Modification of Scatter Search for Examination Timetabling Problem

Majid Salari, Zahra Naji Azimi
Department of Mathematical Sciences
Ferdowsi University of Mashhad
salarimajid,najiazimi@yahoo.com

Abstract:

The construction of an exam timetable is a common problem for all universities and institutions of higher education. Quite often it is done by hand or with the limited help of a simple administration system and usually involves taking the previous year's timetable and modifying it so it will work for the new year. But increasing the number of students, changing the courses that are offered and student's freedom in selecting them, needs a great correction on the past year's timetable and also so much time. Therefore it is no longer good enough to use the previous year's timetable.

In this approach we modify Scatter Search method and solve the Examination Timetabling Problem (ETP) with a first solution procedure that is based on scatter search. Also we solve this problem with existing methods such as Simulated Annealing and Tabu search and compare results of them with each other. Finally we apply new Scatter Search algorithm on Carter's datasets and conclude that our algorithm works better than other published results.

Keywords: *Timetabling, Scatter Search, Simulated Annealing, Tabu search*

1. Introduction:

The Examination Timetabling problem regards the scheduling for the exams of a set of university courses, avoiding overlap of exams of courses having common students, and spreading the exams for the students as much as possible.

The process of finding a period for each exam so that no two conflict has been shown to be equivalent to assigning colours to vertices in graph so that adjacent vertices always have different colours [1]. This in turn has been proved to lie in the set of NP_Complete problems [2,3,4], which means that carrying out an exhaustive search for the timetable is not possible in a reasonable time.

There are so many heuristic methods which has been offered for solving this problem based on graph colouring or metaheuristic methods such as Simulated Annealing, Tabu Search, ect, that has been applied to solve the problem [5-18].

We approach this problem with a solution procedure based on the evolutionary approach called Scatter Search (SS). Recent studies have demonstrated the practical advantages of this approach for solving a diverse array of optimization problems from both classical and real world settings. A good overview of SS is provided by Laguna, Glover, and Marti [19,20].

To continue we explain Examination Timetabling problem and modification of scatter search then we try to describe how to solve this problem by using modification of scatter search algorithm. Finally we'll distinguish the results, which have been concluded of this algorithm in compare of another metaheuristics.

2. Common details in all methods

In this section we explain the common details that are used in all methods.

2.1. Problem Description

Given are a set of examinations, a set of (contiguous) time slots, a set of students and a set of student enrolments to examinations. The problem is to assign examinations to time slots satisfying a set of constraints [14].

Many different constraint types have been proposed in the literature. In this work, we consider the version proposed by Carter et al. [6], which is based on the so–called *first-order* and *second-order* conflicts.

First-order conflicts arise when a student has to take two exams scheduled in the same time slot, while second order ones emerge when a student has to take two exams in time slot "close" to each other.

Second–order conflicts are treated as *soft* constraints, thus they are included into the objective function *f* that measure the quality of the solution; conversely, first–order conflicts are modelled as *hard* constraints.

2.2. The Objective Function

Assuming that consecutive time slots lie one unit apart, we define *f* assigning a proximity cost *w(i)* when ever a student has to attend two exams scheduled within *i* time slots. The cost of each conflict is thus multiplied by the number of students involved in both examinations.

As it is mentioned in [14], the cost decrease logarithmically from 16 to 1 for soft constraints as follows: $w(1)=16$, $w(2)=8$, $w(3)=4$, $w(4)=2$, $w(5)=1$ and the cost for violation of hard constraint is 1000.

The objective function (cost function) is then normalized based on the total number of students. This way we obtain a measure of the number of violations "per student", which allows us to compare results for instances of different size. So we have the below cost function,

$$F=(f_1+f_2)/M$$

$$f_1=\sum_{i=1}^{N-1}\sum_{j=i+1}^{N}C_{ij}\cdot w(i,j) \qquad where \qquad w(i,j)=\begin{cases}2^{5-|t_i-t_j|} & if\ 1\le |t_i-t_j|\le 5\\ 0 & otherwise\end{cases}$$

$$f_2=\sum_{i=1}^{N-1}\sum_{j=i+1}^{N}C_{ij}\cdot w'(i,j) \qquad where \qquad w'(i,j)=\begin{cases}1000 & if\ t_i=t_j\\ 0 & otherwise\end{cases}$$

Where N and M indicate the number of exams and students consecutively and $C(i,j)$ shows the number of common students between both exam i and exam j, also t_i is the period of exam i (for $i=1,\dots,N$).

There are other constraints like saloon capacity in the real case that we don't consider them in order to make it easier and because we can compare our results with result of published papers on Carter's data sets [6,11,14].

2.3. Solution representation

We show every solution in all algorithms with one vector, with the length of the vector equal to the number of exams. Each elements of this vector shows the assigned period for each exam. Periods and exams are numbered sequentially.

1	2	3	4				N-2	N-1	N
20	3	18	5	…	…	…	7	11	6

2.4. Neighbour solution

A neighbour solution may not be a feasible solution necessarily (as initial solutions), and it is obtained by random alteration of one element of the solution vector. We also considered the neighbour, which is obtained from exchange of two exams of solution with each other. We didn’t get any significant difference between final solutions with two definitions of neighbourhood.

3. Modification of Scatter Search and Solution Approach:

The solution approach that we have developed for our Examination Timetabling problem consists of adaptation of Scatter Search.

Scatter Search (SS) is a novel instance of evolutionary methods and embodies principles and strategies that are still not emulated by other evolutionary methods, that proves advantageous for solving a variety of complex optimization problems [19,20].

In this algorithm at first *Diversification Generation Method* constructs a set of solutions, then these solutions are improved by *Improvement Method*. In the next step *Reference Set* is constructed based on improved and also diverse solutions. Then in a loop, solutions are selected for combination by *Subset Generation Method* and generate the new solutions by *Solution Combination Method*. The new solutions are improved by same *Improvement method* and this loop continues until we can’t obtain new solution. In this step, the algorithm reinitialized the initial set by generate the *New Set* and run this loop for predefined times.

3.1.What are our modifications for SS?

In the general case of Scatter Search, at first a set of solutions is constructed completely and when this process is finished then in a next step these solutions are improved by a heuristic method. In fact these two steps are completely independent.

But it seems that if we can use the information of best current solution in process of constructing the next one, we will get a better solution. That ‘s the first and main modification of SS in our algorithm.

In fact in our modification, at first a solution is constructed and improved consecutively. Then the next solution produced based on the information of the best solution, which has been obtained through previous steps. This new solution is improved and the best current solution is updated. So the construction and improvement method are applied sequent for each solution and we use the information of improvement step in construction of next solution. This process is repeated for |initial set| (cardinal of initial set) times.

Also in the general case of SS, there is one Improvement Method that is applied two times: after construction of initial solutions and after generation of new solutions from Solution Combination Method.

But as a second modification, we use two different heuristic methods for these steps. As we said before we apply Improvement Method after construction of each solution and we use the information of the best solution to generate the next one. So if we use the heuristic method, which needs a long time for execution, it will increase the amount of time for complete run of this step and it is no good for us.

Besides, if we use the strong heuristic when we generate the new solutions, we can reach the better solutions. Therefore it seems that's better to use two different heuristics for these two steps: improvement after construction of each initial solutions and improvement after generation of new solutions by Solution Combination Method.

So we use the simple descent algorithm for improving the initial solution and Tabu Search method is applied to improve the new solutions, which are obtained by Solution Combination Method.

Also in general case of SS there is one method (which is named Diversification Generation Method) to construct solutions in two steps: generation of initial solutions and to reinitialize of algorithm with New Set of solutions when we can't reach any new solution in internal loop.

Since our generation method for initial solutions depends to information of best solution and it will get the new solution very similar to this one, we can't reach enough diversification in new set by applying this method. So it is better to use another generation method to give diverse solutions in this step, which is need for climbing up from local optimum point. In Second Generation Method we apply frequency memory and the probability of selecting each period is inverse of frequency of this and neighbour periods in solution.

To continue we describe several parts of this algorithm and explain how to apply these three modifications.

Because we apply diversification generation and improvement methods consecutively for each initial solution and because of affections of these steps on each other, we explain these two steps in one section with name Mixed Diversification and First Improvement Method.

3.2. Mixed Diversification and First Improvement Method:

In our modification of SS, The construction process of initial solutions is applied in an artificial manner, so that in selection of period for each exam, increased amount of cost is considered and a period will selected that causes minimum increasement.

Also the information of the best solution is considered and finally a random period (timeslot) is selected according to the two factors that are mentioned.

To generate the initial solutions we do as follows:

At first we assign the same initial values for each pair (exam,timeslot) and we call it “credit “. Suppose that there are *k* solutions, which have been constructed, and now we want to produce *k+1* Th solution.

In the beginning, the best solution is considered and then the credit of each pair (exam,timeslot) is updated. Credits of the pairs that have been appeared in the best solution, is increased and the credit of others is decreased by this formula:

$$credit(e,t) = \begin{cases} 0.2.credit(e,t)+1 & if\ (e,t)\ is\ in\ the\ best\ solution \\ 0.2.credit(e,t) & otherwise \end{cases} \qquad (3.1.1)$$

Now we consider the exams one by one and assign to each of them one period (timeslot). The selection probability of each period is based on the increasing amount of cost and credit value of pairs as a followed formula:

$$p_{(e,t)}(credit(e,t), c(e,t)) = \frac{(credit(e,t))^{\alpha}.(c(e,t))^{\beta}}{\sum\limits_{\theta \in T}(credit(e,\theta))^{\alpha}.(c(e,\theta))^{\beta}} \qquad (3.1.2)$$

That *c(e,t)* is inverse of cost and achieved from:

$$c(e,t) = \frac{1.0}{1.0 + V(e,t)} \qquad (3.1.3)$$

And *V(e,t)* shows the increased amount of cost by adding *(e,t)* .*T* is a set of timeslots and the coefficients α, β indicate the effect of each factor.

For selecting a timeslot, at first a random number of [0,1] is selected and if this number is equal to or less than selection probability of period, (that has been reached from formula.3.1.2) this period is assigned to the exam and else another random timeslot is selected. This process continues until a period is assigned to this exam. When a complete solution is constructed, First Improvement Method improves it and the best solution is updated. If this improved solution is better than the best current solution, this solution is considered as a best solution instead of it.

- ***The First Improvement Method***

First Improvement method is applied after construction of each initial solution. For this aim one exam is selected randomly and its period exchange to another random period and then the cost of new solution is computed.

If new solution has less cost, it is considered instead of previous one and else this process is going to be continuing until the first better solution is reached. Then the best solution, which is considered in construction process, is updated and the credits are changed regards to it (as it is explained above).

3.3. Reference Set Update Method:

The reference set, *Refset*, is a collection of both high quality and diverse solutions that are used to generate new solutions by way of applying the solution combination method [19,20]. Specifically, The reference set consists of the union of two subsets, $Refset_1$ and $Refset_2$, of size b_1 and b_2, respectively. That is, $|Refset| = b = b_1+b_2$. The construction of the initial reference set starts with the selection of the best b_1 solutions from the set of initial solutions, which is named ***P*.** These solutions are added to *Refset* and deleted from *P*. For each improved solution in *P-Refset*, the minimum of Euclidian distances to the solutions in *Refset* is computed. Then the solution with the maximum of these minimum distances is selected. This solution is added to *Refset* and deleted from *P* and the minimum distances are updated. This process is repeated b_2 times. The resulting reference set has b_1 high quality solutions and b_2 diverse solutions.

To explain the Euclidian distance, for example we consider one solution as a current solution and another as a solution of *Refset*. The Euclidian distance between these two solutions are computed as follows:

X = current solution

20	8	13	…	…	…	7	32

Y = solution of Refset

12	30	10	…	…	…	7	16

D(X,Y) = $\sqrt{(20-12)^2+(8-30)^2+(13-10)^2+...+(7-7)^2+(32-16)^2}$

After the initial reference set is constructed, the Solution Combination Method is applied to the subsets generated as outlined in the following section. The reference set is dynamically updated during the application of the Solution Combination Method. A newly generated solution may become a number of the reference set if either one of the following conditions is satisfied.

1. The new solution has a better objective function value than the solution with the worst objective value in *Refset*$_1$.
2. The new solution has a better d_{min} value than the solution with the worst d_{min} value in *Refset*$_2$.

In both cases, the new solution replaces the worst and the ranking is updated to identify the new worst solution in terms of either quality or diversity.

The reference set is also generated when a combination method is incapable of creating solutions that can be admitted to Refset according to the rules outlined above.

The regeneration consist of keeping Refset$_1$ intact and using the Second Generation Method to construct a new diverse subset Refset$_2$. As we mentioned before, in general case of SS in this step the same Diversification Generation Method is used but in our modification, we use another method because of reasoning that are mentioned in section 3.1. We will express this method in section 3.7.

3.4. Subset Generation Method:

This method consists of generating the subsets that will be used for creating new solutions with the Solution Combination Method [19,20].

We limit our scope to considering five solutions. Four solutions of high quality and one of diverse solutions of Refset are selected. Then we consider all pair wise of them.

3.5. Solution Combination Method:

This method consists of generating new solutions from the combination of two existing solutions.

When we select two solutions to combine, at first we consider all periods one by one and then choose one of two solutions randomly. Combined solution will consist of all the exams that have been appeared in this period of this selected solution. Then another period is considered and this process continues until all periods are considered.

3.6. The Second Improvement method:

We use Tabu Search algorithm in this part on the new solutions, which are given from Solution Combination Method (see section 3.5) and it is terminated before its convergence. Stopping condition of TS is a little limited time and the best improving solution is considered as an improved solution. This part is used in internal loop of Scatter Search algorithm. The details of this algorithm are as same as section 6.

3.7.The second Generation Method

When a combination method is incapable of creating solutions that can be admitted to Refset according to the rules outlined in section 3.3, the Refset is regenerated. The regeneration consists of keeping Refset$_1$ intact and constructs a new diverse subset Refset$_2$. In general case of SS in this step the same Diversification Generation Method is used for generation of Refset$_2$ but in our modification, we use another method to achieve diverse solutions because of reasoning that are mentioned in section 3.1.

This generation method employs controlled randomisation and frequency memory to generate a set of diverse solutions. We divide the range of N periods into [N/5] sub_ranges (this is may differ for instances with different size). Then a solution is constructed in two steps. First a sub_range is randomly selected. The probability of selecting a sub_range is inversely proportional to its frequency count. Then a period is randomly generated within the selected sub_range. The number of times sub_range j has been chosen to assign a period for exam i is accumulated in $freq(i,j)$. This diversification generation method focuses on diversification and not on the quality of solutions. Note that the best b_1 solutions are entered in *Refset* intact and because of this we don’t consider the quality of new solutions and we just consider diversification.

4. An outline of the procedure:

This outline is a general algorithm of SS that is mentioned in [19,20] but with our modifications and uses the following parameters:

PSize = the size of the set of initial solutions, which are generated by the Mixed Diversification and First Improvement Method.

b= the size of the *Reference set* .

b_1= the size of the high quality subset (*Refset*$_1$).

b_2= the size of the diverse subset (*Refset*$_2$).

MaxIter = maximum number of iterations.

The procedure consist of the steps in the outline of Table.1, where *P* denotes the set of solutions generated with the Mixed Diversification and First Improvement Method and *Refset* is the set of solutions in the reference set. Also *New Set* is the set of new solutions that are constructed with Second Generation Method.

Table 1. Modification of Scatter Search for Examination timetabling problem

```
1. Start with $P = \phi$
   Use the Mixed Diversification and First Improvement method to construct
   the initial solution $x$.
   Repeat this step Until $| P | = PSize$.
2. Order the solutions in $P$ according to their objective function value (where
   the best overall solution is first on the list).
For ($Iter$=1 to $MaxIter$)
     3. Build $RefSet=RefSet_1 \cup RefSet_2$ from $P$, with $| RefSet | = b$,
        $| RefSet_1| = b_1$ and $| Refset_2 | = b_2$. Take the first $b_1$ solutions in $P$ and
        add them to $RefSet_1$. For each solution $x$ in $P$-$RefSet$ and $y$ in $RefSet$
        calculate a measure of distance $d(x,y)$. Select the
        solution $x'$ that maximizes $d_{min}(x)$, where $d_{\min}(x) = \min_{y \in RefSet} \{d(x,y)\}$
        add $x'$ to $RefSet_2$, until $| RefSet_2 | = b_2$. Make $NewElements$=TRUE
     While ($NewElements$) do
       4. Calculate the number of subsets ($MaxSubset$) that include at least one
          new element. Make $NewElements$ = FALSE.
       For ($SubsetCounter$ = 1, ... , $Maxsubset$) do
          5. Generate the next subset "$s$" from $RefSet$ with the Subset
                Generation Method.
          6. Apply the Solution Combination Method to" $s$" to obtain one new
             solutions $x_s$.
```

```
                7. Apply the Second Improvement Method (TS) to $x_s$ , to obtain the
                      improved solution $x_s^*$.
                If ( $x_s^*$ is not in *RefSet* and the objective function value of $x_s^*$ is
                better than the objective function value of the worst element in
                RefSet$_1$) **then**
                   8.  Add $x_s^*$ To RefSet$_1$ and delete the worst element
                       Currently in RefSet$_1$. (The worst element is the solution
                       With worst objective value.)
                   9. Make *NewElements* = TRUE.
               **Else**
                  **If** ( $x_s^*$ is not in *RefSet$_2$* and $d_{min}(x_s^*)$ is larger than $d_{min}(x)$
                       for a solution $x$ in *RefSet$_2$*) **then**
                     10 . Add $x_s^*$ to *RefSet$_2$* and delete the worst
                         element currently in *RefSet$_2$*. (The worst element is the
                       solution $x$ with the smallest $d_{min}(x)$ value.)
                     11. Make NewElements = TRUE.
                  **End If**
               **End If**
           **End For**
        **End While**
        **If** (*Iter* < *MaxIter*) **then**
            12. Build the *New Set* using the Second Generation method.
                (Initialize the generation process with the solutions currently in *RefSet$_1$*
                 That is, the first $b_1$ solutions in the new $P$ are the best $b_1$ solutions in
                 the current *RefSet*.)
        **End If**
    **End For**
```

5. Simulated Annealing:

The principle of the **SA** metaheuristic is deduced from the physical annealing process of solids. Kirckpatrick et al. [21] and Cerny [22] proposed the use of SA for combinatorial problems. Their work is based on the research of Metropolis et al. [23]

in the field of Statistical Mechanics. For an overview of the research and applications of SA, the reader is referred to Vanlaarhoven and Aarts [24], Aarts and Korst [25], Collins et al. [26] and Eglese [27].

The representation of solution, definition of cost function and neighbour solution are defined in section 2 and the initial solution is produced completely random.

As far as our implementation is concerned, the following choices have been made. In order to determine the value of the *initial temperature*, T_{begin} is computed by solving the expression:

$$P_a = e^{-\Delta C / T_{begin}}$$

and hence

$$T_{begin} = \frac{-\Delta C}{\ln P_a} \qquad (1)$$

Here ΔC represents the average deterioration value, which is computed as the cumulative value of the values of all worsening moves possible from the initial solution, divided by the number of moves, which cause a deterioration of the objective function value. Parameter P_a represents the acceptance fraction, i.e. the ratio of the accepted to the total number of generated moves.

The *cooling function* we use for the reduction of the temperature is the simple geometric function.

The temperature at iteration t, T_t, is obtained from the temperature of the previous iteration as follows:

$$T_t = R.T_{t-1} \qquad (2)$$

Here, R represents the cooling rate and we consider it equal to 0.99.

5.1. Algorithm

A general description of SA is given in Table 2.

Table 2: The General Simulated annealing technique

```
Select an initial state $i \in S$
Select an initial temperature $T > 0$;
Set  temperature change counter $t = 0$;
Repeat
        Set repetition Counter $n = 0$;
        Repeat
                Generate state j, a neighbour of i;
                Calculate $\delta = f(j) - f(i)$;
                if $\delta < 0$ Then $i := j$;
                else if random (0,1) < $\exp(-\delta/T)$ Then $i := j$;
                $n := n+1$;
        Until $n = N(t)$;
         $t := t+1$;
        $T := T(t)$;
Until Stopping Criterion true.
```

6. Tabu Search

Tabu search was conceived by Glover [28]. TS is base on the principles of intelligent problem solving.

The representation of solution, definition of cost function and neighbour solution are defined in section 2 and the initial solution is produced completely random.

Each time a *move* is performed and linked the couple (exam, period) to *tabu list* that includes inhibited moves. It means, period of this exam can't change until | *tabu list* | (length of tabu list) times. From a given solution not all neighbours can usually be reached. A new candidate move in fact brings the solution to its best neighbour, but if the move is present in the tabu list, it is accepted only if it decreases the objective function value below the *aspiration level*. *Aspiration level* is minimum of cost function so far achieved. This process is repeated until a stopping criterion is reached. The stopping criterion of this algorithm is reaching to the limited number of iteration between current iteration and iteration that best solution is reached.

A good overview of TS and its applications is provided by Glover, Laguna and Rosing [29,30].

6.1. Algorithm

A general description of TS is given in Table 3.

Table 3: The general tabu search technique

```
Initialization
    s: = initial solution in X;
    nbiter: = 0;
        (* Current iteration *)
    bestiter: = 0;
        (* iteration when the best solution has been found *)
    bestsol: = s;
        (* Best solution *)
    T: = $\phi$; (* T is Tabu list *)
    Initialize the aspiration function A;
While (f (s) > f*) and (nbiter-bestiter < nbmax) do
    nbiter: = nbiter + 1;
    Generate a set V* of solutions $s_i$ in N(s) which are either
    not tabu or such that A( f(s)) >= f ( $s_i$ ) ;
    Choose a solution s* minimizing f over V*;
    Update the aspiration function A and the tabu list T;
    If f(s*) < f(bestsol) then
    bestsol: = s*; bestiter: = nbiter ;
        s: = s* ;
End while
```

7. Datasets:

We produce several random problems in different size in order to apply these algorithms for different ones. In these problems the number of exams averts from 40 to 200 and respect to the number of exams, the number of students and periods has been determined. The elements of conflict matrix of student A_{ij} (that shows the common students in both *i* and *j* exams) have been produced randomly. You can see the information about these problems in the Table 4.

Table 4: Characteristics of Data Sets

Data set	Exams	Timeslots	Students
1	40	15	800
2	60	15	1400
3	80	20	1900
4	100	24	2850
5	120	20	3600
6	140	24	4552
7	150	25	4800
8	160	32	5226
9	180	28	6540
10	200	30	7000

8. Metaheuristic analysis

Due to the fact that the stopping criterions of the metaheuristics are not defined samely, a simple comparison of only the final solution values of the three metaheuristics would not be appropriate.

Besides, the computing time of heuristics highly depends on the value assigned to the parameters. Also it is difficult to estimate the processing time of heuristics. Moreover, the probability of finding a better final solution increases with the run time. Therefore a simple comparison of the final solution of the three metaheuristics without taking into account the run time is not appropriate.

An important analysis tool for the dynamic heuristic analysis is the graphical representation of the path of the objective function value of each heuristic versus computing time. Example is given in Fig.1.

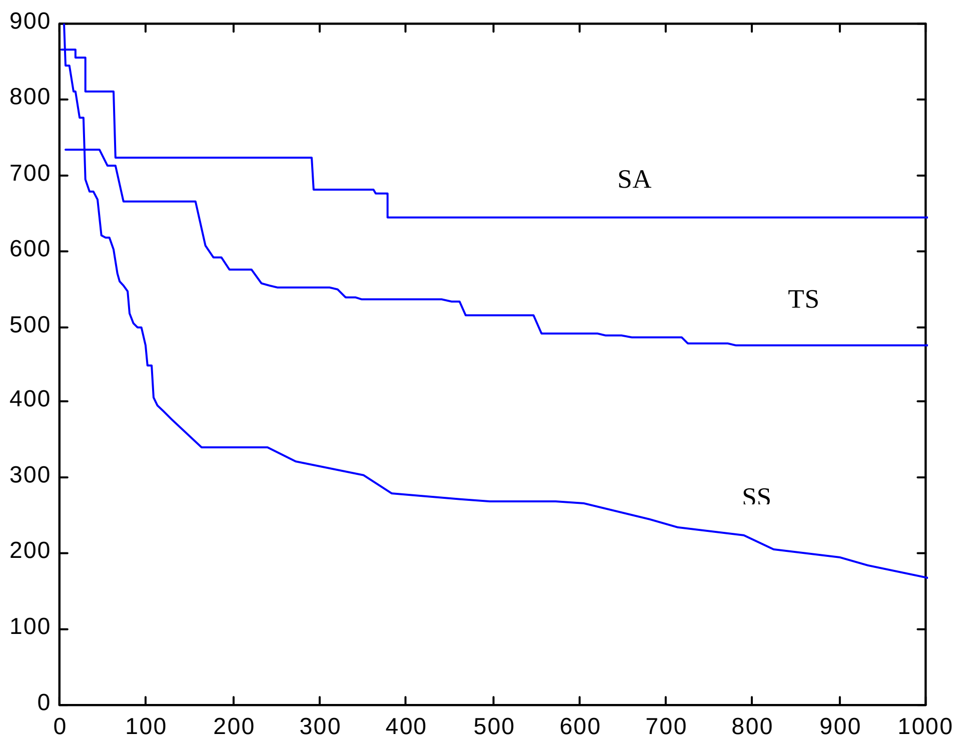

Fig.1. Example of the path of the objective function value versus computing time for Simulated Annealing, Tabu Search and Scatter Search for average solutions for all instances.

An alternative for comparing the improvement heuristics dynamically is required. The specific feature of the dynamic analysis is that intermediary solutions of metaheuristics at various time points are compared. We considered three time points and report the minimum value of cost for each algorithm at these times (which are named minimum 1,2,3).

In Table .5 the symbol '*' indicates which heuristic attains its minimal value after the given run time. The best solutions of the three metaheuristics at each time point are printed in bold face. The column at the right of each cell contains the relative difference with respect to the best solution at that time point and this formula:

$$relative\ difference(solution) = \frac{\cos t\ of\ solution - \cos t\ of\ best\ solution}{\cos t\ of\ best\ solution}$$

In column 1, each S_i is consisting to instance that we generated and explained in section 7 and in column 3 we report the cost of initial solution.

The same computer has been used for all experiments and our computer is Pentium III, 600MHZ(half cash), under IBM and the program is written in MATLAB software language under Microsoft Windows 98.

Table 5. Heuristic analysis of SA, TS and SS

Problem		*Initial*	*Minimum 1*		*Minimum 2*		*Minimum 3*	
S1	*Time*		50		140		180	
	SA	938.5788	881.9550	0.48	813.7112	0.90	765.5462*	1.09
	TS	877.0063	732.1413	0.23	532.6425*	0.24	532.6425	0.45
	SS	774.4750	**594.1800**		**428.7437**		**366.1112***	
S2	*Time*		30		120		480	
	SA	1116.4	1100.2	0.03	1038.1	0.14	976.0064*	0.30
	TS	1197.5	**1066.8**		**912.3136**		835.0114*	0.12
	SS	1242.4	1225.6	0.15	969.4100	0.06	**747.4143***	
S3	*Time*		60		300		540	
	SA	1217.3	1217.3	0.04	1150.3	0.20	1059.2*	0.17
	TS	1365.0	1802.8	0.54	**956.0505**		**905.3221***	
	SS	1175.8	**1170.9**		1058.5	0.10	971.1268*	0.07
S4	*Time*		60		300		540	
	SA	940.2081	723.8502	0.07	697.2733	0.14	676.2716*	0.15
	TS	750.8274	697.1719	0.03	646.0586	0.06	**585.7909***	
	SS	690.7649	**674.6260**		**611.0365**		595.9551*	0.02
S5	*Time*		100		300		540	
	SA	1015.2	994.5783	0.11	979.9100*	0.10	979.9100	0.13
	TS	1125.5	1001.8	0.12	952.2086	0.07	915.0758*	0.06
	SS	899.8344	**895.5561**		**888.5972**		**863.5269***	
S6	*Time*		300		600		960	
	SA	1456.3	1417.6	0.12	1417.6	0.17	1379.2*	0.14
	TS	1468.8	1390.4	0.10	**1207.8***		**1207.8**	
	SS	1360.8	**1268.0**		1252.0	0.04	1235.9*	0.02
S7	*Time*		350		700		1020	
	SA	1739.3	1494.2	0.11	1494.2	0.13	1355.9*	0.06

	TS	1736.4	1434.1	0.07	1335.0	0.01	**1273.5***	
	SS	1352.2	**1340.8**		**1318.4**		1308.0*	0.03
S8	*Time*		380		800		1200	
	SA	1266.3	1147.4	0.04	1143.9	0.04	1124.9*	0.06
	TS	1304.3	1258.5	0.14	1147.4	0.04	1068.3*	0.01
	SS	1136.1	**1100.5**		**1100.5**		**1061.9***	
S9	*Time*		470		670		1200	
	SA	1507.6	1389.5*	0.05	1389.5	0.05	1389.5	0.07
	TS	1546.0	1376.0	0.04	1347.4	0.04	1299.3*	0.00
	SS	1322.1	**1321.4**		**1321.4**		**1295.2***	
S10	*Time*		300		640		1200	
	SA	1573.2	1557.7	0.08	1557.7	0.09	1441.4*	0.08
	TS	1624.6	1530.6	0.06	**1427.4**		**1334.7***	
	SS	1440.6	**1438.2***		1438.2	0.01	1438.2	0.11

As it is shown in the Table.5_6, we gain the best solution from SS in different times. SS is in first grade and then TS after that SA has the third grade.

Table 6. Ability of metaheuristics to find the best solution

Algorithm	SA	TS	SS
Ability of Alg. to find the best solution	% 0	%33.3	%66.6

The values that have been shown in above table achieved from following formula:

$$\frac{The\ number\ of\ po\mathrm{int}s\ that\ a\lg orithm\ has\ been\ found\ the\ best\ solution}{The\ number\ of\ all\ po\mathrm{int}s}\times 100$$

Since the initial solution of SA and TS are produced completely random and they are different and this effects on quality of the best solution, which is reached on each time point, for fair comparison we calculate amount of cost reduction for these algorithms. The difference between cost of initial and final solution (minimum 3) for each data set is given in table 7. In this way we can compare them fairly.

Table 7.Values of cost reduction in each data set for SA and TS

Algorithm/test problem	**S_1**	**S_2**	**S_3**	**S_4**	**S_5**	**S_6**	**S_7**	**S_8**	**S_9**	**S_{10}**
SA	173.0	140.3	158.1	**263.9**	35.29	77.1	383.4	141.4	118.1	131.8
TS	**344.3**	**362.4**	**459.6**	165.0	**210.42**	**261.0**	**462.9**	**236.0**	**246.7**	**289.9**

As it is shown in Table.7 TS has maximum cost reduction in %90 cases and it worked better than SA. This result is consisting with the result of Tables 5,6.

9. Comparison with published Results

In previous section it is shown that Modified Scatter Search is better than SA and TS. Now this best algorithm is compared with other published results on the same problem and Carter's test bed [6]. We consider Sequencing Heuristics with backtracking (SH) [6], Tabu Search with a variable tabu list (TS) [14] and Degraded Ceiling algorithm (DC) [11]. These results are shown in Table 8. The best results are presented in bold.

Table 8.Published and our results for proximity cost of ETP

Data Sets	*Exams*	*Students*	*Time slots*	*Published Results (Best Cost/ Time for best)*						*Modification of Scatter Search Algorithm*	
				SH	*Time*	*TS*	*Time*	*DC*	*Time*	*Best cost*	*Time for best*
EAR-F-83	190	1125	24	36.4	24.7	45.7	4.6	**35.4**	134	41.6693	17.2
HEC-S-92	81	2823	18	10.8	7.4	12.4	3.7	10.8	278	**10.4623**	388.6
KFU-S-93	461	5349	20	14.0	120.2	18.0	12.3	**13.7**	729	17.8267	403.7
LSE-F-91	381	2726	18	10.5	48.0	15.5	20.3	**10.4**	1030	14.2913	370.6
RYE-S-93	486	1483	23	7.3	507.2	__	__	8.9	752	**7.1220**	190.0
STA-F-83	139	611	13	161.5	5.7	160.8	3.9	159.1	157	**95.3502**	18.1
TRE-S-92	261	4360	23	9.6	107.4	10.0	16.2	**8.3**	392	11.5229	187.9
UTE-S-92	184	2750	10	25.8	9.1	29.0	42.4	25.7	236	**18.0058**	296.6
YOR-F-83	181	941	21	41.7	271.4	41.0	25.2	36.7	546	**34.8321**	165.0

10. Conclusions:

As it is shown in Tables 5_6, SS works better than TS and SA. One of its reasoning refers to our modifications; in both TS and SA, one solution is constructed randomly and then improved. In those algorithms, neighbour solution may have no similarity to current one and just the cost of it, is less than current solution. But in our modification of SS the solutions are not constructed randomly. While constructing each solution, amount of increasing cost is considered, and also we try to construct the solution, which is similar to the best solution by justifying the amount of the credits of each pair (exam,period). As we mentioned before, the information of best solution is considered and the new solution is generated based on it. So we get each new solutions of initial set better than previous ones.

Also we use a strong searching engine _*TS* _ as the second improvement method of SS, which improved the solution well. Also existence of diverse solutions in Refset inhibits the fast convergence of the algorithm and cause to have better solutions in solution space.

In addition we use two different generation methods in two steps of SS: generation of initial solutions and generation of new solutions after we can’t obtain any new solution regards to mentioned conditions. We construct initial solutions in artificial manner and use the information of best solution in generation of new solution in Mixed Diversification and First Improvement Method. In this way we generate high quality initial solutions and this effects on final solutions. As it is shown in Table.5, SS starts with the better initial solution in 90% of problems. So regards to Table.6 we have better solutions from SS in 66% cases. Also we use Second Generation Method when we are at local optimum point and we generate diverse solutions by this method. So we clime up from these points and we get better solutions.

Using Ts as a part of SS instead of applying it after SS sequent has advantages for us. If we use SS and then apply TS after it, TS may converge in a local optimum, but if we apply TS as a part of SS, we can inhibit from fast convergence. Because of in our modification of SS, we use Second Generation Method to regenerate the Refset with diverse solutions for continuing the search process and climb up from local optimum points. So in this way we can obtain better solutions.

In compare of real datasets, as it is shown in Table.8 SS works better than other published algorithms in five cases and it is hopeful for us to apply it in real problems.

The computing times are incomparable due to the use of different hardware. But we present the computing time for best results only to show that the given solutions were produced in quite acceptable time.

Acknowledgment

I offer my thanks to Dr.Ibrahim Osman for his good comments to this paper.

References:

[1] Welsh D.J.A and Powell M.B, An Upper bound for the Chromatic Number of a

Graph and Its Application to Timetabling Problems, Comp. Jrnl 10 (1967) 85-86 .

[2] Karp R.M , Reducibility among combinatorial Problems, In Complexity of computer computations, Plenum Press, New York,1972 .

[3] D. de Werra , The combinatorics of timetabling, European Journal of Operational Research 96(1997)504-513 .

[4] A.S. Asratian and D. de Werra , A generalized class_teacher model for some timetabling problems , European Journal of Operational Research 143(2002) 531-542

[5] Michael W. Carter, A survey of practical applications of Examination Timetabling Algorithms, Operation Research 34(2)(1986)193-202 .

[6] Michael W.Carter, Gilbert Laporte, Sau Yan Lee, Examination Timetabling : Algorithmic Strategies and Applications, Journal of the Operational Research Society 47(1996) 373-383 .

[7] Mirjana Cangalovic,Jan A.M. Schreuder, Exact Colouring algorithm for weighted graphs applied to timetabling problems with lectures of different lengths, European Journal of Operational Research 51(1991)248-258 .

[8] Luis F. Paquete, Carlos M. Fonseca , A Study of Examination Timetabling with Multiobjective Evolutionary Algorithms, MIC 2001- 4 th metaheuristic International Conference. Porto, Portugal, July 16-20, 2001 .

[9] O.Rossi-Doria, Ch. Blum, J. Knowles, M.Sampels, K. Socha, B. Paechter, A Local Search for the Timetabling Problem, Technical Report No.TR/IRIDIA/2002-16, July 2002 .

[10] Edmund Burke, Yuri Bykov, James Newall and Sanja Petrovic, A Time-Predefined Local Search Approach to Exam Timetabling Problems, Computer Science Technical Report No. NOTTCS-TR-2001-6 **.**

[11] E.K. Burke, Y. Bykov, J.P. Newall and S.Petrovic, A New Local Search Approach with Execution Time as an Input Parameter, Computer Science Technical Report No. NOTTCS-TR-2002-3 .

[12] Jonathan M. Thompson, Kathryn A. Dowsland, A Robust Simulated Annealing Based Examination Timetabling System, Computers and Operations Researchs 25(7/8)(1998)637-648 .

[13] A. Hertz, Tabu search for large scale timetabling problems, European Journal of Operational Research 54(1991)39_47 .

[14] Luca Di Gaspero and Andrea Schaerf, Tabu Search Techniques for Examination

Timetabling, Third International Conference Patat 2000 , Lecture Notes in Computer Science 2079(2000)p.104-117 .

[15] J Wood, D Whitaker, Student centred school timetabling, Journal of the Operational Research Society 49(1998)1146-1152 .

[16] Victor A. Bardadym, Computer-Aided School and University Timetabling:The New Wave, Lecture Notes in Computer Science 1153(1996) 22-45 .

[17] Safaai Deris, Sigeru Omatu, Hiroshi Ohta, Timetable planning using the constraint-based reasoning, Computer & Operations Research 27(2000)819-840 .

[18] Edmund Kieran Burke, Sanja Petrovic, Recent research directions in automated timetabling, European Journal of Operational Research 140(2002)266-280 .

[19] Manuel Laguna,Scatter Search,to appear in handbook of applied optimization,P.M.Pardalos and M.G.C.Resende(Eds),Oxford Academic Press(2000).

[20] Fred Glover,Manuel Laguna and Rafael Marti , Scatter Search ,to appear in Theory and Applications of Evolutionary Computation : Recent Trends, A . Ghosh and S.Tsutsui(Eds),Springer-Verlag(1999) .

[21] Kirckpatric S,Gellat Jr. C,Vecchi M,Optimization by simulated annealing, Science 220(1983)671-680 .

[22] Cerny V, A Thermodynamical approach to the travelling salseman problem : an efficient simulation algorithm, Journal of Optimization Theory Application 45(1985)41-51 .

[23] Metropolis N, Rosenbluth A, Rosenbluth M, Teller A,Teller E, Equation of state calculations by fast computing machines, Journal of Chemical Physics 21(1953)1087-1092 .

[24] Van Laarhoven P, Aarts E, Simulated Annealing :Theory and Practic, Dordrecht:Kluwer Academic Publishers, The Netherlands, 1987 .

[25] Aarts E. Korst J, Simulated annealing and Boltzmann machines, Chichester: Wiley,1989 .

[26] Collins N, Eglese R, Golden B, Simulated annealing an annotated bibliography, American Journal of Mathematical and Management Sciences 8(1988)209-307 .

[27] R.W.Eglese,Simulated Annealing : A tool for Operational Research , European Journal of Operational Research 46(1990)271-281.

[28] Glover F, The general employee scheduling problem: an integration of management science and artificial intelligence, Computers and Operations

Research 15(1986)563-593 .
[29] Glover F., Laguna M, Tabu search, Kluwer academic Publishers, 1997 .
[30] K.E.Rosing, C.S.ReVelle, E.Rolland, , D.A.Schilling, J.R.Current, Heuristic concentration and Tabu Search ：A head to head comparison, European Journal of Operational Research 104(1998)93-99.