Journal of Industrial and Systems Engineering
Vol. 6, No. 4, pp 249-271
Winter 2013

Memetic and scatter search metaheuristic algorithms for a multi-objective fortnightly university course timetabling problem: a case study

Nasibeh Movahedfar [1*], Mohammad Ranjbar [2*], Majid Salari[3], Salim Rostami[4]

[1,2,3,4] Department of Industrial Engineering, Faculty of Engineering, Ferdowsi University of Mashhad, P.O. Box: 91775-1111, Mashhad, Iran
[1]na.movahhedfar@stu-um.ac.ir, [2]m_ranjbar@um.ac.ir, [3]msalari@um.ac.ir, [4]Salim.Rostami@gmail.com

ABSTRACT

This paper studies a multi-objective fortnightly university course timetabling problem. In this research, the Industrial Engineering Department of the Ferdowsi University of Mashhad is considered as the case study for investigation. Essentially, four objectives should be optimized where we used the Lp-metric method to aggregate them into a single objective. An integer linear programming model and two metaheuristic algorithms, i.e. memetic and scatter search, have been developed for the studied problem. Comparing the proposed algorithms with the results obtained by CPLEX, indicates the superiority of the scatter search algorithm. In particular, high quality solutions are achieved in a reasonable CPU time.

***Keywords*:** Fortnightly timetabling problem; integer linear programming; memetic algorithm; scatter search.

1. INTRODUCTION

During the last decades, production has spread its inclusion area, so that nowadays it covers providing services beside goods. Thus production scheduling entered to a wide relevant area of scheduling services like timetabling problems which are concerned in optimally assigning some limited resources to servicing tasks over time. This study focuses on a particular field of scheduling problems known as university course timetabling problem which has been proved to be NP-hard (Bardadym, 1996) and has been remained a confusing problem even after many years of research. A reason for this complexity may be the high variety of constraints which come from the various policies of institutions and universities, while, another reason could be the fact that educational rules and methods are changing, so that the models have to be modified. In addition, reaching an optimal solution for a large timetabling problem can be quite time-consuming. Finally, computing facilities have been entirely influential in this issue. Therefore, it is necessary to adopt efficient search algorithms to generate optimal or near optimal solutions.

The general term of university timetabling refers to both *exam* and *course timetabling* problems

* Corresponding Author

(Lewis, 2008). In the exam timetabling problem, the goal is to spread the different exams to the best possible extent for each individual student, while this is to propose a compact and uniform timetable for the course timetabling problem in which recurring timetables (i.e. weekly, fortnightly and etc.) are appreciated. The university course timetabling is the procedure of assigning lectures, presented by professors (instructor, assistant professor, associate professor and professor), into room-time *slot*s subject to a given set of side constraints. Usually, constraints are classified into two categories namely, *hard* constraints and *soft* constraints. An assignment satisfying all hard constraints is called a *feasible* timetable. The objective of each timetabling problem is to minimize the total number of soft constraints, violated in a feasible timetable.

The class of scheduling problems includes a wide variety of problems such as machine scheduling, events scheduling, personnel scheduling and many others (see, e.g. Brucker et al., 1999 and Pinedo, 2008). Many real world scheduling problems are multi-objective by nature for which several objectives have to be considered simultaneously (see, e.g. Bagchi, 1999; Ehrgott and Gandibleux, 2000; and T'kindt and Billaut, 2002). Examples of such objectives are: minimizing the length of the schedule, optimizing the utilization of the available resources, satisfying the preferences of human resources (personnel scheduling), minimizing the tardiness of orders (production scheduling), maximizing the compliance with regulations (educational timetabling) and etc. Over the years, several approaches have been used to deal with the various objectives in such problems. Traditionally, the most common approach is to aggregate the multiple objectives into a single scalar value by using weighted aggregating functions according to the preferences set by the decision-maker (see, e.g. Belton and Stewart, 2002; and T'kindt and Billaut, 2002). However, in many real multi-objective scheduling problems, it is preferable to present various compromise solutions to the decision-makers, so that the most adequate schedule can be chosen. Although this can be achieved by performing the search several times using different preferences each time, another approach is to generate the set of compromise solutions in a single execution of the algorithm. The latter strategy has attracted the interest of researchers for investigating the Pareto set of solutions to multi-objective scheduling problems (see, e.g. Bagchi, 1999; and Ishibuchi et al., 2002). The aim in Pareto optimization is to find a set of compromise solutions that represents a good approximation to the Pareto optimal front (see Rosenthal, 1985). In recent years, the number of algorithms proposed for Pareto optimization has increased tremendously mainly because multi-objective optimization problems exist in almost any domain (see, e.g. Ehrgott and Gandibleux, 2000, Jones et al., 2001 and Tan et al., 2002). The main weakness of the Pareto optimization technique is that when the number of objectives increases, the computational works will be intractable and also analyzing the results will be very complicated. Thus, this technique is usually used for bi-objective optimization problems.

A large variety of metaheuristic algorithms have been applied to the course timetabling problems. Examples of these algorithms include genetic algorithm (Wang, 2003), simulated annealing (see Ceschia et al. ,2012), tabu search (see Lü and Hao, 2010), particle swarm optimization (see Shiau, 2011), graph coloring (see Burke et al., 2007) and hybrid variable neighborhood approach (see Burke at al., 2010). Interested readers are referred to Lewis (2008) for a comprehensive survey of metaheuristic-based techniques for the university timetabling problems.

In contrast to the metaheuristic algorithms, the literature of exact solution methods in the context of timetabling is very limited, because, these approaches cannot tackle large scale timetabling problems. Integer programming approach is one of the most used exact approaches for the course timetabling problem (see, e.g. Daskalaki et al., 2004; Daskalaki and Birbas, 2005; Mirhassani, 2006; and Boland et al., 2008).

In this paper, we study the fortnightly university course timetabling problem (FUCTP) for the Ferdowsi University of Mashhad. In each semester, we face the challenging timetabling problem which varies from a semester to another because some professors prefer to change their courses in some academic years. Both professors and students prefer to have a compact and uniform timetable. Also, students have two other preferences as follows: (1) they prefer to participate in more hard-understanding lectures in the morning; (2) also students prefer that non-precedent courses do not overlap such that they have more options for course selection. We need to develop a timetable which considers the aforementioned objectives. This task has been done manually already using try and error method. In this study, we try to improve the quality of the developed timetable using optimization techniques. For this purpose, we aggregate the four conflicting objectives into a single objective by applying the L_p-metric method.

Each lecture of a course has to be presented in a 120-minutes time slot where 100 minutes of that is assigned for teaching and the remaining 20 minutes is dedicated to the gap between two consecutive lectures. Courses are partitioned into two subsets. Particularly, the first and the second subsets include the courses which must be presented 120 and 180 minutes in each week, respectively. Thus, each course may need 1 or 1.5 lecture per week. Since it is not possible to assign half of a time slot to a lecture, the total frequency period of the timetable is set to two weeks in which each course may need two or three lectures per two-weeks. In contrast to weekly timetables in which course timetable is identical for all weeks, in fortnightly timetables there are *odd* and *even* weeks. For those courses needing two lectures per two-weeks, their corresponding timetables are identical in odd and even weeks. In contrast, for the courses with three lectures per two-weeks, the timetable for odd and even weeks are different such that each course has a fixed lecture per week and an alternate lecture per two weeks (either in odd or in even weeks).The Engineering Faculty of the Ferdowsi University includes seven departments where each department has a given number of identical rooms, all having the same capacity and other equipments. Moreover, each department can share its rooms with other departments where, a central educational office coordinates the sharing. Thus, it is assumed that for each department, a minimum number of identical rooms are available in each time slot but they may be increased in different time slots. In this paper, we focus on developing an optimal timetable for the Industrial Engineering Department presenting Bachelor of Science (B.Sc.) and Master of Science (M.Sc.) programs for which three identical rooms are available in each time slot.

The contributions of this article are twofold: (1) we develop an integer linear programming model for the FUCTP and solve it using ILOG CPLEX 12.3 solver; (2) we develop a memetic algorithm (MA) and also a scatter search (SS) to find optimal or near optimal solutions.

The remainder of this paper is organized as follows. In Section 2, a formal description of the problem and an integer linear programming formulation are provided. Section 3 is dedicated to the description of the MA and the SS. Computational experiments are reported in Section 4. Finally, conclusions and suggestions for future works are presented in Section 5.

2. PROBLEM DESCRIPTION AND FORMULATION

In order to define the FUCTP, assume we are given *n* courses gathered in set $C = \{1, \dots, n\}$which is divided into two subsets C^2 and C^3 in terms of number of the required lectures per two-weeks. In particular, $C^2 = \{i: u_i = 2\}$ and $C^3 = \{i: u_i = 3\}$ where, for each $i \in C$, u_i indicates the number of required lectures per two-weeks. These two subsets are mutually exclusive and jointly exhaustive. In another point of view, we divide set *C* into 4 families where each family $F_f$indicates a set of coursesthat are usually attended by a group of students with

identical entrance year. Also, the families are mutually exclusive $(\forall f, f': F_f \cap F_{f'} = \emptyset)$and jointly exhaustive$(\bigcup_f F_f = C)$. All families are determined based on the curriculum of the Ferdowsi University. It should be noticed that it is possible for a student to take courses from different families. Let $P = \{1, \dots, m\}$ represent the set of professors in which each professor$j \in P$must present a subset of courses shown by PC_j. Moreover, each lecture of a course $i \in C$should be assigned to a time slot t where $t \in \{1, \dots, 60\}$. Number of time slots has been determined by considering six time slots in each working day (i.e. [8-10], [10-12], [12-14], [14-16], [16-18] and [18-20]) and ten working days in a two- week period in which the union of time slots $\{6(d-1)+1, 6(d-1)+2, \dots, 6d\}$ constitutes the day d.

It should be noticed that each lecture of a course $i \in PC_j$ can be assigned to a time slot$t$for which professor j is available on that time slot. In order to determine the optimal time slot for each lecture of a course, we consider preferences of students and professors, shown by the set *PF*. The relative preference for assigning a lecture of course *i* to the time slot is determined based on both professors and students' opinion.

As a set of hard constraints, the courses of a family F_f could not overlap but there may be courses from different families which are attended by a noticeable number of students. Particularly, we consider a penalty, shown by $oc_{ii'}$, for overlapping of courses $i \in F_f$ and $i' \in F_{f'}$ where $f \neq f'$.The overlapping costs are determined based on the students' enrollment data. Also, in order to have a compact program for both students and professors, we consider a cost for each busy day for students and professors, respectively.

On the other hand, for constructing a uniform timetable we consider an upper bound on the maximum number of lectures participated by a student (ls^{max}) or presented by a professor (lp^{max}) in each day. Since some courses are presented by professors coming from outside of the department, we assume that the timetable of these courses has been pre-assigned in advance. Descriptions of the parameters introduced for the model are provided in Table 1 while Table 2 gives the definition of the variables used to model the problem.

Table1 Set of parameters used to model the FUCTP

Parameter	***Description***
$C = \{1, \dots, n\}$	The set of courses,
C^2	The set of courses having two lectures per two-weeks$(C^2 \subseteq C)$,
C^3	The set of courses having three lectures per two-weeks$(C^3 \subseteq C)$,
$\mathbf{F} = \{F_f\}$	The set of families,
$P = \{1, \dots, m\}$	The set of professors,
$\mathbf{PC} = \{PC_j\}$	The set of professor-course where PC_j indicates a subset of courses presented by professor $p_j; j = 1, \dots, m$,
$TS = \{1, \dots, 60\}$	The set of time slots,
$\mathbf{D} = \{1, \dots, 10\}$	The set of days,
$\mathbf{PTS} = [pts_{jt}]$	$pts_{jt} = 1$if professor $j \in P$ is available at time slot$t \in TS$, and zero otherwise,
$PF = \{pf_{it}\}$	The set of preferences, where,pf_{it} indicates the relative preference of professors and students to participate

Parameter	***Description***
	in course $i \in C$ duringtime slot$t \in TS$,
$OC = \{oc_{ii'}\}$	The set of overlapping costs where $oc_{ii'}$ indicates overlapping cost,
u_i	The number of lectures of course $i \in C$ in two weeks,
ls^{max}	The maximum number of lectures for a family of students in a day,
lp^{max}	The maximum number of lectures for a professor in a day,
pa_{it}	$= \begin{cases} 1; \text{If course } i \in C^2 \text{ is preassigned to the time slot } t; t = 1, \dots, 60 \\ 0; \text{Otherwise,} \end{cases}$
pa_{it}^1	$= \begin{cases} 1; \text{If the fixed lecture of course } i \in C^3 \text{ is preassigned to the time slot } t; t = 1, \dots, 60 \\ 0; \text{Otherwise,} \end{cases}$
pa_{it}^2	$= \begin{cases} 1; \text{If the alternative lecture of course } i \in C^3 \text{ is preassigned to the time slot } t; t = 1, \dots, 60 \\ 0; \text{Otherwise,} \end{cases}$

Table2. Set of variables used to model the FUCTP

Variable	***Definition***
X_{it}	$= \begin{cases} 1; \text{If course } i \in C^2 \text{ is assigned to the time slot } t \\ 0; \text{Otherwise,} \end{cases}$
X_{it}^1	$= \begin{cases} 1; \text{If the fixed lecture of course } i \in C^3 \text{ is assigned to the time slot } t \\ 0; \text{Otherwise,} \end{cases}$
X_{it}^2	$= \begin{cases} 1; \text{If the alternative lecture of course } i \in C^3 \text{ is assigned to the time slot } t \\ 0; \text{Otherwise,} \end{cases}$
$Y_{ii'}$	$= \begin{cases} 1; \text{If two diffrent course } i \text{ and } i' \text{ are overlapped for at least one time slot} \\ 0; \text{Otherwise,} \end{cases}$
W_{jd}	$= \begin{cases} 1; \text{If at least a lecture of a course from seubset } PC_j \text{ is assigned to day } d \\ 0; \text{Otherwise,} \end{cases}$
Z_{fd}	$= \begin{cases} 1; \text{If at least a lecture of a course from seubset } F_f \text{ is assigned to day } d \\ 0; \text{Otherwise,} \end{cases}$

There are four objectives f_1 to $f_4$in our introduced model, i.e. maximization of the courses' profits (f_1), minimization of the overlapping costs(f_2), minimization the number of professors' busy days (f_3)and minimization the number of students' busy days (f_4). These four objectives are described mathematically as follows:

$$f_1 = \sum_{t=1}^{60} \sum_{i \in C^2} pf_{it} X_{it} + \sum_{t=1}^{60} \sum_{i \in C^3} pf_{it} \left(X_{it}^1 + X_{it}^2\right),$$

$$f_2 = \sum_{i=1}^{n} \sum_{i'=i+1}^{n} oc_{ii'} Y_{ii'},$$

$$f_3 = \sum_{d=1}^{10} \sum_{f=1}^{4} Z_{fd} \ ,$$

$$f_4 = \sum_{d=1}^{10} \sum_{j=1}^{m} W_{jd} \ .$$

This problem is referred to as a Multi-Objective Decision Making (MODM) problem in the literature in which each objective function may have a different weight and scale. The weighted metric method, referred to as L_P-metric method (Triantaphyllou, 2000), is developed to solve such a multi-objective problems. The L_P metric is a MODM optimization technique that aggregates multiple objectives into a single one by multiplying each objective function k by its corresponding weight w_k. In a maximization problem, the weighted L_P-metric distance measure of any f_k from its ideal value f_k^+can be minimized as follows $\{\sum_{k=1}^{4} w_k (f_k^* - f_k)^p\}^{1/p}$ where w_k is a non-negative weight assigned to each objective function by the decision maker and p indicates the importance of each objective function's deviation from its ideal value. We use $p = 1$and the resulting problem reduces to a weighted sum of the deviations. In order to obtain f_k^+, only the objective function f_k is included in the model while other objective functions are excluded. Then, the objective value of the optimal solution results in f_k^+. It should be noticed that objective functions f_1 to f_4 do not have the same scale. So each objective function f_k is made scale-less as follows. We define f_k^- as the worst value of f_k when only f_k is included in the objective function. Now, we consider $\frac{f_k^+ - f_k}{f_k^+ - f_k^-}$ instead of f_k in the objective function to prevent the problems of different scales.The FUCTP reads as follows:

$$MinO.F. = w_1 \frac{f_1^+ - f_1}{f_1^+ - f_1^-} + w_2 \frac{f_2 - f_2^+}{f_2^- - f_2^+} + w_3 \frac{f_3 - f_3^+}{f_3^- - f_3^+} + w_4 \frac{f_4 - f_4^+}{f_4^- - f_4^+} \quad (1)$$

s.t.

$$\sum_{t=1}^{30} X_{it} = 1; \quad \forall i \in C^2 \quad (2)$$

$$X_{i(t+30)} = X_{it}; \ \ \forall t \in \{1, \dots, 30\} \text{ and } \forall i \in C^2 \quad (3)$$

$$\sum_{t=1}^{30} X_{it}^1 = 1; \quad \forall i \in C^3 \quad (4)$$

$$X_{i(t+30)}^1 = X_{it}^1; \quad \forall t \in \{1, \dots, 30\} \text{ and } \forall i \in C^3 \quad (5)$$

$$\sum_{t=1}^{60} X_{it}^2 = 1; \quad \forall i \in C^3 \quad (6)$$

$$X_{it} \leq pts_{jt}; \quad \forall i \in C^2, \forall j: i \in PC_j \text{ and } \forall t \in TS \quad (7)$$

$$X_{it}^1 \leq pts_{jt}; \quad \forall i \in C^3, \forall j: i \in PC_j \text{ and } \forall t \in TS \quad (8)$$

$$X_{it}^2 \leq pts_{jt}; \quad \forall i \in C^3, \forall j: i \in PC_j \text{ and } \forall t \in TS \quad (9)$$

$$\sum_{t=\tau}^{\tau+11}\left(X_{it}^{1}+X_{it}^{2}\right)\leq 1;\quad \forall i\in C^{3}\text{ and }\tau\in\{1,7,13,19,31,37,43,49\} \tag{10}$$

$$X_{it}\geq pa_{it};\quad \forall i\in C^{2}\text{ and }t\in\{1,\ldots,30\} \tag{11}$$

$$X_{it}^{1}\geq pa_{it}^{1};\quad \forall i\in C^{3}\text{ and }t\in\{1,\ldots,30\} \tag{12}$$

$$X_{it}^{2}\geq pa_{it}^{2};\quad \forall i\in C^{3}\text{ and }t\in TS \tag{13}$$

$$\sum_{\forall i\in C^{2}}X_{it}+\sum_{\forall i\in C^{3}}\left(X_{it}^{1}+X_{it}^{2}\right)\leq 3;\ \ \forall t\in TS \tag{14}$$

$$\sum_{\forall i\in C^{2}\&i\in PC_{j}}X_{it}+\sum_{\forall i\in C^{3}\&i\in PC_{j}}\left(X_{it}^{1}+X_{it}^{2}\right)\leq 1;\quad \forall t\in TS\text{ and }\forall j\in P \tag{15}$$

$$Y_{ii'}\geq\frac{\left(X_{it}+X_{i't}+X_{it}^{1}+X_{i't}^{1}+X_{it}^{2}+X_{i't}^{2}-1\right)}{5};\ \forall i\neq i'\in C\text{ and }\forall t\in\{1,\ldots,60\} \tag{16}$$

$$\textstyle\sum_{\forall i\in C^{2}\&\ i\in F_{f}}X_{it}+\sum_{\forall i\in C^{3}\&\ i\in F_{f}}\left(X_{it}^{1}+X_{it}^{2}\right)\leq 1;\ \ \forall t\in TS\text{ and }f=1,\ldots,5 \tag{17}$$

$$\sum_{t=\tau}^{\tau+5}\sum_{i\in\left(C\cap PC_{j}\right)}\left(X_{it}+X_{it}^{1}+X_{it}^{2}\right)\leq lp^{max};\quad \forall\tau\in\{1,7,13,19,25,31,37,43,49,55\}\text{ and }\forall j\in P \tag{18}$$

$$\sum_{t=\tau}^{\tau+5}\sum_{i\in\left(C\cap F_{f}\right)}\left(X_{it}+X_{it}^{1}+X_{it}^{2}\right)\leq ls^{max};\quad \forall\tau\in\{1,7,13,19,25,31,37,43,49,55\}\text{ and }\forall F_{f}\in\mathbf{F} \tag{19}$$

$$Z_{fd}\geq\frac{\left(X_{it}+X_{it}^{1}+X_{it}^{2}\right)}{3};\ \forall i\in\left(C\cap F_{f}\right),\ \ \forall F_{f}\in\mathbf{F},\forall t:\left\lceil t/6\right\rceil=d\text{ and }\forall d\in D \tag{20}$$

$$W_{jd}\geq\frac{\left(X_{it}+X_{it}^{1}+X_{it}^{2}\right)}{3};\ \forall i\in\left(C\cap PC_{j}\right),\ \ \forall F_{f}\in\mathbf{F},\forall t:\left\lceil t/6\right\rceil=d\text{ and }\forall d\in D \tag{21}$$

$$X_{it}\in\{0,1\};\quad \forall i\in C^{2}\text{and }\forall t\in TS \tag{22}$$

$$X_{it}^{1},X_{it}^{2}\in\{0,1\};\quad \forall i\in C^{3}\text{and }\forall t\in TS \tag{23}$$

$$Y_{ii'}\in\{0,1\};\quad \forall i,i'\in C \tag{24}$$

$$Z_{fd}\in\{0,1\};\quad \forall F_{f}\in\mathbf{F}\text{ and }\forall d\in TS \tag{25}$$

$$W_{jd}\in\{0,1\};\quad \forall j\in P\text{ and }\forall d\in TS \tag{26}$$

The objective function (1) is to minimize the total weighted sum of objective functions from their ideal values. For each $i\in C^{2}$, constraint (2) indicates that a time slot of the first (odd) week must be assigned to the first lecture while constraint (3) imposes the similar restriction for the second lecture of this course in the second (even) week. Similarly, for the fixed lecture of a course $i\in C^{3}$, we have constraints (4) and (5) but in order to assign a time slot to the alternative lecture of this course, constraint (6) is applied. For each $i\in C$ and $j\in P$, constraints (7) to (9) indicate that each lecture of

a course i where $i \in PC_j$must be presented in a time slot$t$ in which the professor j is available. On the basis of the Educational Office's rules, two lectures of a course $i \in C^3$are not allowed to be presented in the same day or in two consecutive days and such is imposed to the model by using constraint (10). Since it is not known whether the alternate lecture of such a course is presented in odd or even weeks, τ gets its values from$\{1,7,13,19,31,37,43,49\}$. It must be noticed that we have excluded τ=25 because there are always two days between the last working day of the first week and the first working day of the second week. The constraints (11) to (13) indicate pre-assignments. Constraint (14) shows that the number of parallel lectures in a time slot cannot be greater than the three available rooms on that time slot. Also, each professor can present at most one lecture in each time slot where such is imposed by applying constraint (15). The constraint (16) is related to the overlapping issue in which if a lecture of course i overlaps with a lecture of course i', then we must have $Y_{ii'} = 1$. In order to prevent overlapping of courses of a given family, constraint (17) is applied. The constraints (18) and (19) impose a threshold on the maximum number of lectures in a day for students and professors, respectively. In order to determine busy days for each family of students, we need the constraint (20) in which $\lceil * \rceil$ indicates the smallest integer larger than or equal to *. Similarly, the constraint (21) determines the busy days of each professor. Finally, constraints (22) to (26) restrict the variables to be binary.

3. SOLUTION APPROACHES

Due to the high complexity of the timetabling problems especially in large scale instances, heuristic and metaheuristic algorithms seem to be more efficient than exact solutions approaches. In this section, we develop two population-based metaheuristic algorithms, i.e. memetic and scatter search, for which most of their components are identical and they differ mainly in the generation strategy of new populations. The details concerning the proposed techniques are provided in the following sections.

3.1. Memetic algorithm

The MA is the genetic algorithm (GA) hybridized with local search ideas (see Neri et al. 2011). The GAs and MAs are successfully used in solving a set of difficult search and optimization problems, including the real-world timetabling problems (see, e.g. Paechter et al., 1998, Krasnogor, 2002, and Alkan and Ozcan, 2003).

Figure 1 represents the pseudo-code of the MA, developed for the FUCTP. At first, an initial population set *POP* of size *psize* is generated using *the initial population generation method*, described in section 3.1.2. Next, for each parent chromosome P_1, denoting i^{th} chromosome of *POP* and shown as *POP*(*i*), we apply the selection phase in which another parent chromosomes P_2 will be selected based on the *selection procedure*, described in section 3.1.3. In step 5, the two-point crossover operator, described in section 3.1.4, is applied to parents P_1 and $P_1$to generate children Q_1 and Q_2. Following this step, the mutation operator, described in section 3.1.5, is applied over each child chromosome by probability of p_{mut}. If generated children are feasible, they are added to the *POP*, otherwise; the repairing operator, described in section 3.1.6, is applied over each infeasible generated child in step 5.

1.Generate an initial population *POP* with size of *pszie* using the *initial population generation method*.

2. $i = 1$.

3. $P_1 \leftarrow POP(i)$.

4. Select P_2 from POPbased on the *selection procedure* where $P_1 \neq P_2$.

5. Apply the two-point crossover over two parent chromosomes $P_1$and $P_2$to obtain children chromosomes Q_1 and Q_2. Next, apply the mutation operator to Q_1 and Q_2 by probability of p_{mut}.

6. If Q_1 and $Q_2$are infeasible, apply the repairing operator to infeasible individuals to obtain feasible chromosomes R_1 and R_2. Next, add R_1 and R_2 to the *POP*.

7. If $i < psize$, let $i = i + 1$ and go to step 3.

8. Apply the local search operator to each of chromosomes of *POP* by probability of p_{ls}.

9. Keep the best *psize* of *POP*'s chromosomes and remove the others.

10. If the termination criterion is met, return the best element of *POP* as the solution and stop; else, go to step 2.

Figure 1: the pseudo-code of MA developed for the FUCTP

The repaired feasible children, shown as R_1 and R_2, are added to the *POP*. Steps 3 to 6 are repeated for all chromosomes $1, \dots, psize$. In step 8, the local search operator, described in section 3.1.6, is applied to each chromosome of *POP* by probability of p_{ls}. In step 9, the best *psize* chromosomes of *POP* are kept and the remaining chromosomes are removed. If the termination criterion, considered as a maximum time limit, is met the best solution of *POP* is retuned as the best found solution, otherwise; the algorithm is restarted from step 2.

3.1.1. Solution representation

In the FUCTP, each solution is represented by a two-dimensional chromosome including three rows and sixty columns where each row indicates a classroom and each column indicates a time slot. Each cell is either empty or assigned to a lecture of a course for which the corresponding course number is written inside the cell. An illustrative example is given in Figure 2. In this figure, the fixed lectures of the three-unit course 9 are presented in time slots 1 and 31 while its alternate lecture is in time slot 30. Also, the second cell of time slot 1 is allocated to the alternate lecture of course 25 since it is not repeated in time slot 31.

	1	2	…	30	31	…	60
1	9			9	9		
2	25						
3							

Figure 2: Solution representation

3.1.2. The initial population generation method

The initial population is generated based upon a biased random sampling method. Before description of this procedure, we define the flexibility of a course *i* as the total number of time slots to which course *i* could be assigned. The pseudo-code of the initial population generation method is provided in Figure 3. At first, the set of available courses *C* are sorted in non-decreasing order of their flexibility (we use course numbers as a tie breaker). Starting from the first unassigned course *i* in the sorted list, a time slot is selected to assign the fixed lecture of course *i*. Essentially, assuming $i \in F_f$ and to be presented by professor *j*, in step 4, for the fixed lecture of course *i*, all feasible time slots$(ATSFL_i)$ will be determined. Assuming time slot *t* is in odd (even) weeks and corresponding to the constraints (3) and (5), $t \in ATSFL_i$ if and only if $t + 30$ $(t - 30) \in ATSFL_i$. If $ATSFL_i$ is empty, we should stop because no feasible solution can be obtained. Otherwise, a time slot $t \in ATSFL_i$ is selected based on the *roulette-wheel* rule (see step 5). Roulette-wheel is a biased random selection method in which the probability selection of each time slot corresponds to its profit value. After selection a time slot *t*, the two fixed lectures of course *i* are assigned to the time slots *t* and *t*+30 (*t*-30). Also, these two time slots will be inaccessible for professor *j* and all courses of family F_f (see step 6).

1. Sort set *C* based on the non-decreasing order of the courses flexibility degree.
2. *i*=1.
3. Select course $i \in C$. Assume course *i* is a member of family F_f and presented by professor *j*.
4. Determine all available time slots to which the fixed lecture of course *i*could be allocated $(ATSFL_i)$. If $ATSFL_i = \emptyset$, stop (no feasible solution can be obtained).
5. Select a time slot $t \in ATSFL_i$ based on roulette-wheel rule. If *t* belongs to the odd (even) weeks, assign course *i* to the time slots *t* and *t*+30 (*t*-30).
6. Make time slots *t* and *t*+30 (*t*-30) inaccessible for professor *j* and all courses of family F_f.
7. Assume time slot *t* is placed in day *d*. If the number of assigned lectures to day *d* for professor *j*(family F_f) equals $lp^{max}(ls^{max})$, make all other times slots of this day inaccessible for the professor *j* (family F_f).
8. If $i \in C^3$, determine time slots for alternate of course *i* $(ATSAL_i)$. If $ATSAL_i = \emptyset$, stop (no feasible solution can be obtained). Otherwise, select a time slot $t' \in ATSAL_i$ based on roulette-wheel rule and assign alternate lecture of course *i* to that. Similar to steps 6 and 7, update accessible time slots for professor *j* and family F_f and remove course *i* from set *C*.
9. If $C = \emptyset$, stop (a feasible solution has been obtained), otherwise; let *i*=*i*+1 and go to step3.

Figure 3 Pseudo-code of the initial population generation method

According to the constraints (22) and (23), in step 7 we check to see if the number of assigned lectures to day *d*, which includes time slot *t*, is equal to$lp^{max}(ls^{max})$ for professor *j* (family F_f). If

such equality holds, other time slots of that day have to be inaccessible for professor j (family F_f). Step 8 is related to the assignment of the alternate lecture of each course $i \in C^3$. Similar to $ATSFL_i$, we define $ATSAL_i$ as the set of all feasible time slots for assigning the alternate lecture of course $i \in C^3$. A feasible time slot $t' \in ATSAL_i$, based on roulette-wheel rule, is selected to assign course i. Steps 3 to 8 are repeated until lectures of all courses are assigned to the suitable time slots and a feasible solution is obtained. As it has been indicated in steps 4 and 8, this procedure does not necessarily lead to a feasible solution.

3.1.3. The selection procedure

After generation of the initial population, for each parent chromosomes P_1, a partner chromosome P_2 should be selected. The selection procedure is based on *roulette-wheel* rule (Mitchell, 1998), described in the previous section. In this stage, the selection chance of each chromosome corresponds to its objective function. Thus, chromosomes with better objective cost have more chance to be selected as the partner.

3.1.4. The crossover operator

We choose to work with the well-known two-point crossover to generate the children chromosomes (Q_1 and Q_2) from parent chromosomes (P_1 and P_2)(Mitchell, 1998). For this purpose, two random integers r_1 and r_2 are taken from [1,60]. Following this step, all cells corresponding to chromosome Q_1 between columns r_1 and r_2 are copied from chromosome P_1 and the other cells are copied from chromosomes P_2. Chromosome Q_2 is generated similar to chromosome Q_1 by exchanging the role of chromosome P_1 and P_2. It is obvious that children chromosomes Q_1 and Q_2 may be infeasible and need to be repaired by the repairing operator, described in section 3.1.6.

3.1.5. The mutation operator

The mutation operator is applied over each generated child chromosome with probability p_{mut} which is an input parameter. This operator selects two set of cells, each including five genes, randomly, and exchange each cell of the first set by the corresponding cell in the second set.

3.1.6. The repairing operator

Assume we are given a timetable in which some of the hard constraints are violated. In order to repair this timetable, we keep in the timetable all courses for which no hard constraint is violated. Also, for those courses that their lectures number is more than predetermined values, we delete extra time slots starting from these having less flexibility. In addition, if there is a course for which the number of fixed lectures is one and the corresponding time slot in the other week is empty, that time slot is assigned to the second fixed lecture of this course. We remove all other courses for which at least one hard constraint has been violated. Now, we have a partial timetable and a set of unassigned courses. To make a feasible solution, the algorithm follows the *initial population generation method* to stop with a feasible or infeasible solution but with two differences. The first difference is that in step 1, in which the set *C* includes only unassigned courses. The second difference is related to the steps 5 and 8 of Figure 3 where from the set of feasible time slots, one is selected randomly based on roulette-wheel rule. In the repairing operator, we define the flexibility degree for each feasible time slot *t* as the total number of courses which could be assigned to it. Based on this definition, course $i \in C$ is assigned to the time slot *t* where it has the least flexibility among all time slots $t \in ATSFL_i$.

3.1.7. The local search operator

The local search operator is an iterative procedure in which the rescheduling of all courses will be considered and the best feasible substitution, i.e. the one with the greatest impact on the objective function, will be implemented. When a course i is selected to be rescheduled, exchanging the time slots of this course with empty time slots and also with possible time slots of another course i' is considered while other courses remain unchanged in their original places. Starting from the first course, this process is repeated for all available courses and the cost and benefit of each substitution is calculated. Finally, the best feasible move, having the most improvement in the objective function improvement is selected to be applied on the original timetable. The local search process is repeated with the new timetable and will be stopped, whenever, there is no improvement in the objective function by rescheduling all available courses.

3.2. Scatter search algorithm

Scatter search is an evolutionary method that was first introduced by Glover (1977) for integer programming problems. This algorithm uses strategies in which both diversification and intensification of solutions are maintained using combination strategies as opposed to probabilistic learning approaches. For a general introduction to SS, we refer the reader to Laguna and Marti´ (2003). The general framework of the developed SS algorithm is depicted in Figure 4.

1. Construct an initial population *POP* of size *psize* using the initial population generation method.
2. Build the reference set using the *RefSet* building method.
3. Generate *Newsubsets* with the subset generation method. Set $P=\emptyset$.

while (*Newsubsets*$\neq\emptyset$) **do**

4. Select the next subset σ in *NewSubsets*.
5. Apply the crossover operator to σ to obtain two new solutions.
6. Apply the repairing operator over each new generated solution and add the new obtained solutions to the *POP*.
7. Apply the local search operator over each new element of *POP* by probability of p_{ls}.
8. Delete σ from *Newsubsets*.

end while

9. If one of the termination criteria is met, stop. Otherwise, go to 2.

Figure 4 Scatter search algorithm

In the first step, we generate an initial population *POP* containing *psize* solutions using the initial population generation method, described in Section 3.1.1. In the second step, we construct the reference set *RefSet* including $RefSet_1$ and $RefSet_2$ where the former containing b_1 solutions with low objective functions and the latter containing b_2 solutions with high diversity ($b=b_1+b_2$). The solutions of $RefSet_1$ and $RefSet_2$ are called reference solutions. Next, the algorithm generates the *NewSubsets*, each of them containing two reference solutions. The Reference set building and subset generation methods are described in section 3.2.1. Subsequently, the two solutions of each

subset are combined, and new solutions are generated using the uniform crossover operator, described in section 3.2.2. Since generated solutions in step 5 may not be feasible, the repairing operator, described in section 3.1.5, is applied over each new generated infeasible solution. In order to improve the new generated solutions, we perform a local search around each of them with local search probability p_{ls}. Steps 2 to 8 are repeated until the termination criterion, a given time limit, is reached.

3.2.1. Reference set building and subset generation methods

The set of reference solutions (*RefSet*), includes both high quality and diverse solutions. The construction of high-quality solutions, $RefSet_1$, starts with the selection of the solution in *POP* with the best objective functions (ties are broken randomly). This solution is added to $RefSet_1$ and removed from *POP*. Following this step, the algorithm proceeds by selecting from *POP*, the solution with the best objective function (ch) and $D_{min}(ch) \geq th\_dist_1$, where $D_{min}(ch)$ is the minimum distances of solution ch from those currently in $RefSet_1$ and $th_dist_1$is a threshold distance. Essentially, the distance between two solutions is equal as the total number of identical lectures in different time slots divided by 60. This process is repeated until $b_1$solutions are selected for $RefSet_1$. To construct diverse solutions ($RefSet_2$), we follow the same strategy as that already explained for solutions 2 to b_1 of $RefSet_1$, but with$th_dist_2 > th_dist_1$, and with $D_{min}(ch)$ as the minimum distance to the solutions in both $RefSet_1$ and $RefSet_2$. Therefore, both $RefSet_1$ and $RefSet_2$ contain diversified solutions, with more emphasis on diversification in $RefSet_2$. When no qualified solution can be found in the population, we populate *RefSet* with randomly generated solutions by following the initialization procedure, explained in Section 3.1.2. In this case, we do not check the minimum threshold distance condition for the generated solutions.

Upon constructing the *RefSet*, the algorithm proceeds by generating the *NewSubsets*. Essentially, *NewSubsets* contains all possible selection of two solutions belonging to $RefSet_1$ or one solution from $RefSet_1$ and the other from $RefSet_2$. Assuming $|RefSet_1| = b_1$ and $|RefSet_2| = b_2$, then $|NewSubsets| = \frac{b_1(b_1-1)}{2} + b_1 b_2$.

3.2.2. Uniform crossover

In this crossover operator, each cell of the first new solution is randomly selected either from its father or from its mother corresponding cell. For generation of the second new solutions, we use the first new solution but with exchanging the role of parents.

4. COMPUTATIONAL EXPERIMENTS

The MA and SS algorithms have been coded in C using Visual Studio 2010 while all the experiments were run on a computer with Intel (R) Core ™2 Duo, 2.93 GHz processor, 3.21 GB of internal memory and a 32-bit operating system, equipped with Windows XP. CPLEX 12.3 was used for solving the developed integer linear programming model. As the test problem, we consider a real test problem including the set of courses of B.Sc. and M.Sc. programs of Industrial Engineering at Ferdowsi University of Mashhad. In our test problem, there are 36 courses including 6 courses of M.Sc. program and 30 courses of B.Sc. program. Totally, 25 of undergraduate courses belong to the set C^3 while the remaining 11 courses belong to the set C^2. Moreover, all courses are grouped into four families and there is no course to be pre-assigned to specified timeslots. There are 18 professors to present courses including 8 faculty members and 10 adjunct professors.

In addition of real test problem, we generated a set of 21 random instances to evaluate the efficiency

of three different solution approaches.

4.1. Setting of parameters

4.1.1. Test problems

In this section we explain the way of setting parameters involved in the design of the test problems, i.e. pf_{it}, $oc_{ii'}$ and w_k. To obtain values corresponding to pf_{it}, a simple questionnaire was designed and random sample of students and also all faculty members are asked to determine ideal values for pf_{it}.

We divide all courses into four groups in terms of profitability in different time slots. For example, the first group includes harder courses which need more concentration for both students and professors. Thus, it is unfavorable to assign this set of courses to time slots such as noon time that either students or professors are tired. The profit values of four groups of courses in different time slots are shown in Table 3 in which group 4 is related to the courses of the M.Sc. program.

Table 3: Profit values of courses in different time slots

Group \ Time slot	8-10	10-12	12-14	14-16	16-18	18-20
1	9	10	4	5	7	6
2	7	8	3	4	6	5
3	5	5	2	5	5	3
4	5	5	2	3	6	6

In order to calculate overlapping cost for each two different courses i and i', we first need to know the earliest and the last semesters in which a courses i can be registered by a student, shown by es_i and ls_i, respectively. These two values are determined based on the number of predecessors and successors of each course in the corresponding program. Letting $cs_{ii'}$ to represent the number of common semesters of two intervals $[es_i, ls_i]$ and $[es_{i'}, ls_{i'}]$, then overlapping cost $oc_{ii'}$ is calculated as $oc_{ii'} = cs_{ii'}/\big((ls_i - es_i + 1) + (ls_{i'} - es_{i'} + 1)\big)$.

In order to determine four weights w_1 to w_4, we used the well-known Analytical Hierarchy Process (AHP) method (Triantaphyllou, 2000). For this purpose, we first considered a 4×4 matrix of pair-wise comparison in which cell k and k' show the relative preference of objective functions k and k', respectively. For objective functions f_1 to f_4, we considered pair-wise comparison matrix, shown in Table 4. Then, using the AHP, the weights w_1 to w_4 are extracted as follows: $w_1 = 0.49$, $w_2 = 0.1$, $w_3 = 0.25$ and $w_4 = 0.16$.

Table 4: Pair-wise comparison of different objective functions

	f_1	f_2	f_3	f_4
f_1	1	5	2	3
f_2	$\frac{1}{5}$	1	$\frac{2}{5}$	$\frac{3}{5}$
f_3	$\frac{1}{2}$	$\frac{5}{2}$	1	$\frac{3}{2}$
f_4	$\frac{1}{3}$	$\frac{5}{3}$	$\frac{2}{3}$	1

In addition of the real test problem, we generated 21 other theoretical test problems which differ in the number of time slots that professors are available. We define the stretchability degree (*SD*) of a test problem as the summation of available time slots for all professors divided by the maximum number of available time slots (60**m*). We considered three values *SD*=1, 0.75 and 0.5 and generated 10 test instances corresponding to each of values *SD*= 0.75 and 0.5 in which available time slots of professors are assigned to set of time slots randomly such that a feasible timetable could be obtained. Since *SD*=1 means that all professors are available in all time slots, only one test problem could be considered for this case. Also, it should be noticed that for the real test problem we have *SD*= 0.96.

It should be mentioned that four parameters w_1 to w_4 are determined randomly for theoretical test instances such that $0 \leq w_i \leq 1; i = 1, 2, 3, 4$ and $\sum_{i=1}^{4} w_i = 1$. The values of weights for theoretical test problems are reported in Table 5.

Table 5: The weight values of theoretical test problems

SD	Test problem#	w_1	w_2	w_3	w_4
1	1	0.63	0.08	0.22	0.07
0.75	1	0.20	0.45	0.30	0.05
	2	0.21	0.10	0.24	0.45
	3	0.34	0.27	0.10	0.29
	4	0.21	0.13	0.50	0.16
	5	0.13	0.25	0.26	0.36
	6	0.57	0.16	0.15	0.12
	7	0.16	0.14	0.65	0.05
	8	0.08	0.48	0.30	0.14
	9	0.12	0.43	0.15	0.30
	10	0.33	0.12	0.27	0.28
0.5	1	0.42	0.37	0.07	0.14
	2	0.30	0.10	0.41	0.19
	3	0.24	0.18	0.33	0.25
	4	0.30	0.30	0.18	0.22
	5	0.06	0.29	0.22	0.43
	6	0.29	0.24	0.25	0.22
	7	0.11	0.12	0.66	0.11
	8	0.15	0.23	0.25	0.37
	9	0.20	0.43	0.17	0.20
	10	0.14	0.07	0.40	0.39

4.1.2.Setting of the MA parameters

In order to improve the performance of the MA, we set the values of parameters *psize*, p_{mut} and p_{ls} using a design of experiments. We considered three values for each parameter based on the given time limits (*TL*), i.e. *TL*=10,30,60,180,300,600,900,1800 and 3600, but the best values of parameters, reported in Table 6, are strictly dependent to the given time limits.

Table 6: Setting of the MA parameters

TL(Seconds)	*psize*	p_{ls}	p_{mut}
10	50	0.05	0.01
30, 60, 180	75	0.05	0.01
300, 600, 900, 1800, 3600	500	0.05	0.01

4.1.3.Setting of the SS parameters

Similar to the previous section, in this section the obtained values for parameters b_1 and b_2 are reported in Table 7.

Table 7: Setting of the SS parameters for the real test problem

TL(Seconds)	b_1	b_2	p_{ls}	th_dist_1	th_dist_2
10	3	7	0.05	0.1	0.6
30	5	10	0.05	0.1	0.6
60, 180, 300, 600, 900, 1800, 3600	10	20	0.05	0.1	0.6

4.2. Comparative Results with time limits

4.2.1. Comparative results for the real test problem

Based upon the results acquired from the implementation of the linear integer model using the CPLEX over the data of the real test problem, we obtained the values of f_k^+ as $f_1^+ = 736, f_2^+ = 3.54, f_3^+ = 51$ and $f_4^+ = 27$. Also, we obtained vales of f_k^- using a long run of MS and SS algorithm as $f_1^- = 413, f_2^- = 257.63, f_3^- = 95$ and $f_4^- = 49$. Figure 5 shows the objective function's improvement of different solution methods over the time. At the beginning, CPLEX has the worst objective cost but it has been improved quickly. Although CPLEX cannot find a better solution than SS during 3600 seconds of CPU time, but it surpasses the MA after 600 seconds of CPU time. Furthermore, we conclude that SS performs better than MA in all time limits and also, the best solution has been obtained by the SS.

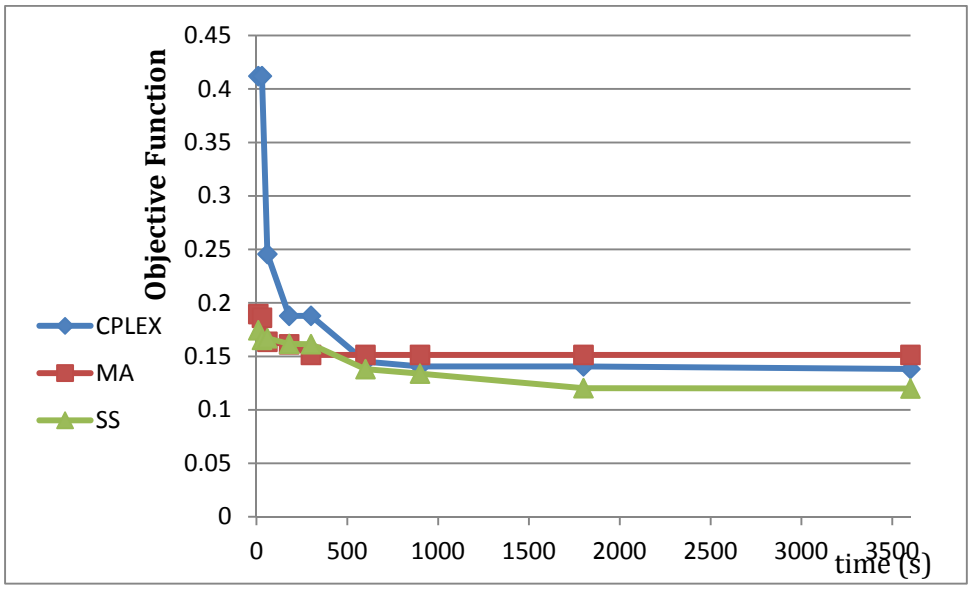

Figure 5: Comparative results for the real test problem

4.2.2. Comparative results for the theoretical test problems

Similar to the previous section, we first calculate the values of f_k^+ and f_k^- for all $k = 1,2,3$ and 4 which are reported in Table 8.

Table 8: The values of f_i^+ and f_i^-

SD	Test problem#								
100	1	736	395	1.067	267.62	51	97	27	49
0.75	1	732	397	1.067	250.27	51	97	27	49
	2	734	395	1.067	246.86	51	97	27	49
	3	736	399	1.067	251.08	51	96	27	49
	4	718	397	1.067	265.08	56	94	28	49
	5	731	397	1.067	257.97	51	97	27	49
	6	716	397	1.067	246.55	55	91	27	49
	7	728	405	3.067	240.37	56	91	28	49
	8	734	395	1.067	251.11	52	97	27	49
	9	735	395	1.067	258.86	51	96	27	49
	10	717	411	3.428	234.963	52	81	27	49
0.5	1	723	401	1.067	252.64	52	97	27	49
	2	729	401	1.067	229.49	56	97	29	49
	3	717	403	1.142	236.42	59	91	28	49
	4	717	395	1.067	246.18	55	97	27	49
	5	705	395	1.067	246.48	54	94	27	49
	6	720	395	1.067	232.49	52	94	27	49
	7	725	395	1.067	241.29	52	96	27	49
	8	729	395	1.067	242.11	51	97	27	49
	9	717	411	3.428	213.523	52	82	27	49
	10	729	395	1.067	235.10	52	94	27	49

Figures 6, 7 and 8 indicate over different time limits, how the objective function is being improved

by different solutions approaches. In Figure 6, the improvement trends for all three solutions algorithms are almost similar to Figure 5 but this time, the CPLEX surpasses the MA after 900 seconds. Also, the best solution has been found by the SS.

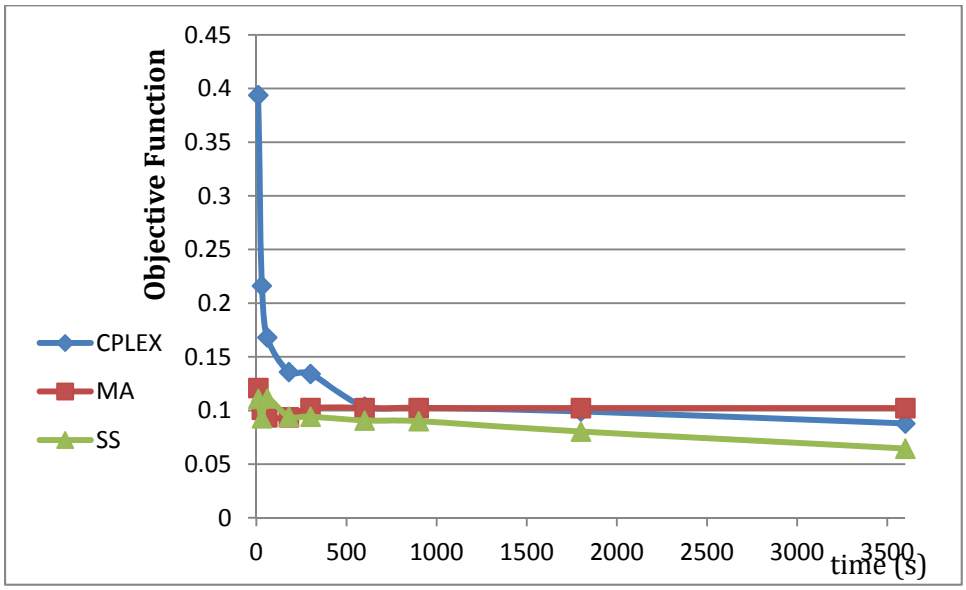

Figure 6: Comparative results for the test problem with *SD*=1

Since for *SD*=0.75 and 0.5, we have 10 test problems, the average of objective functions over 10 test instances has been reported in Figures 7 and 8, respectively.

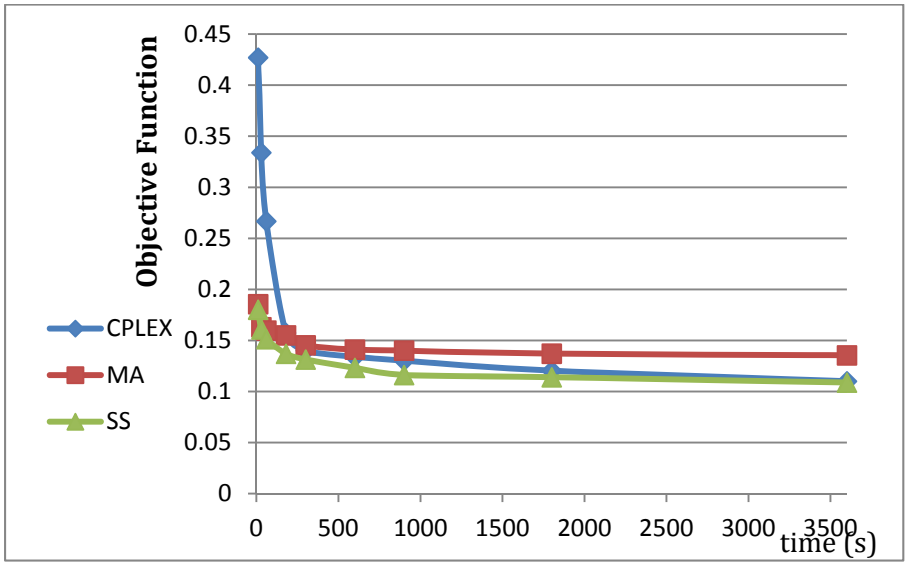

Figure 7: Comparative results for the test problems with *SD*=0.75

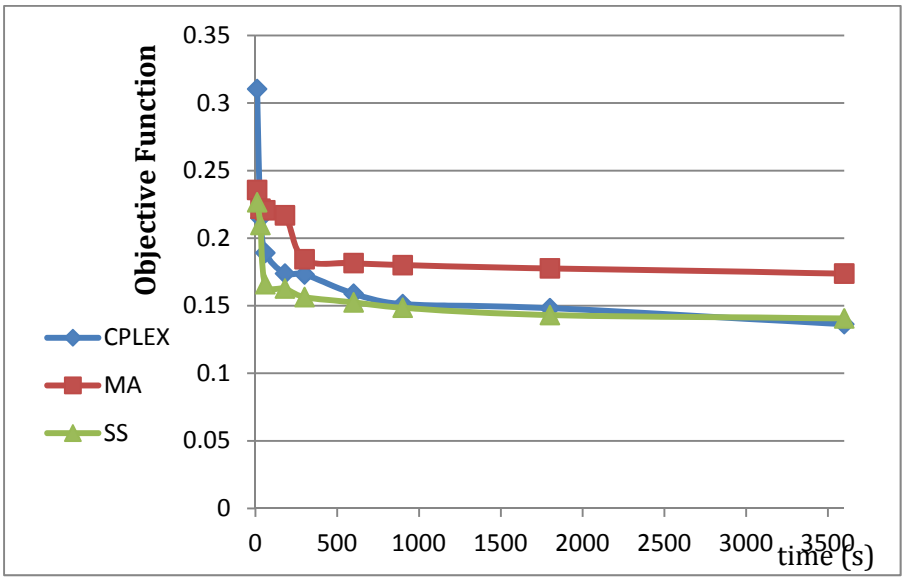

Figure 8: Comparative results for the test problems with *SD*=0.5

Figure 7 indicates that the SS algorithm has the best performance among three proposed solution approaches but Figure 8 reveals that CPLEX has surpassed the MA and SS algorithms after end of maximum time limit. By comparing the trend of improvements in three previous figures, we conclude that for harder test instance (higher value of *SD*) and for shorter time limits, the MA and SS have better performance that CPLEX but when the *SD* is decreased or the given time limit is increased, CPLEX is able to find better solutions that MA and SS.

Tables 9 and 10 report the attained values for different objective functions over test problems with *SD*=0.75 by the MA and SS, respectively. These results have been obtained by the time limits *TL*=300 seconds.

Table 9: Objective functions values obtained by MA over test problems with *SD*=0.75 in 300 seconds

f_1	f_2	f_3	f_4	f
634	12.30	54	40	0.13
650	16.15	60	29	0.15
686	46.71	65	29	0.16
640	43.62	56	35	0.13
604	40.46	55	30	0.16
682	51.68	61	36	0.17
622	45.53	57	34	0.11
612	12.37	56	35	0.13
587	21.66	58	29	0.14
661	53.21	55	33	0.20

Table 10 : objective functions values obtained by SS over test problems with *SD*=0.75 in 300 seconds

f_1	f_2	f_3	f_4	f
635	11.25	54	40	0.13
658	39.64	61	28	0.14
684	32.22	65	30	0.16
664	45.20	57	39	0.15
621	28.99	56	27	0.10
686	75.86	56	40	0.19
638	30.22	56	34	0.07
615	8.40	57	32	0.11
591	10.30	63	27	0.11
636	50.99	54	30	0.17

4.2.3. Statistical comparisons

In order to have a more meticulous comparison of the three developed solution approaches, we have developed a statistical comparisons based on randomized complete block designs (Montgomery 2012). In particular, for each value of *SD*, we perform a statistical test in which different solution approaches and time limits are considered as treatments and blocks, respectively. For the real test problem and the randomly generated instances with *SD*=1, we have only one replication (one test problem) while for random instances with *SD*=0.5 and 0.75, totally 20 replications corresponding to 20 test problems, are available. We denote as *"rp"* the number of replications. In all of the four aforementioned statistical tests, the null hypothesis is $H_0: f^{\text{CPLEX}} = f^{\text{MA}} = f^{\text{SS}}$ and the alternate hypothesis is $H_1: Otherwise$ in which $f^{\text{CPLEX}}, f^{\text{MA}}$ and f^{SS} indicate the average performance of CPLEX, MA and SS over the corresponding instances, respectively. Table 11 reports the analysis of variance (ANOVA) in which SS_T, SS_{Tr}, SS_b and SS_E denote, respectively, the total corrected sum of squares, sum of squares of the differences between the treatments averages and the grand average, sum of squares of the difference between the blocks average and the grand average and sum of squares errors, obtained by subtraction. In other words, we have $SS_T = SS_{Tr} + SS_b + SS_E$. Also, MS_{Tr}, SS_b and SS_E denote the mean squares of treatments, blocks and errors, respectively. The test statistic corresponding to H_0 and H_1 is $F_0 = \frac{MS_{Tr}}{MS_E}$ which has *F*-distribution.

The ANOVA corresponding to the real test problem is shown in Table 12 for which the

Table 11: ANOVA table

Source of deviation	Sum of squares	Degree of freedom	Mean squares	F_0
Treatments (algorithms)	SS_{Tr}	3-1=2	$MS_{Tr} = \frac{SS_{Tr}}{2}$	$\frac{MS_{Tr}}{MS_E}$
Blocks (time limits)	SS_b	9-1=8	$MS_b = \frac{SS_b}{8}$	
Error	SS_E	27*rp*-11	$MS_E = \frac{SS_E}{27rp - 11}$	
Total	SS_T	27*rp*-1		

P-value is around 0.03. Also, if we use the paired *t*-tests for pair comparisons of the three solution approaches, for statistical tests $H_0: f^{CPLEX} = f^{MA}$, $H_0: f^{CPLEX} = f^{SS}$ and $H_0: f^{SS} = f^{MA}$ the corresponding *P-values* are 0.048, 0.026 and 0.008, respectively. These results indicate that for the

real test problem, the difference between algorithms performance is not significant especially for two algorithms MA and SS.

Table 12: ANOVA corresponding to the real test problem

Source of deviation	Sum of squares	Degree of freedom	Mean squares	F_0
Treatments (algorithms)	0.028	3-1=2	0.014	4.39
Blocks (time limits)	0.056	9-1=8	0.007	
Error	0.051	27-11=16	0.003	
Total	0.135	27-1=26		

Similar to the real test problem, we develop Table 12 for the ANOVA of the theoretical test with *SD*=1 in which the *P-value* is again around 0.03. Also, If we use the paired *t*-tests for pair comparisons of the three solution approaches, for statistical tests $H_0: f^{CPLEX} = f^{MA}$, $H_0: f^{CPLEX} = f^{SS}$ and $H_0: f^{SS} = f^{MA}$ the corresponding *P-values* are 0.047, 0.023 and 0.04, respectively.

Table 13: ANOVA corresponding to the test problem with *SD*=1

Source of deviation	Sum of squares	Degree of freedom	Mean squares	F_0
Treatments (algorithms)	0.024	3-1=2	0.012	4.49
Blocks (time limits)	0.033	9-1=8	0.004	
Error	0.043	27-11=16	0.0026	
Total	0.1	27-1=26		

Tables 14 and 15 represent the ANOVA for test instances with *SD*=0.75 and 0.5, respectively. Since number of replications is 10, the degrees of freedoms are changed. Table 14 indicates that the performance of the three developed algorithms are significantly different such that the *P-value* of this statistical test is approximately zero. Thus, we establish three paired *t*-tests to compare more exactly the solution approaches. For test problems with *SD*=0.75, the *P-values* of statistical tests $H_0: f^{CPLEX} = f^{MA}$, $H_0: f^{CPLEX} = f^{SS}$ and $H_0: f^{SS} = f^{MA}$ are all approximately zero which indicates they have significantly different performance.

Table 14: ANOVA corresponding to the test problem with *SD*=0.75

Source of deviation	Sum of squares	Degree of freedom	Mean squares	F_0
Treatments (algorithms)	0.217	3-1=2	0.108	30
Blocks (time limits)	0.621	9-1=8	0.078	
Error	0.93	259	0.0036	
Total	1.768	269		

Analysis of the results of Table 15 is exactly similar to our analysis over the results of Table 13 and we conclude exactly the same *P-values* and conclusions.

Finally, we conclude that performance of the three solutions approaches (i.e. CPLEX, MA and SS) are not significantly different for easier test problems (test problems with higher value of *SD*) but they show different performance on harder instances in which SS has the best performance and the

CPLEX has the worst performance.

Table 14: ANOVA corresponding to the test problem with *SD*=0.5

Source of deviation	Sum of Squares	Degree of Freedom	Mean Squares	F_0
Treatments (algorithms)	0.045	3-1=2	0.0225	14.06
Blocks (Time limits)	0.287	9-1=8	0.036	
Error	0.427	259	0.0016	
Total	0.759	269		

5. SUMMARY AND CONCLUSIONS

In this paper, we considered a case study of the fortnightly university course timetabling problem. We developed an integer linear programming model in which four different objective functions are combined into a single objective using the L_P-metric method. We solved the developed model using CPLEX 12.3. Moreover, we developed a MA and a SS to solve the problem. For harder test problems and in shorter time limits, the metaheuristic algorithms have better performance than the CPLEX while SS outperforms always the MA. We showed this conclusion by the statistical tests as well. By decreasing the stretchability degree or complexity of test problems and in larger time limits, the CPLEX is able to surpass the developed metaheuristic algorithms. Since exact methods are unable to solve such a hard problem, developing heuristic or other metaheuristic solution techniques can be interesting research topics.

REFERENCES:

[1] Alkan A, Ozcan E (2003), Memetic algorithms for timetabling; *Proc. of IEEE Congress on Evolutionary Computation*; 1796–1802.

[2] Bagchi T P (1999), Multi-objective scheduling by genetic algorithms; *Kluwer Academic Publishers*.

[3] Bardadym V A (1996), Computer-aided school and university timetabling: The new wave. In E. Burke & P. Ross (Eds.), Practice and theory of automated timetabling; *Lecture notes in computer science* 1153; 22–45, *Berlin: Springer*.

[4] Belton V, Stewart T J (2002), Multiple criteria decision analysis - An Integrated Approach; *Kluwer Academic Publishers*.

[5] Boland N, Hughes B D, Merlot L T G, Stuckey P J (2008), New integer linear programming approaches for course timetabling; *Computers & Operations Research* 35(7); 2209-2233.

[6] Burke E K, Eckersley A J, McCollum B, Petrovic S, Qu R (2010), Hybrid variable neighborhood approaches to university exam timetabling; *European Journal of Operational Research* 206(1); 46-53.

[7] Brucker P, Drexl A, Mohring R, Neumann K, Pesch E (1999), Resource-constrained project scheduling: notation, classification, models and, methods; *European Journal of Operational Research* 112; 3-41.

[8] Burke E K, McCollum B, Meisels A, Petrovic S, Qu R (2007), A graph-based hyper heuristic for timetabling problems; *European Journal of Operational Research* 176(1); 177–192.

[9] Ceschia S, Di Gaspero L, Schaef A (2012), Design, engineering, and experimental analysis of a simulated annealing approach to the post-enrolment course timetabling problem; *Computers & Operations Research* 39(7); 1615-1624.

[10] Daskalaki S, Birbas T (2005), Efficient solutions for a university timetabling problem through integer programming; *European Journal of Operational Research* 160(1); 106-120.

[11] Daskalaki S, Birbas T, Housos E (2004), An integer programming formulation for a case study in university timetabling; *European Journal of Operational Research* 153(1); 117-135.

[12] Ehrgott M, Gandibleux X (2000), A survey and annotated bibliography of multi-objective combinatorial optimization; *OR Spectrum* 22(4); 425-460.

[13] Glover F (1977), Heuristics for integer programming using surrogate constraints; *Decision Sciences* 8; 156-166.

[14] Holland J H (1975), Adaptation in natural and artificial systems; *Univ. Mich. Press.*

[15] Ishibuchi H, Yoshida T, Murata T (2002), Selection of initial solutions for local search in multi-objective genetic local search*; Proceedings of the 2002 Congress on Evolutionary Computation* (CEC 2002), *IEEE Press*; 950-955.

[16] Jones D F, Mirrazavi S K, Tamiz M (2001), Multi-objective meta-heuristics: an overview of the current state-of-the-art; *European Journal of Operational Research* 137(1); 1-9.

[17] Krasnogor N (2002), Studies on the theory and design space of memetic algorithms. *Ph.D. Thesis; University of the West of England, Bristol, United Kingdom*.

[18] Laguna M, Martí R (2003), Scatter search—methodology and implementations in C; *Kluwer Academic Publishers, Boston*.

[19] Lewis R (2008), A survey of metaheuristic-based techniques for university timetabling problems. *OR Spectrum* 30 (1); 167–190.

[20] Lü Z, Hao J K (2010), Adaptive Tabu search for course timetabling; European Journal of Operational *Research* 200(1); 235-244.

[21] Mirhassani S A (2006), A computational approach to enhancing course timetabling with integer programming; *Applied Mathematics and Computation* 175(1); 814-822.

[22] Mitchell M (1998), An introduction to genetic algorithms (Complex Adaptive Systems)*; A Bradford Book*.

[23] Montgomery D C (2012), Design and analysis of experiments; *Springer*.

[24] Neri F, Cotta C, Moscato P (2011), *Handbook of memetic algorithms; Springer*.

[25] Paechter B, Rankin R C, Cumming A, Fogarty T C (1998), Timetabling the classes of an entire university with an evolutionary algorithm; *Proc. of Parallel Problem Solving from Nature* (PPSN V); 865– 874.

[26] Pinedo M (2008), Scheduling, theory, algorithms, and systems, 3nd Edition; *Prentice-Hall*.

[27] Rosenthal R E (1985), Principles of multi-objective optimization; *Decision Sciences* 16; 133-152.

[28] Shiau D F (2011), A hybrid particle swarm optimization for a university course scheduling problem with flexible preferences; *Expert Systems with Applications* 38(1); 235-248.

[29] Tan K C, Lee T H, Khor E F (2002), Evolutionary algorithms for multi-objective optimization: performance assessments and comparisons; *Artificial Intelligence Review* 17; 253-290.

[30] Triantaphyllou E (2000), Multi-criteria decision making methods: A comparative Study; *Springer*.

[31] T'kindt V, Billaut J C (2002), Multicriteria scheduling: Theory, Models and Algorithms; *Springer*.

[32] Wang Y Z (2003), Using genetic algorithm methods to solve course scheduling problems; *Expert Systems with Applications* 25(1); 39–50.