	15	4	23
13	400	20	10	300	20	24	2	1
14	400	20	10	300	20	24	3	1
15	200	10	20	500	41	15	2	3
16	200	10	20	500	40	15	5	3
17	100	10	10	500	195	23	4	2
18	200	10	10	500	65	23	4	2
19	300	10	10	1000	55	14	3	1
20	400	10	10	1000	40	15	3	1
21	500	20	20	300	16	23	3	1
22	600	20	20	500	22	25	3	2
23	400	20	30	1000	69	24	5	3
24	400	20	30	1000	41	15	5	3

Table 10
Complete specification of ITC – 2007 CBCT dataset.

Categories	Features						
	#Events	#Periods	#Rooms	#Courses	#Course of groups	Lower Bound Best	Upper Bound Best
COMP01	160	30	6	30	14	5	5
COMP02	283	25	16	82	70	24	10
COMP03	251	25	16	72	68	66	38
COMP04	286	25	18	79	57	35	35
COMP05	152	86	9	54	139	291	114
COMP06	361	25	18	108	70	27	16
COMP07	434	25	20	131	77	6	6
COMP08	324	25	18	86	61	37	37
COMP09	279	25	18	76	75	96	66
COMP10	370	25	18	115	67	4	4
COMP11	162	46	5	30	13	0	0
COMP12	218	36	11	88	150	300	53
COMP13	308	25	19	82	66	59	48
COMP14	275	25	17	85	60	51	51
COMP15	251	25	16	72	68	66	41
COMP16	366	25	20	108	71	18	13
COMP17	339	25	17	99	70	56	44
COMP18	138	36	9	47	52	62	0
COMP19	277	25	16	74	66	57	49
COMP20	390	25	19	121	78	4	0
COMP21	327	25	18	94	78	83	0

educational program which is called CBCT. In this Table, let m be a course within a set of n rooms and a set of h proper timeslots with a set of constraints. Each course; c, includes a number of course topic l_i related to one teacher. Each timeslot p is a pair of daily and weekly timeslot. After introducing Tables 9 and 10 (in appendix), now we are reviewing algorithms which were run according to Table 11 (in appendix) on this dataset. In Tables 9 and 10 COMP01 to COMP24 are the competitions between groups of solutions. Also in Table 11, DS_1 is ITC-2007 CBCT and DS_2 is ITC-2007.

4.5. Comparison of algorithms performances studied in Table 11

Comparison of performance and efficiency of studied algorithms listed in Table 11 is shown in Fig. 3. Performance of algorithms on DS1 is presented in Fig. 3a and results of algorithms on DS2 are shown in Fig. 3b.

5. Discussion

After reviewing approaches which solved UCTTP problem, now we could reach an objective classification of these methods. Approaches based on operational research methods do not have good efficiency in solving such problem, but rather they have easier implementation, since they are analyzed by software integrated with efficient and heuristic algorithms. This is because complexity of operational research methods increases as number of students and universities increases; consequently analysis of these problems lay in the category of NP-hard problems with exponential

Table 11
The performance of algorithms on DS_1[a] = ITC – 2007 CBCT, DS_2[b] = ITC – 2007 datasets.

Competitions	Algorithms																			
	A_{44}		A_{45}		A_{46}		A_{47}		A_{48}		A_{49}		A_{50}		A_{51}		A_{52}		A_{53}	
	DS_1	DS_2	DS_1	DS_2	DS_1	DS_2	DS_1	DS_2	DS_1	DS_2	DS_1	DS_2	DS_1	DS_2	DS_1	DS_2	DS_1	DS_2	DS_1	DS_2
COMP01	24	–	5	5	5	5	5.1	5	6.7	5	27	10	–	5	5	–	13	–	–	5
COMP02	299	–	61.3	51	61	55	65.6	50	142.7	111	131.1	111	–	34	75	–	43	–	–	56
COMP03	270	–	94.8	84	84.5	71	89.1	82	160.3	128	138.4	119	–	70	93	–	76	–	–	79
COMP04	166	–	42.8	37	46.9	43	39.2	35	82	72	90.2	72	–	38	45	–	38	–	–	38
COMP05	456	–	343.5	330	326	309	334.5	312	525.4	410	811.5	426	–	298	326	–	314	–	–	316
COMP06	255	–	56.8	48	69.4	53	74.1	69	110.8	100	49.3	130	–	47	62	–	41	–	–	55
COMP07	253	–	33.9	20	41.5	28	49.8	42	76.6	57	153.4	110	–	19	38	–	19	–	–	26
COMP08	173	–	46.5	41	52.6	49	46	40	81.7	77	96.5	83	–	43	50	–	43	–	–	42
COMP09	271	–	113.1	109	116.5	105	113.3	110	164.1	150	148.9	139	–	99	119	–	102	–	–	104
COMP10	239	–	21.3	16	348	4	36.9	27	81.3	71	101.3	85	–	16	27	–	14	–	–	19
COMP11	220	–	0	0	0	0	0	0	0.3	0	5.7	3	–	0	0	–	0	–	–	0
COMP12	751	–	351.6	333	360.1	343	361.6	351	485.1	442	445.3	4.8	–	320	358	–	405	–	–	342
COMP13	214	–	73.9	66	79.2	73	76.1	68	110.4	622	122.9	113	–	65	77	–	68	–	–	72
COMP14	221	–	61.8	59	65.9	57	62.3	59	99	90	105.9	84	–	52	59	–	54	–	–	57
COMP15	238	–	94.8	84	84.5	71	89.1	82	160.3	128	138	119	–	69	87	–	–	–	–	79
COMP16	236	–	41.8	34	49.1	39	20.2	40	92.6	81	107.3	84	–	38	47	–	–	–	–	46
COMP17	280	–	86.6	83	100.7	91	107.3	102	143.4	124	166.6	152	–	80	86	–	–	–	–	88
COMP18	173	–	91.7	83	80.7	96	73.3	68	129.4	116	126.8	110	–	67	71	–	–	–	–	75
COMP19	276	–	68.8	62	69.5	65	79.6	75	132.8	107	125.4	111	–	59	74	–	–	–	–	64
COMP20	241	–	34.3	27	60.9	47	65	61	97.5	88	179.3	144	–	35	54	–	–	–	–	32
COMP21	346	–	108	103	124.7	106	138.1	123	185.3	174	185.8	169	–	105	117	–	–	–	–	107
COMP22	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
COMP23	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
COMP24	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

Competitions	Algorithms																			
	A_{54}		A_{55}		A_{56}		A_{57}		A_{58}		A_{59}		A_{60}		A_{61}		A_{62}		A_{63}	
	DS_1	DS_2	DS_1	DS_2	DS_1	DS_2	DS_1	DS_2	DS_1	DS_2	DS_1	DS_2	DS_1	DS_2	DS_1	DS_2	DS_1	DS_2	DS_1	DS_2
COMP01	–	5	–	5	–	5	–	5	–	523	–	571	–	61	–	1482	–	15	–	1861
COMP02	–	39	–	50	–	60	–	48	–	342	–	993	–	547	–	1635	–	0	–	2174
COMP03	–	76	–	82	–	81	–	76	–	379	–	164	–	382	–	288	–	391	–	272
COMP04	–	35	–	35	–	39	–	41	–	234	–	310	–	529	–	385	–	239	–	425
COMP05	–	315	–	312	–	31	–	303	–	0	–	5	–	5	–	559	–	34	–	8
COMP06	–	50	–	69	–	45	–	54	–	0	–	0	–	0	–	851	–	87	–	28
COMP07	–	12	–	42	–	21	–	25	–	0	–	6	–	0	–	10	–	0	–	13
COMP08	–	37	–	40	–	41	–	47	–	0	–	0	–	0	–	0	–	4	-	6
COMP09	–	104	–	110	–	102	–	106	–	1102	–	1560	–	0	–	1947	–	0	–	2733
COMP10	–	10	–	9	–	17	–	23	–	515	–	2163	–	0	–	1741	–	0	–	2697
COMP11	–	0	–	0	–	0	–	0	–	246	–	178	–	548	–	240	–	547	–	263
COMP12	–	337	–	351	–	349	–	324	–	241	–	146	–	869	–	475	–	32	–	804
COMP13	–	61	–	68	–	43	–	68	–	0	–	0	–	0	–	675	–	166	–	285
COMP14	–	53	–	59	–	59	–	53	–	0	–	1	–	0	–	804	–	0	–	110
COMP15	–	73	–	82	–	82	–	74	–	0	–	0	–	379	–	0	–	0	–	5
COMP16	–	32	–	40	–	49	–	42	–	0	–	2	–	91	–	1	–	41	–	132
COMP17	–	72	–	102	–	81	–	81	–	0	–	0	–	1	–	5	–	68	–	72
COMP18	–	77	–	68	–	79	–	69	–	0	–	0	–	0	–	3	–	26	–	70
COMP19	–	60	–	75	–	67	–	65	–	121	–	1824	–	1862	–	1868	–	22	–	2268
COMP20	–	22	–	61	–	30	–	35	–	304	–	445	–	1215	–	396	–	2735	–	878
COMP21	–	95	–	123	–	110	–	106	–	36	–	0	–	0	–	602	–	33	–	40
COMP22	–	–	–	–	–	–	–	–	–	1154	–	29	–	0	–	1364	–	0	–	889
COMP23	–	–	–	–	–	–	–	–	–	963	–	238	–	430	–	688	–	1275	–	436
COMP24	–	–	–	–	–	–	–	–	–	274	–	21	–	720	–	822	–	30	–	372

[a] DS_1 is ITC – 2007 CBCT dataset.
[b] DS_2 is ITC – 2007 dataset.

time complexity problem. So if complexity of time and space is not important, these approaches (OR) can be used as a one of solution methods of university course timetabling problem. While in turn, the exploration of the search space of solutions performs more efficiently by applying metaheuristic methods and novel intelligent techniques in analyzing this type of problems. However, we could not call a metaheuristic approach as the best method to solve UCTTP problem, since the used datasets are diverse and the way of applying this type of methods is different as separately or in combination with other methods. While approaches to solve UCTTP problem on hybrid techniques (the combination of metaheuristic methods) and multi agent based methods as a distributed architecture of UCTTP problem are important where in multi agent systems based approaches, the independency of scheduling process, negotiation of agents to remove the interference of event and resources with each other, flexibility of agents to combine with different types of heuristic methods in scheduling of common and single events per department, we could use datasets with different sizes as a test bed for above mentioned methods in order to find a reasonable and complete viewpoint on the structure of these algorithms.

6. Conclusion

In this paper, after comprehensive investigation of available approaches in the study of the UCTTP problem, we have focused on methods based on multi agent systems as a distributed architecture of the UCTTP problem. In this method, we have a general look at related works and how to apply them in solving the UCTTP problem where the advantages of using multi agent systems based approach could include increasing the independency of scheduling each department, independence of departments in scheduling, scalability in a distributed environment and to prevent collision among events/resources and unplanned allocation by negotiation among agents in a distributed environment.

Appendix 1.

Datasets Tables along with Tables of applied algorithms' performance to solve UCTTP problem.

Appendix 2.

Full names of applied algorithms (A_i) in Section 5. A1: (MMAS: MAX-MIN Ant Colony System), A2: (CFHH: Credit Function Hyper Heuristic), A3: (FMHO: Fuzzy Multiple Heuristic Ordering), A4: (VNS-TS: Variable Neighborhood Search – Taboo Search), A5: (RIICN: Randomize Iterative Improvement with Composite nears Algorithm), A6: (GBHH: Graph Based Hyper Heuristic), A7: (HEA: Hybrid Evolutionary Algorithm), A8: (GD: Great Deluge Algorithm), A9: (NLGD: Non-Linear Great Deluge Algorithm), A10: (MMAS – LS: MAX-MIN Ant Colony System – Local Search), A11: (GD – LS: Great Deluge Algorithm – Local Search), A12: (EGD: Extended Great Deluge Algorithm), A13: (Elm – GD: Electromagnetic – Like Mechanism and Great Deluge Algorithm), A14: (DHCOABA: Die-hard Co-Operation Ant Behavior Approach), A15: (HPCA: Hybridization Multi-Neighborhood Ppaper Collision Algorithm and Great Deluge Algorithm), A16: (TB-MA: Taboo Based Meme tic Algorithm), A17: (DSSARR: Dual Sequence Simulated Annealing with Round Robin), A18: (HAS: Harmony Search with Multi – Pitch adjusting Rate Algorithm), A19: (VNS-EMC: Variable Neighborhood Search – Exponentially Monte Carlo), A20: (BB-BCA: Big Bang – Big Crunch optimization Algorithm), A21: (GSGA: Guided Search with Genetic Algorithm), A22: (SS: Scatter Search Algorithm), A23: (PCA: Ppaper Collision Algorithm), A24: (Round Robin Strategy over Multi Algorithm), A25: (MA: Memetic Algorithm), A26: (Variable Neighborhood Search), A27: (THHS: Taboo Based Hyper Heuristics), A28: (LS: Local Search), A29: (EA: Evolutionary Algorithm), A30: (AA: Ant Colony optimization Algorithm), A31: (FA: Fuzzy Approach), A32: (EGSGA: Extended Guided Search with Genetic Algorithm), A33: (GA w LS: Genetic Algorithm with Local Search), A34: (GA: Genetic Algorithm; Rossi-Doria et al.), A35: (SABH: Simulated Annealing Based Heuristic), A36: (ETTS: Efficient TimeTabling Solution with Taboo Search), A37: (TSLS: Three Stage Local Search), A38: (AMLS: Adaptive Memory Local Search), A39: (Dubourg et al. Taboo Search), A40: (Taboo Search by Brigitte Jaumard), A41: (Taboo Search by Gustaro Toro), A42: (GSA – LS: Guided Simulated Annealing – Local Search), A43: (Local Search by Tomas Muller), A44: (IABC: Improved Artificial Bee Colony), A45: (The Constraint Based Solver by Muller), A46: (The Taboo Search Approach by Lu & Hao), A47: (The Constraint Satisfaction Problem by Atsuta), A48: (The Threshold Acceptance Metaheuristic by Geiger), A49: (The Repair Based TimeTable Solver by Clark), A50: (ATS: Adaptive Taboo Search), A51: (The Dynamic Taboo Search), A52: (The Integer Programming Method by Lach & Lubbecke), A53: (The Taboo Search with Relaxed Stopping Condition by Schaerf), A54: (A Hybrid Metaheuristic Approach by Salwani Abdullah), A55: (Incorporating Taboo Search and Local Search by Atsuta et al.), A56: (Great Deluge Algorithm with Kempe Chain by Shaker & Abdullah), A57: (ILS: Iterative Local Search), A58: (HGATS: The Hybrid Approach Hybrid Genetic Algorithm and Taboo search), A59: (Mixed Metaheuristic Approach by Cambazard et al.), A60: (Combination of a General Purpose Constraint Satisfaction Solver, Taboo Search and Iterative Local Search Techniques by Atsuta et al.), A61: (A Hybrid Algorithm by Chiarandini et al.), A62: (Ant Colony Optimization algorithm in Conjunction with A Iterative Local search by Nothegger et al.), A63: (Local Search Based Algorithm by Muller), A_{64}: (SA: Simulated Annealing), A65: (TS: Tabu Search), A66: (GA: Genetic Algorithm).

References

Al-Betar, M. A., Khader, A. T., & Zaman, M. (2012). University Course Timetabling Using a Hybrid Harmony Search Metaheuristic Algorithm. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews, 42*(5), 664–681. http://dx.doi.org/10.1109/TSMCC.2011.2174356.

Abdullah, S., Burke, E.K., & McColloum, B. (2007). A Hybrid Evolutionary Approach to the University Course Timetabling Problem. In *Proceedings of CEC: The IEEE congress on evolutionary computation* (pp. 1764–1768).

Abdullah, S., Burke, E. K., & McColloum, B. (2007b). Using a randomised iterative improvement algorithm with composite neighborhood structures for university course timetabling. metaheuristic – Program in complex systems. *Optimization*, 153–172.

Abdullah, S., Burke, E.K., & McColloum, B. (2005). An Investigation of Variable Neighborhood Search for University Course Timetabling. In *The 2th multidisciplinary conference on scheduling: Theory and applications, NY, USA* (pp. 413–427).

Abdullah, S., & Hamdan, R. (2008). A hybrid approach for university course timetabling. *IJCSNS International Journal of Computer Science and Network Security, 8*(8).

Aladag, C. H., Hocaoglu, G. A., & Basaran, M. (2009). The effect of neighborhood structures on tabu search algorithm in solving course timetabling problem. *Expert Systems with Application, 36*, 12349–12356.

Aladag, C. H. & Hocaoglu, G. (2007). The effect of neighborhood structure and of move types in the problem of course timetabling with the tabu search algorithm. In *Proceedings of the fifth statistics conference* (pp. 14–19).

Alsmadi, O. MK., Abo-Hammour, Z. S., Abu-Al-Nadi, D. I., & Algsoon, A. (2011). *A novel genetic algorithm technique for solving university course timetabling problems*. IEEE.

Alvarez, R., Crespo, E., & Tamarit, J. M. (2002). Design and implementation of a course scheduling system using Tabu search. *European Journal of Operational Research, 137*, 512–523.

Amintoosi, M., & Haddadnia, J. (2005). *Fuzzy C-means clustering algorithm to group students in a course into smaller sections*. Berlin Heidelberg: Springer-Verlag, pp. 47–160.

Asham, G. M., Soliman, M. M., & Ramadan, A. R. (2011). Trans genetic coloring approach for timetabling problem. *Artificial intelligence techniques novel approaches & practical applications*, IJCA, 17–25.

Asmuni, H. (2008). *Fuzzy methodologies for automated university timetabling solution construction and evaluation*, Ph.D. Thesis, School of Computer Science University of Nottingham.

Asmuni, H., Burke, E. K., & Garibaldi, J. M. (2005). Fuzzy multiple heuristic ordering for course timetabling. In *The proceedings of the 5th United Kingdom workshop on computational intelligence (UKCI05), London, UK* (pp. 302–309).

Aubin, J., & Ferland, J. A. (1989). A large scale timetabling problem. *Computers and Operations Research, 16*, 67–77.

Aycan, E., & Ayav, T. (2008). *Solving the course scheduling problem using simulated annealing*. IEEE.

Ayob, M., & Jaradat, G. (2009). Hybrid ant colony systems for course timetabling problems. In *IEEE 2nd conference on data mining and optimization 27–28 October 2009, Selangor, Malaysia* (pp. 120–126).

Bakir, M. A., & Aksop, C. (2008). A 0–1 integer programming approach to a university timetabling problem. *Hacettepe Journal of Mathematics and Statistics, 37*(1), 41–55.

Chaudhuri, A., & Kajal, D. (2010). Fuzzy genetic heuristic for university course timetable problem. *International Journal of Advance Soft Computing and its Applications, 2*(1). ISSN 2074-8523.

Dandashi, A., & Al-Mouhamed, M. (2010). *Graph coloring for class scheduling. Department of Computer Science*. Koura, Lebanon: University of Balamand.

Daskalaki, S., & Birbas, T. (2005). Efficient solutions for a university timetabling problem through integer programming. *European Journal of Operational Research, 160*(2005), 106–120.

Daskalaki, S., Birbas, T., & Housos, E. (2004). An integer programming formulation for a case study in university timetabling. *European Journal of Operational Research, 153*(2004), 117–135.

Deris, S., Omatu, S., & Ohta, H. (2000). Timetable planning using the constraint-based reasoning. *Computers & Operations Research, 27*, 819–840.

Deris, S., Omatu, S., Ohta, H., & Saada, P. (1999). Incorporating constraint propagation in genetic algorithm for university timetable planning. *Engineering Application of Artificial Intelligence, 12*, 241–253.

De Werra, D. (1985). An Introduction to TimeTabling. *European Journal of Operational Research, 19*, 151–162.

Dimopoulou, M., & Miliotis, P. (2001). Theory and Methodology Implementation of a university course and examination timetabling system. *European Journal of Operational Research, 130*, 202–213.

Feizi-Derakhshi, M. R., Babaei, H., & Heidarzadeh, J. (2012). A survey of approaches for university course timetabling problem. In *Proceedings of 8th international symposium on intelligent and manufacturing systems (IMS 2012)* (pp. 307–321). Sakarya University Department of Industrial Engineering, Adrasan, Antalya, Turkey.

Geem, Z. W., Kim, J. H., & Loganthan, G. (2001). A new heuristic optimization algorithm: Harmony search. *Simulation, 76*(2), 60–68.

Golabpour, A., Mozdorani Shirazi, H., Farahi, A., kootiani, M., & beige, H. (2008). A fuzzy solution based on Memetic algorithms for timetabling. *IEEE international conference on multimedia and information technology* (pp. 108–110).

Gaspero, L. D., Missaro, S., & Schaerf, A. (2004). A multi-agent architecture for distributed course timetabling. In *proceedings of the 5th international conference on the practice and theory of automated timetabling (PATAT '04)* (pp. 471–474).

Gotlib, C. C. (1963). The construction of class-teacher timetables. *Proceedings of IFIP Congress, 62*, 73–77.

Hafizah, A. R., & Zaidah, I. (2010). Bipartite graph edge coloring approach to course timetabling. *IEEE*, 229–234.

Jat, N. S. & Shengxiang, Y. (2008). A memetic algorithm for the university course timetabling problem. In *IEEE 20th IEEE international conference on tools with artificial intelligence* (pp. 427–433).

Joudaki, M., Imani, M., & Mazhari, N. (2010). *Using improved Memetic algorithm and local search to solve University Course Timetabling Problem (UCTTP)*. Doroud, Iran: Islamic Azad University.

Kaplansky, E., Kendall, G., Meisels, A., & Hussin, N. (2004). Distributed examination timetabling. In *Proceeding of the 5th international conference on the practice and theory of automated timetabling (PATAT '04)* (pp. 511–516).

Kaplansky, E., & Meisels, A. (2004). Negotiation among scheduling agents for distributed timetabling. In *Submitted to the 5th international conference on practice and theory of automated timetabling (PATAT '04)*, Pittsburgh.

Khonggamnerd, P., & Innet, S. (2009). On improvement of effectiveness in automatic university timetabling arrangement with applied genetic algorithm. *IEEE*.

Kostuch, P. (2005). *The university course timetabling problem with a three-phase approach. In Lecture Notes in Computer science*. Berlin/Heidelberg: Springer, pp. 109–125.

Lewis, M. R. R. (2006). *Metaheuristics for university course timetabling*. Ph.D. Thesis, Napier University.

Mayer, A., Nothegger, C., Chwatal, A., & Raidl, G. (2008). Solving the post enrolment course timetabling problem by ant colony optimization. In *Proceedings of the 7th international conference on the practice and theory of automated timetabling*.

Meisels, A., & Kaplansky, E. (2003). *Scheduling agents-distributed timetabling problems (DisTTP)*. Verlog Berlin Heldelberg: Springer. LNCS 2740, pp. 166–177.

Nandhini & Kanmani, S. (2009). Implementation of class timetabling using multi agents. *IEEE*.

Obit, J. H. (2010). *Developing novel meta-heuristic, hyper-heuristic and cooperative search for course timetabling problems*. Ph.D. Thesis, School of Computer Science University of Nottingham.

Obit, J. H., Landa-Silva, D., Ouelhadj, D., Khan Vun, T., & Alfred, R. (2011). Designing a multi-agent approach system for distributed course timetabling. *IEEE*.

Oprea, M. (2007). MAS_UP-UCT: A multi-agent system for university course timetable scheduling. *International Journal of Computers, Communications & Control, II*(1), 94–102.

Paechter, B., Gambardella, L. M., & Rossi-Doria, O. (2002). International timetabling competition 2002. Metaheuristics Network. <www.idsia.ch/Files/ttcomp2002/>.

Pedroso, J. P. (2003). A multi-agent system for automated timetabling with shared resources. *Faculdade de Ciencias da Universidade do Porto Departamento de Ciencia de Computadores Rua do Campo Alegre 8234150–180 Porto, Portugal*.

Rachmawati, L., & Srinivasan, D. (2005). A hybrid fuzzy evolutionary algorithm for a multi-objective resource allocation problem. In *IEEE Proceedings of the fifth international conference on hybrid intelligent systems*.

Redl, T. A. (2004). *A study of university timetabling that blends graph coloring with the satisfaction of various essential and preferential conditions*. Ph.D. Thesis, Rice University, Houston, Texas.

Rossi, d. O., Sampels, M., Birattari, M., Chiarandini, M., Dorigo, M., Gambardella, L. M., et al. (2003). *A comparison of the performance of different metaheuristics on the timetabling problem*. Berlin Heidelberg: Springer-Verlag. PATAT 2002, LNCS 2740, pp. 329–351.

Selim, S. M. (1988). Split vertices in vertex colouring and their application in developing a solution to the faculty timetable problem. *The Computer Journal, 31*(1), 76–82.

Shahvali Kohshori, M., & Saniee Abadeh, M. (2012). Hybrid genetic algorithms for university course timetabling. *IJCSI International Journal of Computer Science Issues, 9*(2), 2.

Shahvali Kohshori, M., Saniee Abadeh, M., & Sajedi, H. (2011). *A fuzzy genetic algorithm with local search for university course timetabling*. Department of Computer Science and Research Branch, Islamic Azad University, Khouzestan, Iran.

Shatnawi, S., Al-Rababah, K., & Bani-Ismail, B. (2010). Applying a novel clustering technique based on FP-tree to university timetabling problem: A case study. *IEEE*.

Shengxiang, Y., & Jat, N. S. (2011). Genetic algorithms with guided and local search strategies for university course timetabling. *IEEE Transactions on Systems, MAN, and Cybernetics-PART C: Applications and Reviews, 41*, 1.

Socha, K., Knowles, J., & Samples, M. (2002). A max-min ant system for the university course timetabling problem. In *Proceedings of the 3rd international workshop on ant algorithms (ANTS 2002). Lecturer notes in computer science* (Vol. 2463, pp. 1–13). Springer-Verlag.

Srinivasan, S., Singh, J., & Kumar, V. (2011). Multi-agent based decision support system using data mining and case based reasoning. *IJCSI International Journal of Computer Science Issues, 8*(4), 2.

Strnad, D., & Guid, N. (2007). A multi-agent system for university course timetabling. *Applied Artificial Intelligence: An International Journal, 21*(2), 137–153.

Tuga, M., Berretta, R., & Mendes, A. (2007). A hybrid simulated annealing with kempe chain neighborhood for the university timetabling problem. In *6th IEEE/ACIS international conference on computer and information science (ICIS 2007)*.

Turabieh, H., & Abdullah, S. (2009). Incorporating Tabu search into memetic approach for enrolment-based course timetabling problems. *IEEE 2nd conference on data mining and optimization 27–28 October 2009, Selangor, Malaysia* (pp. 115–119).

Wangmaeteekul, P. (2011). *Using distributed agents to create university course timetables addressing essential desirable constraints and fair allocation of resources*. Ph.D. Thesis, School of Engineering & Computing Sciences Durham University.

Welsh, D. J. A., & Powell, M. B. (1967). An upper bound for the chromatic number of a graph and its application to timetabling problems. *The Computer Journal, 10*, 85–86.

Xiang, Y. & Zhang, W. (2008). Distributed university timetabling with multiply sectioned constraint networks. In *Proceedings of the twenty-first international FLAIRS conference*.

Yanga, Y., & Paranjape, R. (2011). A multi-agent system for course timetabling. *Intelligent Decision Technologies Computer Science and Artificial Intelligence, 5*(2), 113–131. IOS Press.

Yanga, Y., Paranjape, R., & Benedicenti, L. (2006). An agent based general solution model for the course timetabling problem. Hakodate, Hokkaido, Japan, AAMAS'06, ACM.

Yanga, Y., Paranjape, R., & Benedicenti, L. (2004). *An examination of mobile agents system evolution in the course scheduling problem*. Regina, Canadian: Electronic and Software Systems Engineering University of Regina.

Yanga, Y., Paranjape, R., Benedicenti, L., & Reedc, N. (2006). A system model for university course timetabling using mobile agents. *Multi-agent and Grid Systems – An International Journal, 2*, 267–275. IOS Press.

Zhang, L., & Lau, S. (2005). Constructing university timetable using constraint satisfaction programming approach. In *IEEE Proceedings of the 2005 International conference on computational intelligence for modeling, control and automation, and international conference on intelligent agents, web technologies and internet commerce (CIMCA-IAWTIC'05), 28–30 November, No. 2* (pp. 55–60).