Intelligent Decision Technologies 7 (2013) 137–150
DOI 10.3233/IDT-130157
IOS Press

Evolutionary algorithms using cluster patterns for timetabling

Nandita Sharma*, Tom D. Gedeon and B. Sumudu U. Mendis
Information and Human Centred Computing Group, Research School of Computer Science, Australian National University, Canberra, ACT, Australia

Abstract. The examination timetabling problem (ETP) is a NP complete, combinatorial optimization problem. Intuitively, use of properties such as patterns or clusters in the data suggests possible improvements in the performance and quality of timetabling. This paper investigates whether the use of a genetic algorithm (GA) informed by patterns extracted from student timetable data to solve ETPs can produce better quality solutions. The data patterns were captured in clusters, which then were used to generate the initial population and evaluate fitness of individuals. The proposed techniques were compared with a traditional GA and popular techniques on widely used benchmark problems, and a local data set, the Australian National University (ANU) ETP, which was the motivating problem for this work. A formal definition of the ANU ETP is also proposed. Results show techniques using cluster patterns produced better results than the traditional GA with statistical significance of $p < 0.01$, showing strong evidence. Our techniques either clearly outperformed or performed well compared to the best known techniques in the literature and produced a better timetable than the manually constructed timetable used by ANU, both in terms of quality and execution time. In this work, we also propose clear criteria for specifying the top results in this area.

Keywords: Timetabling, clusters, constraints, data patterns, genetic algorithms

1. Introduction

Scheduling is a common real world problem that arises in many real world domains such as education [**?**], sport [**?**,**?**], nursing [**?**], and transportation [**?**, **?**]. The aim in scheduling is to assign a set of entities to a set of resources satisfying a given set of constraints. Scheduling algorithms have received much research interest because of their complexity and real world usefulness. The most widely studied instance of the problem is in the academic domain – universities and high schools regularly need to schedule classes and exams. This instance of the scheduling problem is commonly called *timetabling*.

The *exam timetabling problem* (ETP) is a constraint satisfaction problem where a solution has to satisfy a set of given constraints. It requires allocating exams to timeslots based on *hard* (required) and *soft* (desirable) constraints. Hard constraints govern the feasibility of a timetable whereas soft constraints define the quality and acceptability of the timetable.

To build a university exam timetable, suppose 500 exams must be scheduled to 28 timeslots with constraints that a student can only sit one exam at a time, a timeslot has resources for up to 1500 students, and the number of consecutive exams that a student takes within a day must be minimized. There are 28^{500} possible ways of allocating the exams to the timeslots. An exhaustive search by generating and testing solutions that satisfies all constraints will be very costly (exponential in time). A large proportion of these possibilities will not produce feasible timetables. Moreover, underconstrained ETPs could result in combinatorial solution spaces whereas overconstrained situations could lead to no solution. An approach is to search for promising subspaces for solutions and detect overconstrained situations as soon as possible in the search process.

Currently, most universities develop their timetables largely manually with assistance of some timetabling

*Corresponding author: Nandita Sharma, Centre of Information and Human Centred Computing, Research School of Computer Science, Building 108, Australian National University, Canberra, ACT 0200, Australia. Tel.: +61 2 6125 9664; E-mail: nandita.sharma@anu.edu.au.

ISSN 1872-4981/13/$27.50

support tools. It takes them up to a week typically as is the case at the Australian National University (ANU). Once developed, these timetables are difficult to revise even for simple additional constraints and any what-if analysis is almost impossible due to time pressures. The timetabling support tools do not in general support backtracking. Better techniques are required to exploit the advantages of computers and the latest research on finding approximate solutions to such intractable problems. The computer algorithms available need improvement for the efficiency of generation, provision of what-if analysis, and incremental construction and revision of solutions.

Over the past several decades, the timetabling problem has gained much research attention and a variety of computer based approaches have been developed. Recent techniques include:

- Local search – The search starts in a given timetable state, which progresses through its neighborhood by moving from one state to another in the form of swapping or allocating timeslots for exams guided by a cost function to achieve a higher quality timetable [**?**,**?**].
- Constraint programming – Exams, modeled as a set of variables, must be assigned to resources, which are modeled as values (e.g. timeslots and rooms), using defined rules to satisfy constraints [**?**].
- Graph-based – An ETP can be represented as a graph where vertices and edges represent exams and common students respectively. An ETP is usually reduced to a known graph problem e.g. graph coloring problem or an identifying clique problem and then suitable heuristics are applied to solve the problem [**?**,**?**].
- Clustering – Exams that satisfy a particular hard constraint are put into groups, which are allocated to timeslots to satisfy soft constraints [**?**].

A technique that has gained wide attention in the area of exam timetabling is evolutionary algorithms (EAs) [**?**,**?**,**?**,**?**,**?**]. EAs try to maintain diversity in a population of candidate solutions with the aim to achieve global optimal solutions by applying crossover and mutation operations and selecting better candidate solutions throughout the search. Various aspects of a EA (e.g. representation [**?**], variation operations [**?**,**?**, **?**]) have been adapted to improve the quality of the resulting timetables.

The work in this paper investigates and proposes a novel approach in improving EAs for timetabling. It investigates whether using patterns derived from timetabling data improves the search for good quality examination timetables and whether exploitation of the patterns improves search efficiency. Patterns from the student data will be extracted and captured in clusters. These clusters will be made up of exams, which will be used to:

1. Generate an initial population for a GA; and
2. Mark poor individuals in a population so their fitness values do not get calculated using the fitness functions, which usually require long computation times. However, in order to maintain population diversity, poor individuals will be given a low probability for selection by future populations.

The techniques we developed were compared with a traditional GA on benchmark data sets supplied by the University of Toronto, which are widely used in literature, and a data set for Semester 1 of 2009 supplied by the ANU Timetabling Centre. Our results were compared to the best solutions in literature using the Toronto benchmark data sets.

This paper provides a description of an exam timetabling problem (ETP) that is commonly used in literature, introduces a real-world ETP and presents a definition for the ANU ETP. It moves on to present techniques for capturing information in *clusters* from exam data sets. Then it presents novel hybrid genetic algorithm search algorithms using clusters to generate and find timetables for an ETP. The proposed algorithms were implemented and applied to benchmark ETPs and the ANU ETP and the results along with analyses are provided. A method for ranking techniques for solving ETPs is also presented. The paper concludes with a summary of the work and provides suggestions for future work.

2. The examination timetabling problem

An ETP is defined as allocating exams to ordered timeslots satisfying all *hard* constraints and maximally satisfying all *soft* constraints. The challenges and complexity arise due to the variety of constraints. The kinds and nature (hard or soft) of the constraints may vary depending on the aims, needs and wants of universities, staff and students.

A feasible (*admissible*) timetable satisfies all hard constraints. It is widely accepted that the primary hard constraints are:

- A student must not be assigned to do more than one exam in a timeslot i.e. no clashes; and

– The number of students sitting an exam must not exceed the capacity of the room.

All soft constraints may not need to be satisfied, but the quality of the timetable can be determined by how well the timetable satisfies these constraints. The quality of a timetable is based on the number of soft constraints satisfied. The primary soft constraints are:

– A student should not be assigned to sit two consecutive exams in a day;
– Spread the exams out as evenly as possible for each student;
– Exams with a large number of students should be allocated towards the beginning of the exam period; and
– All exams should be scheduled within a minimum number of timeslots.

Sometimes the use of too many hard constraints may not give feasible timetables. In these circumstances, some hard constraints could be turned into soft constraints with high penalty values.

Universities define their ETPs using different combinations of hard and soft constraints. Universities of Toronto, Nottingham and Melbourne have defined various forms of ETPs. This paper uses the University of Toronto ETP (defined in [**?**]), which has been widely used in literature, to show results and make comparisons of methods proposed in this paper. A definition of the ANU ETP is also proposed. Techniques proposed in this paper solve the Toronto and ANU ETPs.

2.1. ANU ETP

Let
$E = \{e_1, e_2, e_3, \ldots, e_n\}$ i.e. set of n exams
$T = [t_1, t_2, t_3, \ldots, t_m]$ i.e. list of m timeslots
$S =$ total number of students

$$a_{it} = \begin{cases} 1 \text{ exam } i \text{ is assigned to timeslot } t \\ 0 \text{ otherwise} \end{cases}$$

c_{ij} = number of students scheduled to do exam i and exam j
s_i = number of students scheduled to do exam i
p_d = penalty value associated for two exams scheduled with d timeslots apart
p_{BSL} = penalty value for exams with high number of students that are not allocated an early timeslot

The ANU ETP is expressed as a number of constraints, modeled as follows:

The hard constraints are:

1. No clashes i.e. no students must have more than one exam scheduled in a timeslot

$$\sum_{i=1}^{|E|-1} \sum_{j=i+1}^{|E|} \sum_{t=1}^{|T|} a_{it} a_{jt} c_{ij} = 0$$

2. No more than 1500 students must be scheduled to sit an exam in any timeslot

$$\sum_{i=1}^{|E|} \sum_{t=1}^{|T|} a_{it} s_i \leqslant 1500$$

3. Schedule all exams within 28 timeslots

$$|T| = 28$$

4. An exam must be scheduled once

$$\sum_{i=1}^{|E|} \sum_{t=1}^{|T|} a_{it} = 1$$

The soft constraints are:

1. Spread the exams out as evenly as possible for students
For some penalty $p_d = p_{|t_i - t_j|}$, minimize

$$\sum_{i=1}^{|T|-1} \sum_{j=i+1}^{|T|} p_{|t_i - t_j|} c_{ij}$$

2. Exams required to be done by larger groups of students should be scheduled towards the beginning of the exam timetable
Minimize

$$\sum\nolimits_{\forall s_i | s_i > \textit{95th quantile of } \{s_j, \forall j \in E\}} \sum_{t=1}^{|T|} p_{BSL} a_{it} s_i$$

3. Capturing data patterns in clusters

The quality of a timetable is generally governed by the number of students that have exam conflicts. A student data set has students and their exams, which are used to determine exam conflicts in a timetable. Patterns in the student data set can be captured in *clusters*, a novel representation of defining a group of exams with common students. Three different types of clusters are proposed – *Cluster*, *Size-2 Cluster* and *Top-N Size 2 Cluster*.

Algorithm: Cluster Generation. Find clusters of exams and their weights based on exams common to students
Inputs:
- A collection of students
- A collection of exams
- Mappings between students to exams showing which student will sit which exam (or which exam will be done by which student)

Output:
- A collection of exam clusters and their weights

Method:
1. Find unique sets of exams (*SE*) common to at least one student from the student data set. The number of students common to SE_i will be the value of SE_i. Each SE_i must have at least two exams.

$$\text{value}(SE_i) \geqslant 1$$
$$|SE| \geqslant 2$$

where value(SE_i) = number of students required to sit all exams in SE_i
This step is realized through an adaptation of the Apriori algorithm [?].
2. Let $SSE = \{SE_1, SE_2, \ldots, SE_k\}$
Assign a weight to each set using the following definition:

$$\text{weight of cluster } = \sum\nolimits_{\forall se | se \subseteq SE \cap se \subseteq SSE} value(se)(|se| - 1)$$

3. Remove all proper subsets. This step does not change the weights allocated to clusters.

Fig. 1. The Cluster Generation algorithm generates exam clusters and their weights from raw student and exam data. It is a novel way to represent student and exam data.

Algorithm: Size-2 Cluster Generation. Find size-2 clusters and their weights based on exams common to students
Inputs:
- A collection of students
- A collection of exams
- Mappings between students to exams showing which student will sit which exam (or which exam will be done by which student)

Output:
- A collection of Size-2 clusters and their weights

Method:
From the student data set, find all unique sets of exams, *SE*, where all exams in the set, SE_i, are common to at least one student. The number of students common to SE_i will be the weight of SE_i. Each SE_i must have exactly two exams. For each set SE_i:

$$\text{weight}(SE_i) \geqslant 1$$
$$|SE_i| = 2$$

where weight(SE_i) = number of students required to sit all exams in SE_i
This step is realized through an adaptation of the Apriori algorithm [?].
Each SE_i is a Size-2 cluster.

Fig. 2. The Size-2 cluster Generation algorithm generates clusters that have only two exams. Each Size-2 cluster has an associated weight. Size-2 clusters are derived from raw student and exam data.

A cluster has an associated weight value and their definition and purpose depends on the type of cluster. Exams in clusters with higher weights are scheduled earlier in the initialization process of a GA with the aim of reducing exam conflicts. A Size-2 cluster is a variation of a cluster such that it only has two exams in a cluster. On the other hand, Top-N Size-2 clusters are used to determine poor chromosomes and are not used to generate the initial population. Top-N Size-2 clusters are a set that has N Size-2 clusters with the highest N weights in a set of Size-2 clusters.

Clusters are generated by finding unique sets of exams common to students. Each set will be made up of at least two exams, which is the minimum number of exams required to determine exam conflicts. Some of these sets are proper subsets of other sets and it will not be efficient to traverse and use all the sets for generating a population. The redundant sets are removed and their supersets, the clusters, have their weights defined by a linear combination consisting of the number of students common to sets and their subsets, and the cardinalities for the corresponding sets and their subsets. Higher weights were given to clusters with higher number of students common to exams in subsets and subsets with high cardinality. The algorithm for deriv-

Algorithm: Top-N Size-2 Cluster Generation. Find size-2 clusters and their weights such that weights of the clusters are among the higher band value

Inputs:

- A collection of students, S
- A collection of exams, E
- Mappings between students to exams showing which student will sit which exam (or which exam will be done by which student)
- Number of Size-2 clusters, N

Output:

- A collection of Size-2 clusters and their weights

Method:

1. Generate a collection of Size-2 clusters using algorithm presented in Fig. 2 using S and E, $S2C$
2. Sort $S2C$ in descending order of weight
3. Select $S2C_i$ such that $1 \leqslant i \leqslant N$ and i represents an index in $S2C$

Fig. 3. The Top-N Size-2 Cluster Generation algorithm generates Size-2 Clusters with weights fulfilling a certain weight criteria.

Algorithm: TradGA. Generate and find a timetable that satisfies the constraints of an ETP

Inputs:

- A collection of exams, E
- A collection of students, S
- A collection of exam timeslots, T
- Mappings between students to exams showing which student will sit which exam (or which exam will be done by which student), M
- Number of chromosomes in each population, *n_chroms*
- Search crossover rate, *cross_rate*
- Search mutation rate, *mut_rate*
- Number of generations, g
- Fitness function, *fit_funct*
- Chromosome selection procedure, *select_proc*

Output:

- A timetable which has a mapping between elements in E and elements in T

Method:

1. Initialize the population with *n_chroms* chromosomes by randomly allocating a timeslot in T to every exam in E in each chromosome
2. Evaluate the fitness values for each chromosome in the population using *fit_funct* and M
3. Repeat until g generations reached
4. Apply crossover to the population with crossover rate *cross_rate*
5. Apply mutation to the population with mutation rate *mut_rate*
6. Evaluate the fitness values for each new chromosome using *fit_funct* and M
7. Select chromosomes that will proceed to the population for the next generation using *select_proc*
8. End repeat
9. Select the chromosome with the best fitness value in the population, *best_chrom*
10. Generate a timetable represented by *best_chrom*

Fig. 4. The TradGA algorithm is the traditional GA that was extended to include the use of clusters and size-2 clusters in algorithms presented in Fig. 5 to search for a timetable that maximally satisfies ETP constraints.

ing clusters is presented in Fig. 1 along with a mathematical definition for cluster weight.

Note that values of the proper subsets of supersets given above are no less than the values of the supersets because of the Apriori property [**?**]. As a result, if sets are chosen in descending order of weight, then only sets of size two are needed to get a union of all sets and span all exams in all sets, motivating Size-2 clusters.

A Size-2 cluster is made up of only two exams. The weight of a Size-2 cluster represents the number of students common to the two exams. As a result, a Size-2 cluster shows which two exams will cause a clash and the number of students affected when scheduled in the same timeslot (similarly for conflicts). Figure 2 presents the process for developing Size-2 clusters.

Size-2 clusters with the highest N weights form Top-N Size-2 clusters. The process for deriving Top-N Size-2 clusters is presented in Fig. 3.

4. Use of cluster patterns in evolutionary algorithms

The idea behind capturing patterns in student data by clusters is that exams common to students can be represented and exploited in search – intuitively, improving the search. Rather than generating the initial population of a GA randomly and using the selection phase to narrow the search space, using the domain knowl-

Algorithm: ClusterGA/Size2-CGA. Generate and find a timetable that uses a GA search informed by Clusters (or Size-2 Clusters) to generate the initial population

Inputs:

- A collection of exams, E
- A collection of students, S
- A collection of exam timeslots, T
- Mappings between students to exams showing which student will sit which exam (or which exam will be done by which student), M
- Number of chromosomes in each population, *n_chroms*
- Search crossover rate, *cross_rate*
- Search mutation rate, *mut_rate*
- Number of generations, g
- Fitness function, *fit_funct*
- Chromosome selection procedure, *select_proc*
- **Preferred minimum number of timeslots between any two exams in the same cluster (or size-2 cluster), d**

Output:

- A timetable which has a mapping between elements in E and elements in T

Method:

1. Generate clusters/size-2 clusters (for clusters, refer to Fig. 1 or for size-2 clusters, refer to Fig. 2)
2. Put the clusters (or size-2 clusters) in a sorted cluster collection C sorted by descending order of weight
3. Initialize the population with *n_chroms* chromosomes **using clusters (or size-2 clusters) as presented in sub-steps 3.1–3.14**
3.1 For each chromosome *chrom* in the population {
3.2 Initialize an empty *E* to *T* mapping record for *chrom* which shows which exams in *E* are allocated to which timeslots in *T* (or vice-versa) in *chrom ET_mapping_chrom*
3.3 For each cluster (or size-2 cluster) *c* and corresponding weight *w* in C where *c* does not have all its exams in *ET_mapping_chrom*
3.4 Set the minimum number of timeslots between two exams $m = d$
3.5 For each exam e in c and e not in *ET_mapping_chrom*
3.6 While e is not in *ET_mapping_chrom* {
3.7 Randomly select a timeslot t from T
3.8 If t is not at least m timeslots away from all the other exams in c using *ET_mapping_chrom*
3.9 Set $m = m - 1$
3.10 If $m < 0$ {
3.11 Add a mapping e to t in *ET_mapping_chrom* }}
3.12 Else {
3.13 Add a mapping e to t in *ET_mapping_chrom* }}
3.14 Allocate all exams in *E* and not in *ET_mapping_chrom* to randomly selected timeslots in *T* and record the allocations in *ET_mapping_chrom*}
4. Evaluate the fitness values for each chromosome in the population using *fit_funct* and M
5. Repeat until g generations reached
6. Apply crossover to the population with crossover rate *cross_rate*
7. Apply mutation to the population with mutation rate *mut_rate*
8. Evaluate the fitness values for each new chromosome using *fit_funct* and M
9. Select chromosomes that will proceed to the population for the next generation using *select_proc*
10. End repeat
11. Select the chromosome with the best fitness value in the population, *best_chrom*
12. Generate a timetable represented by *best_chrom*

Fig. 5. Algorithms for ClusterGA and Size2-CGA – novel algorithms we propose for exam timetabling. They are extensions to a conventional GA for timetabling (TradGA, which was presented in Fig. 4) that uses clusters (or size-2 clusters) to search for a timetable that maximally satisfies ETP constraints. The differences in the algorithms from TradGA are shown with lines in bold.

edge from the beginning of the search may improve the quality of the search.

This paper proposes three EA techniques, *ClusterGA*, *Size2-CGA* and *TopN-CGA*, based on the traditional GA, to solve the ETPs. Each proposed technique uses clusters derived from domain knowledge with different interpretations. The differences from the traditional GA for these techniques are shown with steps in bold, being the extensions to a standard or traditional GA approach (TradGA), in Algorithms 5 and 6.

TradGA: The initial population is generated randomly where every exam in each chromosome is allocated to a valid timeslot. In addition, the fitness value for each chromosome is calculated using raw student data and not Top-N Size-2 clusters, which is used by TopN-CGA.

ClusterGA/Size2-CGA: The purpose of the techniques was to determine whether cluster patterns used in the initialization process of a GA improve the quality of solutions. ClusterGA/Size2-CGA use clusters/size-

Algorithm: TopN-CGA. Generate and find a timetable that uses a GA search informed by Top-N Size-2 Clusters
Inputs:
- A collection of exams, E
- A collection of students, S
- A collection of exam timeslots, T
- Mappings between students to exams showing which student will sit which exam (or which exam will be done by which student), M
- Number of chromosomes in each population, *n_chroms*
- Search crossover rate, *cross_rate*
- Search mutation rate, *mut_rate*
- Number of generations, g
- Fitness function based on the ETP, *etp_fit_funct*
- Chromosome selection procedure, *select_proc*
- **Number of Size-2 clusters, N**

Output:
- A timetable which has a mapping between elements in E and elements in T

Method:
1. **Generate an ordered collection of Top-N Size-2 Clusters using N sorted in descending order of weight (presented in Fig. 3) *TopN-S2C***
2. **Define the fitness evaluation phase so that TopN-S2C is used to determine poor chromosomes in the search *hybrid_fit_funct*. Chromosomes found to be poor will not have their fitness values calculated using the fitness function but will be given a significantly low fitness value, approximately 0 so that the probability is low for the chromosomes to be selected for future generations. Chromosomes that are not poor will have their fitness values calculated in the same way as TradGA using *etp_fit_funct*. *hybrid_fit_funct* is defined by sub-steps 2.1–2.7**

2.1 **For each chromosome chrom in the population {**
2.2 **For each Top-N Size-2 cluster c in TopN-S2C selected in descending order of weight {**
2.3 **If all exams in c are allocated to the same timeslot in *chrom* {**
2.4 **Record (or mark) *chrom* as poor**
2.5 **Assign a low fitness value to *chrom* which is essentially 0 }}**
2.6 **If *chrom* is not recorded as poor {**
2.7 **Calculate the fitness value of *chrom* using *etp_fit_funct* and M }}**

3. Initialize the population with *n_chroms* chromosomes by randomly allocating a timeslot in T to every exam in E in each chromosome
4. Evaluate the fitness values for each chromosome in the population using *hybrid_fit_funct* and M
5. Repeat until g generations reached
6. apply Crossover to the population with crossover rate *cross_rate*
7. apply Mutation to the population with mutation rate *mut_rate*
8. Evaluate the fitness values for each new chromosome using *hybrid_fit_funct* and M
9. Select chromosomes that will proceed to the population for the next generation using *select_proc*
10. End repeat
11. Select the chromosome with the best fitness value in the population, *best_chrom*
12. Generate a timetable represented by *best_chrom*

Fig. 6. Algorithm for TopN-CGA, another novel algorithm we propose for searching an exam timetable for an ETP. The algorithm extends the conventional GA for timetabling (TradGA, which was presented in Fig. 4) by using Top-N Size-2 Clusters. The differences in the algorithm from TradGA are shown with lines in bold.

2 clusters and their associated weights (defined in Algorithms 1 and 2) to generate the initial population for the GA. Exams in clusters with higher weights get scheduled before exams in clusters with lower weights i.e. exams with more conflicts are allocated timeslots earlier in the initialization process. The initialization process tries to randomly allocate timeslots at least d timeslots away from exams in the same cluster. d is decreased until all exams in a cluster are scheduled. For experiments done in this paper, d was set to 0.

ClusterGA and Size2-CGA use a greedy approach to select clusters in the initialization process by choosing clusters in descending order of weight, but like TradGA, there is still some degree of randomness. Exams are randomly chosen from a cluster and then randomly allocated to a timeslot while minimizing some degree of conflict between exams in the same cluster.

TopN-CGA: Usually fitness values are evaluated by comparing each exam in a timetable to other exams for common students and some fitness functions (e.g. ANU ETP) require complex calculations. These can be computationally costly and the reason for TopN-CGA is to reduce this cost by bypassing these calculations if the chromosome is *poor*. A chromosome is *poor* if all exams in a Top-N Size-2 cluster are scheduled in the same timeslot. The chromosome is given a low probability (or a significantly low fitness value) for proceeding to future generations, or *marked*, and the fitness function is not used to evaluate their fitness. However, if the chromosome is not poor then the fitness function

is used to evaluate the fitness value.

TopN-CGA technique was used to determine whether it reduces the overall execution time of a GA and still give good quality solutions. Execution times for solving ETPs have not been religiously reported in literature, but it is not new [**?**].

TradGA and the proposed techniques, ClusterGA, Size2-CGA and TopN-CGA, were used to solve the Toronto ETP using the benchmark data sets provided by the University of Toronto, as well as the motivating problem for this research, the ANU ETP using an ANU data set. The ANU data set is named ANU-Exam-Data, and has 382 exams and 9318 students. Each technique was run 5 times on a particular ETP and data set.

GA parameters were set after some experimental testing. The setting chosen helped the techniques achieve good quality solutions when compared to state-of-the-art techniques used to solve ETPs. Future work will include testing the proposed techniques using a range of values for GA parameters and optimizing them.

4.1. Representation

A chromosome represented an exam timetable. Loci of chromosomes represented exams and their alleles represented timeslots. This ensured that all exams were scheduled in a timetable, and only once, thus satisfying a hard constraint (ANU ETP Hard Constraint 4) without the need for superfluous calculations or extra computational time during the search.

4.2. Fitness function

Fitness functions, modeled as cost functions, varied depending on the definition of the ETP and the type of data set. The cost function for the Toronto ETP is presented in [**?**]. The cost function for the ANU ETP was made up of both hard and soft constraints to ensure that the EA search improves the quality of candidate solutions. Consider an ETP that has no clashes as hard constraints. Also, consider a population, which has all infeasible timetables and that there is no way to differentiate timetables that more clashes than less clashes. Future generations may not necessarily give feasible timetables and means that all the timetables in the population are regarded as equal i.e. they are all infeasible. As a consequence, the search may not converge. An example of such a case is when the ETP is overconstrained and few feasible solutions exist. Modeling hard constraints as strong soft constraints may not give feasible solutions but help the search to converge to candidate solutions with lower cost values and hopefully converge to feasible candidate solutions.

The cost function for the ANU ETP was a mathematical sum of all the constraints except Hard Constraint 4, which was always satisfied because of the chosen representation. Penalty values for hard constraints were chosen to be significantly larger so that if the hard constraints were violated in a candidate timetable solution then the candidate solution would be given a significantly large cost value. In turn, this would direct the GA search towards candidate solutions that did not violate the hard constraints. The penalty values for the ANU ETP (presented in Table 1) were chosen experimentally with the guidance of the ANU Timetabling Centre, number of students in ANU-Exam-Data and the ratio for penalty values used in the literature. If the penalty values for the hard constraints are set too high, then the penalty values for the soft constraints will become negligible which increases the risk of the GA search to produce a timetable that may have a high number of soft constraint violations. On the other hand, if the penalty values for the hard constraints are set too low, then more hard constraint violations could result in the final timetable produced by the GA search.

Evaluating cost values for each chromosome in the population takes a significant proportion of the total search execution time [**?**]. The cost function for the ANU ETP is much more complex, therefore, would require more computation time. TopN-CGA used information from patterns in timetabling data to mark poor performing chromosomes so their cost values were not calculated using the cost function. The algorithm for TopN-CGA was present in Fig. 6. The performance for TopN-CGA was compared with TradGA.

5. Results and discussion

The proposed techniques, ClusterGA, Size2-CGA and TopN-CGA, produced better quality solutions than TradGA. More specifically, the quality of the timetables produced by ClusterGA and Size2-CGA were better than TradGA, and TopN-CGA produced solutions with lower execution times than TradGA without compromising quality.

Firstly, results for ClusterGA and Size2-CGA are presented on the University of Toronto ETP. Table 2 below summarizes performances for TradGA, ClusterGA and Size2-CGA on benchmark data sets. Clus-

Table 1
Empirical penalty values used in cost function for ANU ETP

Constraint violation	Penalty value
0 order conflict (clash)	5,000
Number of timeslots exceeds the maximum number of timeslots	5,000
Number of students scheduled to sit exams in a timeslot exceeds seating capacity	5,000
1st order conflict (exams in consecutive timeslots for a student)	10
2nd order conflict	4
3rd order conflict	2
4th order conflict	1
Big exams not scheduled in earlier timeslots	4

Table 2
Cost values for ClusterGA, Size2-CGA and TradGA on benchmark data sets

Dataset	Lowest cost for techniques implemented		
	TradGA	ClusterGA	Size2-CGA
CAR91	5.80	**5.56**	5.75
CAR92	4.70	**4.56**	4.68
EAR83	40.12	**38.75**	39.10
HEC92	10.62	**9.61**	9.79
KFU93	15.14	**14.34**	14.52
LSE91	11.60	**11.04**	11.23
PUR93	7.45	7.35	**7.30**
RYE92	9.14	**8.79**	8.93
STA83	159.35	**158.19**	157.29
TRE92	8.65	**8.44**	8.45
UTA92	3.78	3.72	**3.68**
UTE92	25.49	**24.31**	24.41
YOR83	39.80	**38.07**	38.58

The probability associated with Student's t-Test for ClusterGA and TradGA is $p < 0.01$.
The probability associated with Student's t-Test for Size2-CGA and TradGA is $p < 0.01$.

terGA and Size2-CGA performed better than TradGA. They generated better quality solutions and after most generations, they also generated better candidate solutions. However, ClusterGA performed the best on most data sets. According to the Student's t-Test, the performances for ClusterGA and Size2-CGA are significantly different from TradGA.

An example of how ClusterGA and Size2-CGA were compared to TradGA on a data set is presented below. Figures 7 and 8 show graphs for cost values for the techniques on the EAR83 and HEC92 data sets respectively. The graphs show that cost values for TradGA converge to a value higher than ClusterGA and Size2-CGA. Similarly, this was the case for all the data sets.

ClusterGA performed the best on the EAR83 data set after most of the generations (or iterations), with Size2-CGA eventually closing in on similar quality solutions. TradGA algorithm was clearly worse.

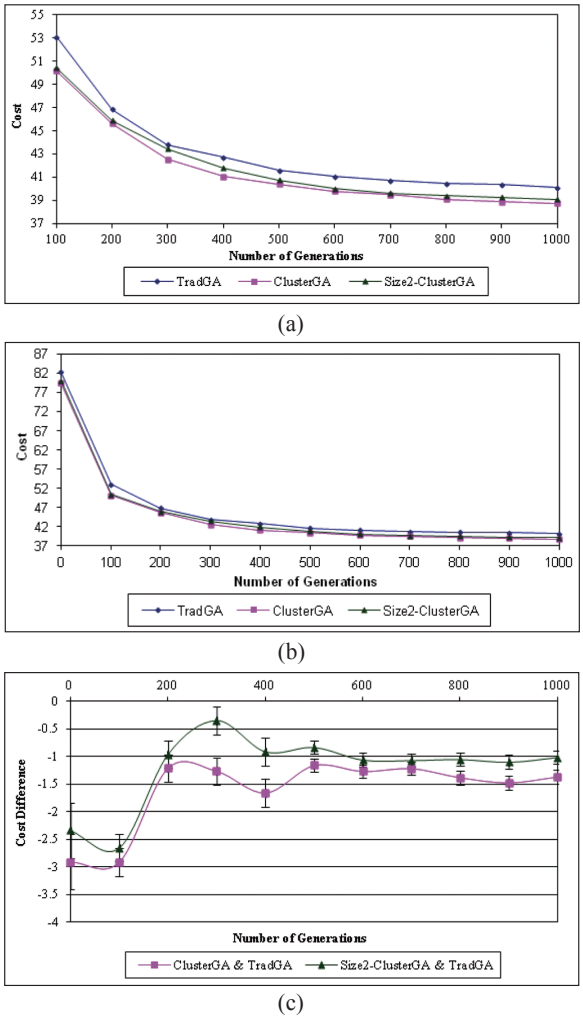
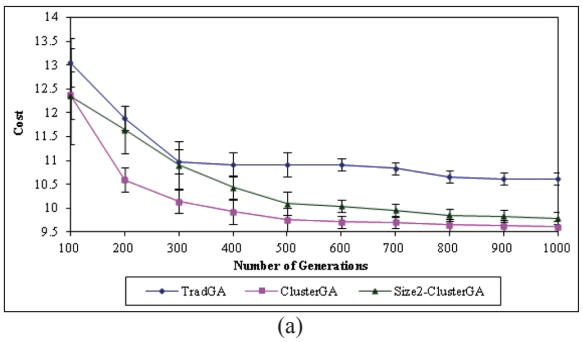


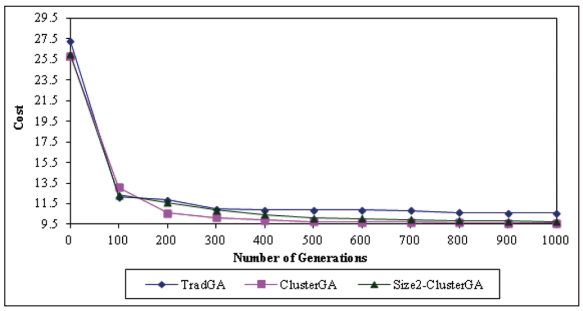
Fig. 7. Mean cost values achieved by TradGA, ClusterGA and Size2-CGA after various generations on EAR83 data set. Both (a) and (b) display the same graph, but (a) provides a zoomed in view of the discrepancies in performance for the techniques. (a) Mean cost values from 100 generations. (b) Mean cost values from 0 generations (c) Difference in mean cost values for EAR83 (cost difference of ClusterGA & TradGA = ClusterGA − TradGA, cost difference of Size2-CGA & TradGA = Size2-CGA − TradGA). (Colours are visible in the online version of the article; http://dx.doi.org/10.3233/IDT-130157)

Figure 8 shows that TradGA performed the worst with significant high cost values after the majority of the generations for the HEC92 data set. The ClusterGA and Size2-CGA algorithms converge on similar good quality solutions, better than TradGA.

ClusterGA and Size2-CGA performed better than TradGA because it involved a type of greedy approach where exams in clusters with higher weights were scheduled earlier in the initialization phase for Clus-



(a)



(b)

Fig. 8. Mean cost values achieved by TradGA, ClusterGA and Size2-CGA after various generations on HEC92 data set. Both (a) and (b) display the same graph, but (a) provides a zoomed in view of the discrepancies in performance for the techniques. (a) Mean cost values from 100 generations. (b) Mean cost values from 0 generations. (Colours are visible in the online version of the article; http://dx.doi.org/10.3233/IDT-130157)

terGA and Size2-CGA. As a result, exams with high numbers of common students were scheduled early with a greater probability that these exams were not scheduled in the same timeslot.

Table 2 showed that ClusterGA performed better than Size2-CGA on most of the data sets. The sizes of clusters in ClusterGA were not restricted as for Size-2 clusters. The sizes of clusters were usually greater than two, which was the size of all Size-2 clusters. As a consequence, ClusterGA scheduled more exams at a time in different timeslots than just two exams from Size-2 clusters.

The largest size for a cluster is equal to the maximum number of exams a student has to sit. Table 3 shows the size and the number of clusters and Table 4 shows the weights for the clusters derived from the EAR83 data set. A cluster with 6 exams was used first to schedule exams in a timetable in the initial population in ClusterGA. The exams were scheduled so that it minimized the number of conflicts with other exams in the same cluster. Alternatively, Size2-CGA used a Size-2 cluster with 2 exams to schedule exams

Table 3
Exam sets of different sizes EAR83

Number of exams in a set	Number of sets
2	4,793
3	19,739
4	32,599
5	30,444
6	17,598

Table 4
Characteristics of clusters for EAR83

Size of cluster	Number of clusters	Minimum weight	Maximum weight	Average weight
2	1	1	1	1
3	0	–	–	–
4	12	10,650	28,662	19,785.5
5	51	15,786,730	117,230,140	4.46E+07
6	17,598	3.07E+11	1.27E+13	1.45E+12

Table 5
Characteristics of size-2 clusters for EAR83

Number of clusters	Minimum weight	Maximum weight	Average weight
4,793	1	192	5.42

in a timetable in the initial population. Size-2 clusters for the EAR82 data set is presented in Table 5. In this case, only 2 exams were scheduled so that there were no conflicts between them. With Size2-CGA it is possible that 6 exams that will be in conflict as recorded in the corresponding cluster (without the size limit) could be scheduled amongst 2 timeslots. However with ClusterGA, there is a higher chance that the 6 exams will not be scheduled in common timeslots. As a result, most chromosomes in the initial population would have had this characteristic, which enabled the algorithm to generate better quality solutions and the reason that ClusterGA performed better than Size2-CGA.

ClusterGA and Size2-CGA performed well against techniques presented in literature. Table 6 shows the cost values of ClusterGA, Size2-CGA and successful techniques reported in literature, which were used to solve the University of Toronto ETP on benchmark data sets. For the UTE92 data set, ClusterGA produced a timetable with a lower cost value than the lowest cost value reported in literature. ClusterGA performed better than the average cost value reported in literature on data sets, CAR92, HEC92, KFU93, LSE91, RYE92, STA83, TRE92, UTE92 and YOR83. On the other hand, Size2-CGA performed better than the average on HEC92, KFU93, LSE91, RYE92, STA83, TRE92 and UTE92. For data sets that ClusterGA and Size2-CGA did not produce cost values lower than the average, the cost values were within one standard deviation of the average.

Table 6
Cost values for ClusterGA, Size2-CGA and techniques reported in literature on benchmark data sets

Techniques	Toronto data set											
	CAR91	CAR92	EAR83	HEC92	KFU93	LSE91	RYE92	STA83	TRE92	UTA92	UTE92	YOR83
ClusterGA	5.5	4.5	38.7	9.6	14.3	11.0	8.7	158.1	8.4	3.7	**24.3**	38.0
Size2-CGA	5.7	4.6	39.1	9.7	14.5	11.2	8.9	157.2	8.4	3.6	24.4	38.5
Average (Literature)	5.1	4.6	36.8	11.2	14.6	11.5	9.1	159.2	8.7	3.5	26.7	38.7
Standard Deviation (Literature)	0.82	0.66	3.43	0.75	1.43	1.58	2.38	3.63	0.62	0.38	1.80	2.16
Caramia et al. (2001) [?]	6.6	6.0	**29.3**	**9.2**	13.8	**9.6**	6.8	158.2	9.4	3.5	**24.4**	**36.2**
Gaspero and Schaerf (2001) [?]	6.2	5.2	45.7	12.4	18.0	15.5	–	160.8	10.0	4.2	27.8	41.0
Paquete and Stutzle (2002) [?]	–	–	38.9	11.2	16.5	13.2	–	168.3	9.3	–	29.0	38.9
Gaspero (2002) [?]	5.7	–	39.4	10.9	–	12.6	–	157.4	–	4.1	–	39.7
Burke and Newall (2003) [?]	4.7	4.1	37.1	11.5	13.9	10.8	–	168.7	8.4	3.2	25.8	37.3
Merlot et al. (2003) [?]	5.1	4.3	35.1	10.6	13.5	10.5	–	157.3	8.4	3.5	25.1	37.4
Burke and Newall (2004) [?]	5.0	4.3	36.2	11.6	15.0	11.0	–	161.9	8.4	3.4	27.4	40.8
Asmuni et al. (2005) [?]	5.2	4.5	36.6	11.6	15.3	11.4	10.1	160.8	8.5	3.5	27.6	39.8
Petrovic et al. (2005) [?]	4.5	3.9	33.7	10.8	13.8	10.4	8.5	158.4	7.9	**3.1**	25.4	36.4
Eley (2007) [?]	5.2	4.3	36.8	11.1	14.5	11.3	9.8	157.3	8.6	3.5	26.4	39.4
Djannaty and Mirzaei (2008) [?]	4.4	5.3	36.8	12.1	15.0	11.3	8.3	158.2	8.5	3.5	27.1	39.1
Den and Poli (2009) [?]	5.2	4.2	37.2	12.0	14.8	11.2	–	158.6	8.7	3.4	–	40.1
Burke et al. (2010) [?]	5.2	4.3	36.4	–	13.9	11.2	–	156.9	8.5	3.4	25.2	36.8
Pillay and Banzhaf (2008) [?]	4.2	4.9	35.9	11.5	14.4	10.9	9.3	157.8	8.4	3.4	27.2	39.3
Rahman et al. (2009) [?]	4.4	5.1	38.4	11.6	14.7	11.7	9.5	157.7	8.8	3.6	26.6	40.5
Abdullah et al. (2010) [?]	4.4	**3.8**	33.8	10.3	**12.9**	10.2	–	156.0	8.2	3.2	25.4	36.4
Abounacer et al. (2010) [?]	6.9	5.9	42.4	11.0	17.3	14.9	14.0	**155.7**	9.9	4.5	32.0	44.1
Turabieh and Abdullah (2011) [?]	4.8	4.1	34.9	10.7	13.0	10.0	9.7	158.3	**7.9**	3.2	26.1	36.2
Sabar et al. (2011) [?]	**3.9**	4.8	34.7	10.7	13.0	10.0	**4.8**	157.0	**7.9**	**3.1**	25.9	**36.2**

Table 7
Qualities of the best timetables generated by the different methods

Techniques	Quality measure	
	Number of clashes	Number of students with 2 exams scheduled in 1 day
ANU timetabling centre	0	346
TradGA	2	224
ClusterGA	0	214
Size2-CGA	1	287
TopN-CGA	2	169

Hybrid EA techniques, such as memetic algorithms, have been reported in literature to perform better than traditional EAs. We have seen that GAs informed by clusters to generate the initial population helps produce better quality solutions. The next step would be to incorporate a local search technique into our ClusterGA algorithm. Intuitively, a memetic algorithm should also perform better when informed by clusters.

When compared on the ANU ETP, ClusterGA produced the best timetable. ClusterGA and Size2-CGA performed significantly better than TradGA. Also, Size2-CGA produced timetables relatively similar to TradGA, which was preferred. Table 7 shows the quality of the timetables generated by the ANU Timetabling Centre and TradGA, ClusterGA, Size2-CGA and TopN-CGA. The ANU Timetabling Centre requires several days to produce a timetable whereas the EA techniques require several minutes and not only produces a single timetable but a population of timetables from which a good quality timetable can be chosen as the final timetable.

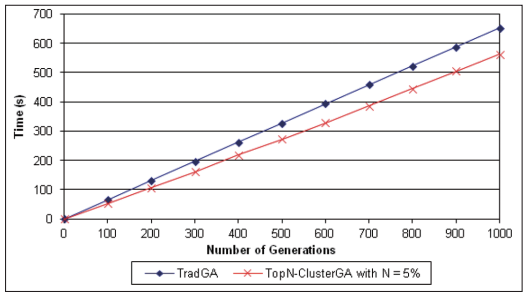


Fig. 9. Execution times for TopN-CGA and TradGA after various generations on ANU-Exam-Data. (Colours are visible in the online version of the article; http://dx.doi.org/10.3233/IDT-130157)

TopN-CGA had a lower overall execution time than TradGA. The low execution time for TopN-CGA is obvious when the complexity for evaluating fitness values using the fitness function is high e.g. ANU ETP. TopN-CGA required less computation for poor chromosomes than TradGA. Consequently, the total execution time for TopN-CGA was lower than TradGA. The execution times for TopN-CGA and TradGA on the ANU ETP are presented in Fig. 9. The cost values for the solutions produced by TopN-CGA were approximately equal to TradGA as shown in Fig. 10.

For TopN-CGA, N must be chosen so that not all chromosomes in the population are poor. All chromo-

Table 8
Ranking of Results in Table 6

Techniques	Rank	Number of data sets		
		Best result	Better than the mean in remaining data sets	Within 1 S.D. of mean in the rest
(Sabar, Ayob, Kendall, and Qu, 2011)	1	5	6	1
(Caramia, Dell'Olmo, and Italiano, 2001)	2	4	5	0
(Petrovic, Yang, and Dror, 2005)	3	2	10	0
(Turabieh and Abdullah, 2011)	4	2	9	1
(Abdullah, et al., 2010)	4	2	9	0
ClusterGA	6	1	8	3
(Abounacer, Boukachour, Dkhissi, and Alaoui, 2010)	7	1	1	0
(Merlot, Boland, Hughes, and Stuckey, 2003)	8	0	10	1
(Eley, 2006)	9	0	9	3
Size2-CGA	9	0	9	3
(E.K. Burke, Eckersley, McCollum, Petrovic, and Qu, 2010)	9	0	9	1
(EK Burke and Newall, 2003)	12	0	8	2
(Pillay and Banzhaf, 2008)	13	0	7	5
(Djannaty and Mirzaei, 2008)	13	0	7	4
(E.K. Burke and Newall, 2004)	15	0	6	5

somes in a population could be poor if N is high. If all chromosomes in a population are poor, then they all will be marked and will not have their fitness values calculated. As a result, no chromosome will be better than any other chromosome in a population, which could result in the next generation not performing better than the generation at hand. N was chosen for ANU-Exam-Data from some experiments. Future work could include optimizing N.

The rate of change for execution times for TopN-CGA was constant as shown in the graph in Fig. 9. This could have been due to the fact that poor chromosomes were in the population earlier in the search, i.e. when the number of generations was low, and there were hardly any or no poor chromosomes in the populations in later generations. This led the search to evaluate fitness values for all chromosomes in all populations generated later in the search. Future research could include increasing the value for N as the search evolves.

The average execution time for the technique used by the ANU Timetabling Centre took several days, which included five days for generating the exam timetable and time for revision. They largely developed the timetable manually with the assistance of some timetabling support tools. On the other hand, TradGA, ClusterGA and Size2-CGA took approximately 11 minutes on a consumer quality laptop. On the other hand, TopN-CGA took the least amount of time of approximately 9 minutes. The difference between the execution times was not very significant. Future work could investigate whether increasing N as the search progresses reduces the overall search time.

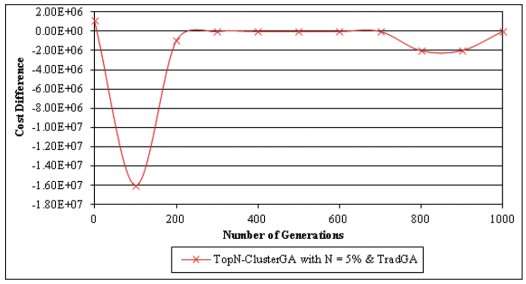


Fig. 10. Difference in cost values for TopN-CGA and TradGA on ANU-Exam-Data (cost difference = cost of TopN-CGA − cost of TradGA. (Colours are visible in the online version of the article; http://dx.doi.org/10.3233/IDT-130157)

6. Ranking the published results

We now propose a more stringent test for ranking the 'best solutions in the literature' as follows: the paper must report results on the Toronto benchmark data sets which are in order of importance:

i the best published result on at least one data set;
ii must be better than average of the top techniques for at least half of the data sets; and
iii must be no more than 1 standard deviation from the mean on the remaining data sets.

If we apply these criteria, we also derive a ranking for the reported results. This is shown in Table 8 for the papers in Table 6.

7. Conclusion

EAs informed by cluster patterns were used to generate initial populations and mark poor individuals in

the search so their fitness values were not evaluated using complex fitness functions. Results showed using clusters to generate initial populations for GAs improved the quality of solutions. It also showed GAs using clusters performed better than a traditional GA on all benchmark data sets (archived at the University of Toronto) and on the Australian National University data set. The quality of the solutions produced by the new techniques compare well against all the state-of-the-art techniques by improving the performance and quality of solutions in all the cases generally. Intuitively, improved EAs and hybrid algorithms to the GA could perform better than their current performance when informed by cluster patterns.

In summary, the results from the techniques proposed were:

1. Better than the best reported in literature for one benchmark data set – UTE92;
2. Better than the average of the top techniques over the last decade for 8 out of 12 benchmark data sets – CAR92, HEC92, KFU93, LSE91, RYE92, STA83, TRE92, YOR83; and
3. Within 1 standard deviation of the mean of the top techniques over the last decade for all the 3 remaining benchmark data sets – CAR91, EAR83, UTA92.

This paper also demonstrated that marking poor individuals in a population based on cluster patterns required less execution time, without compromising quality, compared to a traditional GA, which calculates the fitness value for every individual in a population using the fitness function. However, the difference between the execution times was two minutes, which was not a great difference. Future work would involve optimizing N as the search progresses.

References

[1] E. Burke and J. Landa Silva, The design of memetic algorithms for scheduling and timetabling problems, *Recent advances in memetic algorithms* (2005), 289–311.

[2] S. Knust, Scheduling non-professional table-tennis leagues, *European Journal of Operational Research* **200** (2010), 358–367.

[3] J. Schonberger, D. Mattfeld and H. Kopfer, Memetic algorithm timetabling for non-commercial sport leagues, *European Journal of Operational Research* **153** (2004), 102–116.

[4] V.P. Manotas Niño, Nurse Rostering Problems, *International Journal of Artificial Intelligence* **5** (2010), 109–116.

[5] A.A. Ceder, Optimal multi-vehicle type transit timetabling and vehicle scheduling, *Procedia-Social and Behavioral Sciences* **20** (2011), 19–30.

[6] J.D. Gehrke, O. Herzog, H. Langer, R. Malaka, R. Porzel and T. Warden, An agent-based approach to autonomous logistic processes, *KI-Künstliche Intelligenz* **24** (2010), 137–141.

[7] L. Di Gaspero and A. Schaerf, Tabu search techniques for examination timetabling, *Practice and Theory of Automated Timetabling III* (2001), 104–117.

[8] E.K. Burke and Y. Bykov, A late acceptance strategy in hill-climbing for exam timetabling problems, in: *International Conference on the Practice and Theory of Automated Timetabling*, Montreal, Canada, 2008.

[9] S. Abdennadher, M. Aly and M. Edward, Constraint-based timetabling system for the german university in cairo, *Applications of Declarative Programming and Knowledge Management* (2009), 69–81.

[10] R. Qu, E.K. Burke and B. McCollum, Adaptive automated construction of hybrid heuristics for exam timetabling and graph colouring problems, *European Journal of Operational Research* **198** (2009), 392–404.

[11] S.A. Rahman, A. Bargiela, E.K. Burke, E. Ozcan and B. McCollum, Construction of examination timetables based on ordering heuristics, (2009), 680–685.

[12] A.M. Rahman, S.S. Giasuddin and R.M. Rahman, Decision tree based routine generation (drg) algorithm: A data mining advancement to generate academic routine and examtime tabling for open credit system, *Journal of Computers* **5** (2010), 12–22.

[13] L. Zhang, B. Zhang and J. Yu, Improvement of adaptive genetic algorithm and its application in examination timetabling optimization problem, in: *International Conference on Computer Science and Service System (CSSS)*, Nanjing, 2011, pp. 1326–1329.

[14] N. Pillay and W. Banzhaf, An informed genetic algorithm for the examination timetabling problem, *Applied Soft Computing* **10** (2010), 457–467.

[15] R. Qu, E. Burke, B. McCollum, L.T.G. Merlot and S.Y. Lee, A survey of search methodologies and automated system development for examination timetabling, *Journal of Scheduling* **12** (2009), 55–89.

[16] S.R. Sutar and R.S. Bichkar, University timetabling based on hard constraints using genetic algorithm, *International Journal of Computer Applications* **42** (2012), 1–7.

[17] O.I. Obaid, M.S. Ahmad, S.A. Mostafa and M.A. Mohammed, Comparing performance of genetic algorithm with varying crossover in solving examination timetabling problem, *Journal of Emerging Trends in Computing and Information Sciences* **3** (2012).

[18] Ö. Ülker, E. Özcan and E. Korkmaz, Linear linkage encoding in grouping problems: applications on graph coloring and timetabling, *Practice and Theory of Automated Timetabling VI* (2007), 347–363.

[19] S. Abdullah, H. Turabieh, B. McCollum and P. McMullan, A tabu-based memetic approach for examination timetabling problems, *Rough Set and Knowledge Technology* (2010), 574–581.

[20] E. Özcan, A.J. Parkes and A. Alkan, The interleaved constructive memetic algorithm and its application to timetabling, *Computers & Operations Research* **39** (2011), 2310–2322.

[21] S. Rahman, A. Bargiela, E. Burke, E. Ozcan and B. McCollum, Construction of examination timetables based on ordering heuristics, in: *International Symposium on Computer and Information Sciences*, Guzelyurt, 2009, pp. 680–685.

[22] J. Han, M. Kamber and J. Pei, *Data Mining: Concepts and Techniques*, Morgan Kaufmann, 2011.

[23] N. Jenkins and T. Gedeon, Genetic algorithms applied to

university exam scheduling, in: *Proceedings of the 1997 International Conference on Neural Information Processing (ICONIP)*, Dunedin, 1997, pp. 1034–1037.

[24] E. Fast, C. Le Goues, S. Forrest and W. Weimer, Designing better fitness functions for automated program repair, in: *12th Annual Conference on Genetic and Evolutionary Computation*, New York, 2010, pp. 965–972.

[25] M. Caramia, P. Dell'Olmo and G. Italiano, New algorithms for examination timetabling, *Algorithm Engineering* (2001), 230–241.

[26] L. Paquete and T. Stutzle, Empirical analysis of tabu search for the lexicographic optimization of the examination timetabling problem, in: *International Conference on the Practice and Theory of Automated Timetabling*, Gent, Belgium, 2002, pp. 413–420.

[27] L. Di Gaspero, Recolour, shake and kick: A recipe for the examination timetabling problem, in: *4th International Conference on the Practice and Theory of Automated Timetabling*, Belgium, 2002, pp. 404–407.

[28] E. Burke and J. Newall, Enhancing timetable solutions with local search methods, *Practice and Theory of Automated-Timetabling IV* (2003), 195–206.

[29] L. Merlot, N. Boland, B. Hughes and P. Stuckey, A hybrid algorithm for the examination timetabling problem, *Practice and Theory of Automated Timetabling IV* (2003), 207–231.

[30] E.K. Burke and J. Newall, Solving examination timetabling problems through adaption of heuristic orderings, *Annals of operations Research* **129** (2004), 107–134.

[31] H. Asmuni, E. Burke, J. Garibaldi and B. McCollum, Fuzzy multiple heuristic orderings for examination timetabling, *Practice and Theory of Automated Timetabling V* (2005), 334–353.

[32] S. Petrovic, Y. Yang and M. Dror, Case-based initialisation of metaheuristics for examination timetabling, *Multidisciplinary scheduling: theory and applications* (2005), 289–308.

[33] M. Eley, Ant algorithms for the exam timetabling problem, in: *Practice and Theory of Automated Timetabling VI, Lecture Notes in Computer Science,* Vol. 3867/2007, Springer-Verlag, 2007, pp. 364–382.

[34] F. Djannaty and A. Mirzaei, Enhancing max-min ant system for examination timetabling problem, *International Journal of Soft Computing* **3** (2008), 230–238.

[35] M.B. El Den and R. Poli, Grammar-based genetic programming for timetabling, in: *IEEE Congress on Evolutionary Computation*, Trondheim, 2009, pp. 2532–2539.

[36] E.K. Burke, A.J. Eckersley, B. McCollum, S. Petrovic and R. Qu, Hybrid variable neighbourhood approaches to university exam timetabling, *European Journal of Operational Research* **206** (2010), 46–53.

[37] N. Pillay and W. Banzhaf, A developmental approach to the uncapacitated examination timetabling problem, *Parallel Problem Solving from Nature – PPSN X* (2008), 276–285.

[38] R. Abounacer, J. Boukachour, B. Dkhissi and A.E.H. Alaoui, A hybrid ant colony algorithm for the exam timetabling problem, *Revue ARIMA* **12** (2010), 15–42.

[39] H. Turabieh and S. Abdullah, An integrated hybrid approach to the examination timetabling problem, *Omega* **39** (2011), 598–607.

[40] N.R. Sabar, M. Ayob, G. Kendall and R. Qu, A Honey-bee mating optimization algorithm for educational timetabling problems, *European Journal of Operational Research* **216** (2011), 533–543.