International Journal of Soft Computing 3 (3): 230-238, 2008
ISSN: 1816-9503
© Medwell Journals, 2008

Enhancing Max-Min Ant System for Examination Timetabling Problem

F. Djannaty and A.R. Mirzaei
Department of Mathematics, University of Kurdistan, Sanandaj, Iran

Abstract: Examination Timetabling Problem (ETP) is a real life problem encountered in many academic institutions and has attracted the attention of the Operational Research and Artificial Intelligence research communities since the 1960s. In this study, a variant of Ant Colony Optimization (ACO), the Max-Min Ant System (MMAS) is used to solve the Examination timetabling problem. The key feature of our approach is the combination of a simple local search and MMAS. A preprocessing heuristic is utilized to initially sort the exams. Great Deluge algorithm is used as a local search to improve the constructed solutions by MMAS. We applied our algorithm to a number of test problem data sets. The numerical results obtained from our method shows that the quality of the solutions are better than some or tie the best-published results from the literature, especially on capacitated examination timetabling problem.

Key words: Ant colony optimization, examination timetabling, great deluge algorithm, local search, preprocessing

INTRODUCTION

Examination timetabling is a well-studied combinatorial optimization problem. A significant amount of timetabling research papers on this problem is published in the last decade. This problem is important for educational institutions and requires expensive human and computer resources.

Scheduling an examination timetable is accomplished manually in many high schools, colleges and universities throughout the world, which causes some dissatisfaction among the students and lecturers. The construction of a timetable can be an extremely difficult task for managers and administrators and its solution can require extensive efforts. The quality of solutions in timetabling often has significant impact on the associated institutions.

The examination timetabling problem is essentially defined as scheduling a number of exams into a limited number of timeslots or periods, while satisfying the maximum number of constraints from a prespecified set of constraints. These constraints vary widely from institution to institution. Thus examination timetabling problems are usually varied in their size, complexity, constraint and etc.

Managing a high variety of different constraints is quite a difficult task. Every additional constraint can increase the total complexity of the problem and can make the solution more resource-consuming.

These problems are known to be NP-complete (Karp, 1972). This feature can contribute to making them a difficult class of problems, suggesting some serious research challenges. The difficulties in solving the problem can be represented by the degrees of the vertices in an undirected graph which models the examination timetabling problem by representing the exams as vertices and conflicts by edges. Two exams involving the same students are said to have a conflict. The difficulty of an exam is represented by the number of conflicts it has with other exams. Thus, the objective (goal) of the examination timetabling process can be taken to be that of producing the feasible timetable of the highest possible quality (minimum value of a particular cost function), subject to a number of constraints explained.

Constraints: The constraints in an examination timetabling problem are usually divided into classes, hard constraints and soft constraints, which are described as:

- Hard constraints are those which cannot be violated under any conditions. For example,
- Two exams involving a number of common students cannot be scheduled into the same timeslot.
- The number of students taking part in an exam should not exceed the number of available seats for that exam.
- Soft constraints are those which are desired to be satisfied, but are not absolutely necessary. Practically, it may not be possible to find a feasible

Corresponding Author: F. Djannaty, Department of Mathematics, University of Kurdistan, Sanandaj, Iran

solution that satisfies all of the soft constraints. Soft constraints are usually varied from one institution to another, in terms of both the type and their importance. Sometimes, soft constraints conflict with each other. For example,

- Conflicting exams are better to spread throughout the examination session, so that students should not have exams in consecutive times lots or two exams on the same day.
- Some of the most crowded exams should be scheduled as early as possible to allow enough time for marking.
- Ordering (precedence) of exams is needed to be satisfied.

A solution of examination timetabling which have not violated hard constraint is called a conflict-free assignment. As the real-world exam timetabling problem is difficult, some of the soft constraints may need to be relaxed, because it is not usually possible to generate solutions without violating these constraints, therefore, they are penalised in the objective function.

Timetabling problems, without soft constraints, can be modeled as graph coloring problems. Welsh and Powell (1967) built the bridge between graph coloring and timetabling, which led to a significant amount of later research on graph heuristics in timetabling. The graph coloring problem is assigning colors to vertices so that no adjacent vertices have the same color. The early approaches to exam timetabling tended to employ ordering heuristic where a heuristic is used to measure the difficulty of scheduling a particular exam. Carter *et al.* (1996) studied the first five ordering strategies on real and randomly generated exam timetabling problems.

Constraint programming methods have attracted the attention of researchers in timetabling due to the ease and flexibility with which they can be employed for timetabling problems. Merlot *et al.* (2003) employed constraint programming to produce initial solutions. Then a Simulated Annealing and a hill climbing method were used to improve the solutions. The pure constraint programming obtained the best result for one of the Toronto datasets.

In general, there are a large number of metaheuristics for solving a given examination timetabling problem. Some of these metaheuristics such as tabu search simulated annealing and their variants are based on local search. Tabu Search explores the search space by storing recent moves in a tabu list so that these moves are not visited again. One of the earliest investigations of this metaheuristics for exam timetabling was presented by Hertz (1991). White and Xie (2001) presented a tabu search based approach to examination timetabling. Their approach applied both short term memory and longer term memory to find good quality solutions. Di Gaspero and Schearf (2001) investigated a tabu search method for exam timetabling. Their approaches are heavily dependent on graph coloring based heuristic methods. They present a relatively detailed comparison of their approach with other approaches on a range of benchmark problems and produced very competitive results. These benchmark problems are used in the current paper. Simulated annealing is motivated by the natural annealing process. The idea is to search a wider area of the search space at the beginning of the process by accepting worse moves with a higher probability, which is gradually decreased as the search continues. Thompson and Dowsland (1996) accomplished a valuable work to develop a two-stage approach where feasible solutions from the first stage were improved in the second stage by Simulated Annealing concerning soft constraints. Burke *et al.* (2004) studied a variant of Simulated Annealing, in which the search accepts worse moves as long as the decrease on the quality is below a certain level. Another classes of metaheuristics applied on examination timetabling problems are evolutionary algorithms which are based on population, such as Genetic algorithms, Memetic algorithms and Ant algorithms. Corne *et al.* (1994) provided a brief survey on using Genetic Algorithms in general educational timetabling which was updated in 2003 by Ross *et al.* (2003). Memetic algorithm is a metaheuristic where the neighbourhood of the solutions obtained by a genetic algorithm is explored and the search toward the local optima (for each solution) is navigated, before returning to the genetic algorithm and continuing the process. Burke *et al.* (1996) developed a Memetic Algorithm which employs light and heavy mutation operators to reassign single exams and sets of exams, respectively, with the aim of escaping from local optima. To the best of our knowledge there are a few publications on the topic of Ant Colony Optimization and examination timetabling problem. Naji Azimi (2005) implemented an Ant Colony System algorithm for ETP and compared it with simulated annealing, tabu search and a genetic algorithm under a unified framework for solving some exam timetabling problems. It was observed that the hybrid approaches work better than each single algorithm. However, only randomly generated data was tested. Another work in this context is by Dowsland and Thompson (2005) who developed Ant Algorithms based on the graph coloring model for solving exam timetabling problems without soft constraints.

PROBLEM FORMULATION

The problem model proposed in this research adopts the following notation:

- E is the set of n exams, E = $\{e_1, e_2,...,e_n\}$.
- S is the set of q students, S = $\{s_1, s_2,...,s_n\}$.
- T is the set of p timeslots, T = $\{1,2,..., p\}$.
- $C_{n\times n}$ is a conflict matrix where c_{ij} tells us the number of students enrolled in both exams e_i and e_j.
- $Y_{n\times p}$ is a binary matrix such that $y_{it} = 1$ when exam e_i is assigned to the timeslot $t \in T$ and $y_{it} = 0$ otherwise.
- Cap_t is the number of seats available in timeslot t.

Basically, the mathematical model of the examination timetabling can be stated as follows:

$$\begin{aligned}
&\text{find} && y_u \quad i=1,2,...n \quad t=1,2,...,p\\
&\text{s.t.} && \sum_{t=1}^{p} y_{it}=1 \quad i=1,2,...,n\\
& && \sum_{t=1}^{p} c_{ij} y_{it} y_{jt}=0 \quad i,j=1,2,...n \quad i\neq j\\
& && \sum_{i=1}^{n} c_{ii} y_{it} \leq Cap_t \quad t=1,2,...,p\\
& && y_u = 0 \text{ or } 1 \quad i=1,2,...,n \quad t=1,2,...,p
\end{aligned}$$

Constraint 1 guarantees that each exam must be allocated to exactly one and only one timeslot. Constraint 2 ensures that no student takes two exams scheduled at the same timeslot and Constraint 3 tells that the total number of students having exam in a timeslot should not exceed the total number of seats available in that timeslot. In the current work only the above hard constraints are considered and any set of binary variables y_{it} which satisfies these constraints constitute a feasible solution. Soft constraints are considered by penalizing them in the objective function of the model.

ANT COLONY OPTIMIZATION

Ant Colony Optimization is a metaheuristic approach for solving hard combinatorial optimization problems which is proposed by Dorigo *et al.* (1996). As the name suggests, ACO has been inspired by the behavior of real ant colonies, in particular by their foraging behavior. While walking from food sources to the nest and vise versa, ants deposit a substance called pheromone on the ground. Paths marked by strong pheromone concentrations are more probable to be chosen when deciding about a direction to go. One of the main ideas is to adopt the indirect communication among the individuals of a colony of ants through pheromone, by artificial ants. Artificial ants used in ACO are stochastic solution construction procedures that probabilistically build a solution by iteratively adding solution components to a partial solution, taking into account heuristic information on the problem instance being solved, if available and (artificial) pheromone trails which is changed dynamically at run-time to reflect the agents' acquired search experience. Artificial pheromone trails are a kind of distributed numeric information which is modified by the ants to reflect the experience accumulated while solving a particular problem. This behavior is a basis for a cooperative interaction which leads to the emergence of shortest paths, thus minimizing the length of the path between nest and food source. The seminal work on ACO algorithm was Ant System (AS). In recent years some changes and extensions of AS have been proposed, e.g., Ant Colony System (ACS) and Max-Min Ant System. The standard framework of an ACO algorithm is shown in Fig. 1.

```
 Procedure ACO algorithm
     While (termination condition not met) do
       ConstructSolutions
       ApplyLocalSearch          %optional
       UpdateTrails
     End
End
```

Fig. 1: General framework of ACO

The three steps, construct solution, apply local search and update trails in Fig. 1 are explained in the implementation of ACO for the Traveling Salesman Problem (TSP). The TSP can be represented as a complete graph with a number of nodes also called cities. It is necessary to find a shortest closed tour visiting each of the cities exactly once. Initially, each ant is put on a randomly chosen city and has a memory which stores the partial solution it has constructed so far (Firstly, the memory contains only the start city). Starting from its start city, an ant iteratively moves from city to city.

When at a city i, an ant k chooses to go to an as yet unvisited city j with a probability given by:

$$p_{ij}^{k}(t) = \frac{[\tau_{ij}(t)]^{\alpha}.[\eta_{ij}]^{\beta}}{\sum_{l\in N_i^k}[\tau_{ij}(t)]^{\alpha}.[\eta_{ij}]^{\beta}} \quad \text{if } j\in N_i^k$$

Where $\eta_{ij} = l/d_{ij}$ and d_{ij} is distance from city i to city j. α and β are two parameters which determine the relative influence of pheromone trail and heuristic information and N^k_i is the set of cities which ant k has not yet visited. The next step is local search which is optional and is used in order to ensure local optimality. The solution construction ends after all ants have completed their solution, then, the pheromone trails are updated. For instance in AS this is done by first lowering the pheromone trails by a constant factor (this is pheromone evaporation) and then allowing each ant to lay down pheromone on the edges that belong to its tour:

$$\forall (i,j)\ \ \tau_{ij}(t+1)=(1-\rho).\tau_{ij}(t)+\sum_{k=1}^{m}\Delta\tau_{ij}^{k}(t)$$

Where $0 < \rho \leq 1$ is the pheromone trail evaporation rate and m is the number of ants. The parameter ρ is used to avoid unlimited accumulation of the pheromone trails and enables the algorithm to forget previous bad decisions. In AS, $\Delta\tau^{k}_{ij}$ is defined as follows:

$$\Delta\tau_{ij}^{k}(t)=\begin{cases}1/L^{k}(t) \text{ if arc } (i,j) \text{ is used by ant } k \text{ in iteration } t\\ 0 \quad \text{otherwise}\end{cases}$$

Where L^k (t) is the tour length of the k th ant. One of the key points in developing an ACO algorithm is to establish an appropriate balance between pheromone update and heuristic information.

OUR APPROACH ON ETP

Preparations have to be made to initiate the MMAS algorithm. Initially, we introduce a solution representation for examination timetabling problem. We show each solution by a vector, with the number of components equal to the number of exams. Each component of this vector shows the assigned timeslot for that exam. Timeslots and exams are numbered as follow.

Exam	1	2	3	4	...	...	n-2	n-1	n
Timeslot	10	13	1	4	...	...	10	1	5

We denote each solution of length n by x such that xi = t if $y_{it} = 1$ (i = 1, 2,..., n), for example in the above representation $x_3 = 1$ or ($y_{31} = 1$), $x_n = 5$ or ($y_{n5} = 1$). We have incorporated a preprocessing and a local search in our algorithm which is explained in the following.

Preprocessing: Our preprocessing is carried out in two stages. First, exams are sorted based on Largest Degree first (LD) in which exams are ordered decreasingly by the number of conflict they have with other exams. Second, if two exams have the same LD, then an exam with Larger Enrollment degree (LE) has the priority. LE degree of an exam is the number of students enrolled in that exam.

Local search: The main feature in our implementation of MMAS algorithm is local search. In the algorithm proposed here, the Great Deluge Algorithm (GDA) of Dueck (1993) is used as local search.

```
Set the initial solution s
Specify input parameter $\Delta B$.
Calculate initial cost function f(s)
Initial level B = f(s)
 While stopping condition not met do
   Define neighbourhood N(s)
   Randomly select the candidate solution $s^* \in N(s)$
     If $f(s^*) \le f(s)$ or $f(s^*) \le B$
        s = s*
        $B = B - \Delta B$
     End if
 End while
```

Algorithm 1: Extended Great Deluge Algorithm

GDA accepts every solution whose objective function is less than or equal to the upper limit B. The value of B is monotonically decreased during the search and bounds the feasible region of the search space. In order to prevent a premature convergence and thus to improve the performance of this method, we propose to extend it by accepting all the candidate solutions which are better than the current one. The pseudo code of this extended algorithm is shown in Algorithm 1.

The initial value of level B in algorithm 1 is equal to the initial cost function of initial solutions.

This forestalls sharp descent and idle step in the beginning of the search. In each step B., the decay rate, has to be specified. The neighbourhood we use in GDA is a slight variant of the Kempe chains used by Merlot *et al.* (2003). We choose a neighbour by selecting an exam e_i at random which is located in timeslot t from the set of all exams and selecting a session t' (t' ≠ t) at random from the set of sessions available for e_i in the current solution. Together e_i, t and t'. induce a Kempe chain and hence a neighbour of the current solution.

In the continuation, we described the general structure of the MMAS algorithm. One of the best ACO versions which have successfully been applied to optimization problems is MMAS, which was proposed by Stuzle and Hoos (2000). MMAS algorithms differs in three main aspect from AS.

- In order to exploit good solutions, only the best ant (the global-best or the iteration best) is allowed to add pheromone after each algorithm iteration.
- To limit the stagnation of the search a range of possible pheromone trails should be introduced, such as $[\tau_{min}, \tau_{max}]$, that is , $\tau_{min} \leq \tau_{ij} \leq \tau_{max}, \forall\tau_{ij}$.
- The pheromone trails are initialized to the upper trail limit, which causes the higher exploration at the start of the algorithm.

```
input: A problem instance I
$\tau_{max} \leftarrow 1/\rho$
$\tau_0 \leftarrow \tau_{max}$
$\tau(e, t) \leftarrow \tau_0 \ \forall(e,t) \in E \times T$
sort E by preprocessing operation
 while time limit not reached do
   for a = 1 to m do
      {construction process of ant a}
      $A_0 = \phi$
      for i = 1 to n do
            choose timeslot t randomly according to probabilities $P_{ei}$, t for
          event $e_i$
          $A_i = A_{i-0} \cup \{(e_i, t)\}$
      end for
      C ← solution after applying matching algorithm to $A_n$
        $C_{iteration best}$ ← best of C and $C_{iteration best}$
      end for
        $C_{iteration best}$ ← solution after applying GDA as local search to
        $C_{iteration best}$
   $C_{global best}$ ← best of $C_{iteration best}$ and $C_{global best}$
      global pheromone update for $\tau$ using $C_{global best}$, $\tau_{min}$ and $\tau_{max}$
end while
output: An optimized candidate solution $C_{global best}$ for I
```

Algorithm 2: MAX-MIN Ant System for the ETP

At the first glance, adding limits on the pheromone may seem quite unnatural. But, AS is only rather loosely coupled with the original behavior of ants and the main goal is to provide an effective tool for the solution of combinatorial optimization problem. The basic principle of MMAS for tackling problem is outlined in Algorithm 2.

At each iteration of the algorithm, each of m ants constructs, exam by exam, a complete assignment of the exams to the timeslots. For the construction of an exam-timeslot assignment each ant assigns sequentially timeslots to the exams, which are processed according to a pre-ordered list that was generated by our preprocessing. This means, it constructs assignments A_i: $E_i \rightarrow T$ for $C^o = 0, 1,..., n$. The selection of timeslots are guided by two types of information: heuristic information, which is an evaluation of the constraint violations caused by making the assignment, given that the assignments is already made and pheromone trail information, which is an estimate of the utility of making the assignment, as judged by previous iterations of the algorithm. The pheromone trail is represented by a matrix of pheromone values τ: $E \times T \rightarrow R_{\geq 0}$ where E is the set of exams and T is the set of timeslots. These values are initialized to a parameter τ_0 and then updated by a global pheromone update rule. Generally, an exam-timeslot pair which has been part of good solutions in the past will have a high pheromone value and consequently it will have a higher chance of being chosen again in the future. At the end of the iterative construction, an exam-timeslot assignment is converted into a candidate solution (timetable) using the matching algorithm. Among these solutions the algorithm chooses the best solution and this candidate solution is further improved by our local search routine. This candidate solution is used as the initial solution for the GDA algorithm. After all m ants have generated their candidate solution, a global update on the pheromone values is performed using the best solution found since the beginning. The whole construction phase is repeated, until the time limit is reached.

In the construction step we start with the empty assignment $A_0 = \phi$. After A_{i-1} has been constructed, the assignment A_i is constructed as $A_i = A_{i-0} \cup \{(e_i, t)\}$ where t is chosen randomly out of T with the following probabilities:

$$P(t=t'|A_{i-1},\tau)=\frac{[\tau(e_i,t')]^{\alpha}.[\eta(e_i,t')]^{\beta}}{\sum_{u\in T}[\tau(e_i,u)]^{\alpha}.[\eta(e_i,u)]^{\beta}}$$

Where

$$\eta(e_i,t')=\frac{1}{1+V(e_i,t')}$$

and V (e_i, t'). counts the additional cost of constraint violations caused by adding (e_i, t'). to the partial assignment A_{i-1}. In order to prevent the violation of hard constraints, a large penalty which is considered as an input parameter to the algorithm, is imposed on the objective function. The pheromone matrix is updated only once per iteration and the global best solution is used for update.

The following update rule is used:

$$\tau(e,t):=\begin{cases}(1-\rho).\tau(e,t)+1 & \text{if } A_{globalbast}(e)=t\\ -\rho).\tau(e,t) & \text{otherwise.}\end{cases}$$

Pheromone update is completed using the following:

$$\tau(e,t):=\begin{cases}\tau_{min} & \text{if } \tau(e,t)<\tau_{min}\\ \tau_{max} & \text{if } \tau(e,t)>\tau_{max}\\ \tau(e,t) & \text{otherwise}\end{cases}$$

COMPUTATIONAL EXPERIMENTS

We demonstrate the strength of our approach by evaluating it on three sets of accepted benchmark problems, the uncapacitated benchmark problems and two types of capacitated benchmark problems. We have not mentioned the computational times because many of the previous papers do not reports the relevant times and comparisons across very different platforms are impossible. All algorithms are coded in MATLAB version 7.0 and run on a Pentium IV with 1.8 GHz processor and 256 MB RAM in Windows XP system.

Table 1: Parameter for the MMAS and GDA

Parameter	m	ρ	α	β	τ_{max}	τ_{min}	ΔB
Value	12	0.3	2.8	1	3.3	0.0096	0.0005

The parameters for the algorithm described above were chosen after several experiments on the given test problem and reported in Table 1.

In GDA the local search iteration is stopped when the value of objective function is not improved after 1000 consecutive iterations. For each unscheduled exam a penalty as large as 10^4 is considered.

Uncapacitated benchmarks: In this problem instance it is tried to space out exams throughout the exam period. The objective function applies a penalty) w(t) to a timetable whenever a student has to sit two exams scheduled t period apart, with w(1) = 16, w(2) = 8, w(3) = 4, w(4) = 2 and w(5) = 1. The total penalty is divided by the number of students to get an average penalty per student. No account was taken of weekends and there was no differentiation between consecutive exam periods within the same day.

This objective function can be mathematically formulated as

$$\text{Minimize } \frac{1}{q}[\sum_{i=1}^{n}\sum_{\substack{j=1\\ j\neq 1}}^{n}\sum_{w=1}^{5}\sum_{t=w+1}^{p} c_{ij}\, y_{it}\, y_{j(t-w)}\, 2^{5-w}]$$

The constraints considered in this model are the same as those of problem formulation. On the other hand the algorithmic objective used in this study is equivalent to the objective function defined by Merlot *et al.* (2003) which is identical to that of Carter *et al.* (1996) calculated as follows:

```
Set $o(x) = 0$.
for each exam i in E:
        for each exam j in E, with $x_i < x_j < x_i + 6$ and $C_{ij} > 0$:
$o(x) := o(x) + w(x_j - xi)C_{ij}$
        endfor
endfor
```

In this series of experiments we employ Carter *et al.* (1996) collection of examination timetabling data that are available from (ftp://ftp.mie.utoronto.ca/pub/carter /testprob/). These problems were collected from different high schools and universities around the world. The characteristics of these problems are listed in Table 2.

Table 3 provides the comparison of our computational results with the previous state of the art on the data sets of Table 2 (compared with Carter *et al.* (1996), Caramia *et al.* (2001), Di Gaspero and Schaerf (2001), Merlot *et al.* (2003), Burke and Newall (1999), Casey and Thompson (2003) and Burke *et al.* (1996).

The best results are presented in bold. Our method is superior to that of Di Gaspero and Schaerf (2001) on all problems (with one tie) and produces better results than Carter *et al.* (1996) on 5 out of 12 data sets (with a tie on uta-s-92) and better than Casey and Thompson on 3 of 12 (with a tie and two are not reported). Compared with Burke and Newall (1999) better in 2 data sets and one is not reported and with Merlot *et al.* (2003) better in one plus one tie and in Burke *et al.* (1996) better in 2 data sets. Based on this comparison, the method of Caramia *et al.* (2001) produced the best results. However, our results are better than his results in three datasets.

Capacitated benchmarks 1: Burke *et al.* (1996) created a new class of capacitated examination timetabling problem for the publicly available date sets, having three sessions per weekday with one timeslot on Saturday and Sunday has none. It is assumed that the exam period starts on Monday. The first exam session starts from the first timeslot on Monday morning of the first week which is designated as timeslot 1 and this number is increased sequentially, so that, the single session of Saturday in the first week is numbered as 16 and this same session on next Saturday is numbered a 32. The total capacity of each timeslot is considered as a hard constraint, that is, room capacities are also added to the conflict-free assignment. The objective function is to minimize the number of students having 2 exams on the same day. For this problem class our algorithm uses an objective score identical to Merlot *et al.* (2003). Burke *et al.* (1996) have introduced a new data set of Nottingham University whose characteristics are mentioned in Table 4 and can be accessed from (ftp://ftp.cs.nott.ac.uk/ttp/Data/).

The following objective function which is a adsopted by Merlot *et al.* (2003) is undertaken here.

```
Set $o(x) = 0$.
for each exam i in E, with ($x_i$ mod 16) not equal {0, 3, 6, 9, 12, 15}:
    for each exam j in E, with $x_j = x_i + 1$ and $C_{ij} > 0$:
        $o(x): = o(x) + C_{ij}$
    endfor
endfor
```

The characteristics capacities of six more capacitated problems employed by us are listed in column 2 on Table 5 and results of the comparison of the performance of our algorithm with other algorithms are mentioned in Table 4 as well.

Table 2: Characteristics of toronto benchmark datasets

Problem Instance	Institution	Number of exams	Numberof timeslots	Number of students	Conflict density
car-f-92	Carleton University, Ottawa	543	32	18419	0.14
car-f-91	Carleton University, Ottawa	682	35	16925	0.13
ear-f-83	Earl Haig Colleagiate Institute,Toronto	190	24	1125	0.27
hec-s-92	Ecole des Hautes Etudes Commerciales,Montreal	81	18	2823	0.42
kfu-s-93	King Fahd University, Dharan	461	20	5349	0.06
lse-f-91	London School of Economic	381	18	2726	0.06
rye-f-92	Ryeson University, Toronto	461	23	11483	0.07
sta-f-83	St. Andrew's Junior High School, Toronto	139	13	611	0.14
tre-s-92	Trent University, Peterborough, Ontario	261	23	4360	0.06
uta-s-92	Faculty of Arts and Sciences, Uni. of Toronto	622	35	21266	0.13
ute-s-92	Faculty of Engineering, University of Toronto	184	10	2750	0.08
yor-f-83	York Mills Collegiate Institute, Toronto	181	21	941	0.29

Table 3: Results on the uncapacitated problem

Data set	Our method	Carter *et al.*	Di Gaspero and Schaerf	Burke and Newall	Merlot *et al.*	Casey and Thompson	Caramia *et al.*	Burke *et al.*
car-f-92	4.4	6.2	5.2	4.1	4.3	4.4	6.0	4.2
car-f-91	5.3	7.1	6.2	4.65	5.1	5.4	6.6	4.8
ear-f-83	36.8	36.4	45.7	37.05	35.1	34.8	29.3	35.4
hec-s-92	12.1	10.8	12.4	11.54	10.6	10.8	9.2	10.8
kfu-s-93	15.0	14.0	18.0	13.9	13.5	14.1	13.8	13.7
lse-f-91	11.3	10.5	15.5	10.82	10.5	14.7	9.6	10.4
rye-f-92	8.3	7.3	-	-	8.8	-	6.8	8.9
sta-f-83	158.2	161.5	160.8	168.73	157.3	134.9	158.2	159.1
tre-s-92	8.5	9.6	10.0	8.35	8.4	8.7	9.4	8.3
uta-s-92	3.5	3.5	4.2	3.20	3.5	-	3.5	3.4
ute-s-92	27.3	25.8	29.0	25.83	25.1	25.4	24.4	25.7
yor-f-83	39.1	41.7	41.0	37.28	37.4	37.5	36.2	36.7

Table 4: Characteristics of the nottingham university benchmark datasets

Data set	Institution	Number of exams	Number of timeslots	Number of students	Conflict density
Nott University,	Nottingham UK	800	23.26	7896	0.0311

Table 5: Results on the capacitated benchmarks 1

Data set	Capacity	Number of timeslots	Our method	Di gaspero and schaerf	Burke *et al.*	Merlot *et al.*	Caramia *et al.*
car-f-92	2000	31	503	424	331	158	268
car-f-91	1550	51	0	88	81	31	74
kfu-s-93	1955	20	80	512	974	247	912
tre-s-92	655	35	0	4	3	0	2
uta-s-92	2800	38	75	554	772	334	680
nott	1550	23	11	123	269	83	-
nott	1550	26	0	11	53	2	44

Clearly, our method is the superior method when applied to these capacitated versions of the data sets, providing best solutions in all instances except for data set car-f-92. However, with the car-f-92 dataset, even though our algorithm is run with higher number of iterations, the results are inferior in comparison with the result of other authors.

Capacitated benchmarks 2: Burke and Newall (1999) looked at some of the test problems of Table 2 and 4

Table 6: Results on the capacitated benchmarks 2

Data set	Capacity	Number of timeslots	Our method	Di gaspero and Schaerf	Burke and Newall (2001)	Merlot *et al.* (2003)	Carter *et al.* (1999)
car-f-92	2000	36	2804	3048	1665	1744	2915
kfu-s-93	1955	21	690	1733	1388	1082	2700
nott	1550	23	105	751	498	371	918

optimized them with a different objective function. They consider 3 exam sessions per day on weekdays and one exam session on Saturday morning. The objective function considers only students with 2 exams in 2 consecutive sessions. They give a penalty of 3 per student for 2 exams in a row in the same day and one per student for 2 exams in a row overnight. The arrangement and numbering of timeslots in the study is adopted here. The following objective functions which is also mentioned by Merlot *et al.* (2003) is utilized here.

```
Set $o(x) = 0$.
for each exam i in E, with ($x_i$ mod 16) not equal {0}:
        for each exam j in E, with $x_j = x_i + 1$ and $C_{ij} > 0$:
            if [($x_i$ mod 16) = {3, 6, 9, 12, 15}] then
                $o(x) := o(x) + C_{ij}$
        else
                $o(x) := o(x) + 3C_{ij}$
        end if
    endfor
endfor
```

Our computational results on these data sets are presented in Table 6.

Considering the fact that there are not many papers using these data sets, we had to compare our results with other methods in three data sets. As can be seen from Table 6, our method is much superior in 2 data sets.

CONCLUSION

We combined solution construction with local search. It is turned out that hybridized procedures perform better than single heuristics. Our approach outperformed the current state of the art on three standard benchmark problems that are usually used by researchers working with ETP. Although, our results are not satisfactory in all instances, but proved to be superior to those of others in capacitated timetabling problems. We understood that preprocessing heuristics are effective in finding feasible solutions for large scale ETP problems where finding such solutions is difficult. We realized that MMAS which is a strong version of Ant Colony Optimization can produce better results than other versions of ACO when dealing with a problem instance. On the other hand, since GDA is one of the local search algorithms which do not require many tunings, we propose to use this local search in combinatorial optimization problems where improvement of the solutions is considered.

It is proposed that our approach be applied to other combinatorial optimization problems where the solutions are made constructively.

REFERENCES

Burke, E.K. and J.P. Newall, 1999. A multi-stage evolutionary algorithm for the timetable problem. IEEE Transactions on Evolutionary Computation, 3: 63-74.

Burke, E.K., J.P. Newall and R.F. Weare, 1996. A Memetic Algorithm for University Exam Timetabling. In: Burke, E.K. and P. Ross (Eds.). Practice and Theory of Automated Timetabling: Selected Papers from the 1st International Conference. Springer Lecture Notes in Computer Science, 1153: 241-250.

Burke, E.K., Y. Bykov, J.P. Newall and S. Petrovic, 2004. A time-predefined local search approach to exam timetabling problems. IIE Transactions., 36: 509-528.

Caramia, M., P. DellOlmo and G.F. Italiano, 2001. New Algorithms for Examination Timetabling. In: Naher, S. and D. Wagner (Eds.). Algorithm Engineering 4th International Workshop, Proceedings WAE 2000. Springer Lecture Notes in Computer Science, pp: 230-241.

Carter, M.W., G. Laporte and S.Y. Lee, 1996. Examination timetabling: Algorithmic strategies and applications. J. Operational Res. Soc., 47: 373-383.

Casey, S. and J. Thompson, 2003. GRASPing the Examination Scheduling Problem. In: E.K. Burke and P. De Causmaecker (Eds.). Practice and Theory of Automated Timetabling: Selected Papers from the 4th International Conference. Springer Lecture Notes in Computer Science, 2740: 232-244.

Corne, D., P. Ross and H. Fang, 1994. Evolutionary timetabling: Practice, Prospects and Work in Progress. In: P. Prosser (Ed.). Proceedings of UK Planning and Scheduling SIG Workshop.

Di Gaspero, L. and A. Schaerf, 2001. Tabu Search Techniques for Examination Timetabling. In: Burke, E.K. and W. Erben (Eds.). (2001). Practice and Theory of Automated Timetabling: Selected Papers from the 3rd International Conference. Springer Lecture Notes in Computer Science, 2079: 104-117.

Dorigo, M., V. Maniezzo and A. Colorni. 1996. The Ant System: Optimization by a colony of cooperating agents. IEEE. Trans. Sys. Man and Cybernetics, 26: 29-41.

Dowsland, K.A. and J. Thompson, 2005. Ant colony optimization for the examination scheduling problem. J. Operational Res. Soc., 56: 426-438.

Dueck, G., 1993. New optimization heuristics: the great deluge and the recordtorecord travel. J. Computational Phys., 104: 86-92.

Hertz, A., 1991. Tabu search for large scale timetabling problems. Eur. J. Operational Res., 54: 39-47.

Karp, R.M., 1972. Reducibility among combinatorial problems. Complexity of Computer Computations. Plenum Press, New York.

Merlot, L.T.G., N. Boland, B.D. Hughes and P.J. Stuckey, 2003. A Hybrid Algorithm for the Examination Timetabling Problem. In: Burke, E.K. and P. De Causmaecker (Eds.), Practice and Theory of Automated Timetabling: Selected Papers from the 4th International Conference. Springer Lecture Notes in Computer Science, 2740: 207-231.

Naji Azimi, Z., 2005. Hybrid heuristics for examination timetabling problem. Applied Mathematics and Computation, 163: 705-733.

Ross, P., E. Hart and D. Corne, 2003. Genetic Algorithms and Timetabling. In: Ghosh, A. and S. Tsutsui (Eds.). Advances in Evolutionary Computing: Theory and Applications. Springer-Verlag New York, USA., pp: 755-771.

Stutzle, T. and H.H. Hoos, 2000. MAX-MIN Ant System. Future Generation Computer Sys., 16: 889-914.

Thompson, J. and K. Dowsland, 1996. Variants of simulated annealing for the examination timetabling problem. Ann. Operational Res., 63: 105-128.

Welsh, D.J.A. and M.B. Powell, 1976. The upper bound for the chromatic number of a graph and its application to timetabling problems. The Computer Journal, 11: 41-47.

White, G.M. and B.S. Xie, 2001. Examination Timetables and Tabu Search with Longer-Term Memory. In: Burke, E.K. and W. Erben (Eds.). Practice and Theory of Automated Timetabling: Selected Papers from the 3rd International Conference. Springer Lecture Notes in Computer Science, 2079: 85-103.