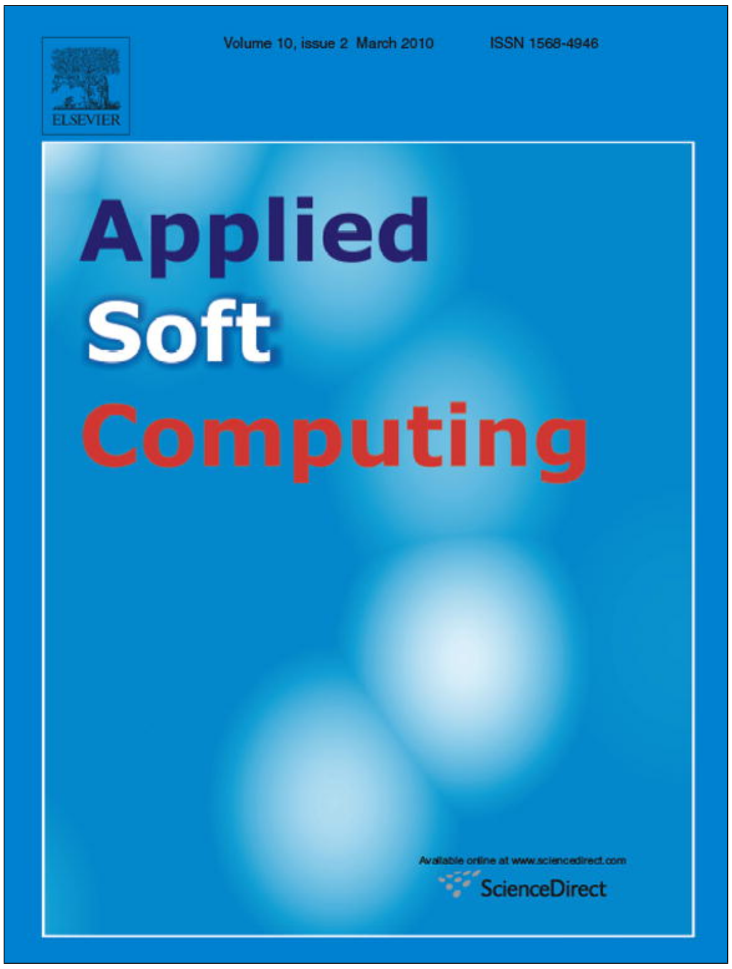

Volume 10, issue 2 March 2010
ISSN 1568-4946
ELSEVIER
Applied Soft Computing
Available online at www.sciencedirect.com
ScienceDirect

Applied Soft Computing 10 (2010) 457–467

Contents lists available at ScienceDirect

Applied Soft Computing

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

An informed genetic algorithm for the examination timetabling problem

N. Pillay [a,*], W. Banzhaf [b]

[a] School of Computer Science, University of KwaZulu-Natal, Pietermaritzburg Campus, Pietermaritzburg, KwaZulu-Natal, South Africa
[b] Department of Computer Science, Memorial University of Newfoundland, St. John's, NL, A1B 3X5, Canada

A R T I C L E I N F O

Article history:
Received 8 February 2007
Received in revised form 20 January 2009
Accepted 3 August 2009
Available online 11 August 2009

Keywords:
Examination timetabling
Evolutionary algorithms
Genetic algorithms
Heuristics

A B S T R A C T

This paper presents the results of a study conducted to investigate the use of genetic algorithms (GAs) as a means of inducing solutions to the examination timetabling problem (ETP). This study differs from previous efforts applying genetic algorithms to this domain in that firstly it takes a two-phased approach to the problem which focuses on producing timetables that meet the hard constraints during the first phase, while improvements are made to these timetables in the second phase so as to reduce the soft constraint costs. Secondly, domain specific knowledge in the form of heuristics is used to guide the evolutionary process. The system was tested on a set of 13 real-world problems, namely, the Carter benchmarks. The performance of the system on the benchmarks is comparable to that of other evolutionary techniques and in some cases the system was found to outperform these techniques. Furthermore, the quality of the examination timetables evolved is within range of the best results produced in the field.

1. Introduction

The induction of examination timetables is a well researched field and various artificial intelligence techniques such as Tabu search, simulated annealing and evolutionary algorithms, including genetic algorithms, have been evaluated for this purpose. The study presented in this paper applies genetic algorithms to the uncapacitated examination timetabling problem. The approach taken in this study differs from previous studies applying genetic algorithms to this domain as follows:

- A two-phased approach is taken to the problem. In the first phase a genetic algorithm is employed to produce timetables that do not violate any hard constraints and in the second phase a genetic algorithm is used to optimize the soft constraint costs of the timetables evolved during the first phase.
- The genetic algorithm implemented in the first phase uses domain specific knowledge, in the form of heuristics, to guide the evolutionary process.

The informed genetic algorithm (IGA) was tested on the Carter benchmark set which is comprised of 13 real-world problems. The results obtained are comparable to that of other artificial intelligence methodologies applied to this domain.

The study presented in this paper has made the following contributions:

- This study has illustrated the effectiveness of using domain knowledge, in the form of heuristics, to guide the search process when applying genetic algorithms to the uncapacitated examination timetabling problem.
- It introduces a new low-level heuristic, namely, highest cost, for the domain of examination timetabling.
- Although genetic algorithms have previously been applied to this domain, most of the systems implemented have been used to solve a problem for a particular institution and have not been tested on the Carter benchmarks which are generally used for comparing the performance of different methodologies applied to this domain. The only study testing a GA system on the Carter benchmark set it that conducted by Erben and Song [14]. However, the results obtained are not presented in the paper and the authors report that the performance of the system was not comparable to that of other techniques for this set of problems. Thus, one of the contributions of this study is the comparison of a GA based system with that of other methodologies on this set of benchmarks. Furthermore, this study provides a starting point for the further improvement of genetic algorithms in this domain.

The following section defines the examination timetabling problem. Section 3 provides an overview of methodologies previously applied to the uncapacitated ETP. A summary of studies

* Corresponding author. Tel.: +27 33 2605644; fax: +27 33 2605648.
E-mail addresses: pillayn32@ukzn.ac.za (N. Pillay), banzhaf@cs.mun.ca (W. Banzhaf).

1568-4946/$ – see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.asoc.2009.08.011

using evolutionary algorithms to solve the examination timetabling problem is presented in Section 4. Section 5 outlines the experimental setup for testing the system. The IGA system implemented is described in Section 6. The performance of the system on the 13 Carter benchmarks is discussed in Section 7. Finally, Section 8 summarises the findings of the study.

2. The examination timetabling problem (ETP)

The examination timetabling problem basically involves allocating a set of examinations to a given set of examination periods. Some versions of this problem require the solution timetable to use a minimum number of periods. Each ETP has its own set of *hard constraints* and *soft constraints*.

Hard constraints must be satisfied by the timetable. For example, a student cannot write two examinations at the same time, i.e. there must not be any clashes. Timetables that meet the hard constraints of the problem are referred to as *feasible* timetables.

Soft constraints on the other hand are wish lists of the characteristics that we would like the timetable to possess, e.g. examinations with a larger number of enrolments must be scheduled early in the timetable to allow sufficient time for them to be marked. All soft constraints cannot be met and we aim to minimise the soft constraint cost for a timetable. Examinations are usually allocated to periods so as to ensure that hard constraints are not violated and soft constraint costs are minimised.

There are two versions of the examination timetabling problem, namely, the *capacitated* ETP and the *uncapacitated* ETP. In the capacitated version of the problem room allocation is taken into consideration while in the uncapacitated version it is not.

3. Techniques applied to the ETP

The approach taken by earlier attempts at solving the ETP generally involved sorting examinations according to the difficulty associated with scheduling the examination, and allocating the most difficult examinations first so as to ensure that clashes did not occur. In the case of clashes re-allocation of examinations was performed. A low-level heuristic was used to assess the difficulty of an examination. Low-level heuristics generally used for this purpose include largest degree, largest enrolment, largest weighted degree and saturation degree. Research in this field was initiated by the study conducted by Carter et al. [7] which employed such a heuristic-based sequential technique with backtracking to generate examination timetables. This system was used to induce examination timetables for 13 different institutions. These 13 real-world problems are now referred to as the Carter benchmarks and are used to compare the performance of different techniques applied to the uncapacitated ETP.

Since this first study the field has advanced fairly rapidly and a number of different optimization techniques have been applied to this domain. This field has basically developed in two directions. The first focuses on applying one or more optimization techniques to the problem to generate examination timetables that meet the hard and soft constraints for the particular problem. The second area examines methodologies to induce hyper-heuristics for the problem. Hyper-heuristics are used to identify which heuristic or combination of heuristics is most suitable for the problem at hand and are used by the search process. This study falls into the first area and thus this section provides an overview of previous studies conducted to generate solutions to the uncapacitated examination timetabling problem. There is a vast amount of work done in this area and the studies described are those that are commonly cited in the literature as having made a contribution to the field.

One of the earlier studies is that conducted by Di Gaspero and Schaerf [11] to implement a Tabu search to generate solutions to 12 of the Carter benchmarks. Input to the search is a graph representing student conflicts with node weights indicating the number of students writing the examination and edge weights representing the number of students writing both the examinations joined by the edge. The search space consists of both feasible and infeasible colourings of the graph. In an attempt to reduce the runtime of the algorithm, the Tabu search was started with an initial solution generated by a greedy algorithm.

Caramia et al. [6] use sequential construction techniques to induce examination timetables. A greedy scheduler is firstly applied to allocate examinations. Examinations are allocated according to their priority value, i.e. the number of conflicts the examination has. Each examination is allocated to a clash-free period that produces the lowest soft constraint cost. A penalty decreaser is then applied to further reduce the soft constraint costs. If the penalties can no longer be decreased a penalty trader is used to add an additional period to the timetable and re-allocate examinations so as to reduce the soft constraint cost.

The GRASP system implemented by Casey and Thompson [8] takes a two-phased approach to the problem. The aim of the first phase is to produce a feasible timetable. This is achieved by ordering the examinations according to one of the low-level heuristics, i.e. largest degree, largest weighted degree, largest enrolment or saturation degree, and applying roulette wheel selection to the first n examinations to determine which examination to allocate next. Each examination is allocated to the first clash-free period available. If a clash-free period cannot be found, a form of backtracking is performed to re-schedule examinations so that clashes are eliminated. The second phase focuses on minimising the soft constraint costs of a feasible timetable found during the first phase. A form of simulated annealing is used for this purpose.

Abdullah et al. [1] use the Ahuja–Orlin large neighbourhood search to generate timetables for 12 of the Carter benchmarks. A feasible timetable is firstly found and this timetable is then optimized to reduce the soft constraint costs. The search space consists of tree representations of each timetable with each branch representing a day and each level a timeslot. Each node stores a subset of examinations scheduled on the day corresponding to the branch the node is in and the timeslot corresponding to the level the node is at.

Ayob et al. [2] apply an iterative re-start variable neighbourhood search (IRVNS) to the uncapacitated examination timetabling problem. The search consists of three phases, namely, the initialization, construction and improvement phases. The first two phases focus on constructing timetables that meet the hard constraints. Examinations are allocated according to their difficulty which is assessed using the saturation degree. If necessary backtracking is performed to eliminate clashes. The improvement phase explores the neighbourhood of the constructed timetable so as to optimize the soft constraint cost.

The Flex–Deluge algorithm is employed by Burke and Bykov [5] to obtain solutions for 11 of the Carter benchmarks. This algorithm is essentially an extension of the Great Deluge algorithm that incorporates the use of hill-climbing.

Section 7 examines the performance of the systems surveyed in this section on the Carter benchmarks and compares the results obtained with that of the IGA.

4. Evolutionary algorithms and the ETP

The study presented in this paper uses an informed genetic algorithm to induce solutions to the uncapacitated ETP. Genetic algorithms are based on Darwin's theory of evolution [16]. The first step of the algorithm involves creating an initial population. This initial population then goes through a process of refinement comprised of a number of iterations referred to as generations. During each generation the fitness of each individual in the population is calculated using a fitness function. Parents are selected based on this fitness measure. Fitness proportionate selection, roulette wheel selection and tournament selection are selection methods commonly used for this purpose. Offspring are created using genetic operators. The genetic operators generally employed are crossover, mutation and reproduction. The population is iteratively refined until the termination criteria for the particular problem are met.

This section describes the evolutionary algorithms that have been applied to the examination timetabling problem including memetic algorithms, genetic algorithms, multi-objective evolutionary algorithms (MOEA), and ant colonization.

Burke et al. [4] investigate the use of a memetic algorithm as a means of inducing examination timetables. Each timetable is represented as a number of memes, one for each timeslot. Each meme stores a list of examinations allocated to the timeslot and the corresponding room that the examination has been scheduled in. The memetic algorithm employed by Burke et al. combines the use of light and heavy mutation together with hill-climbing. The light mutation operator selects one or more examinations and reschedules each examination while maintaining a feasible timetable. The heavy mutation operator reschedules the examinations of one or more selected examination periods. A hill-climbing operator is applied to the timetables output by both these operators to further improve the quality of the timetable. This system was successfully applied to evolve timetables for the University of Nottingham.

A similar approach is taken by Ozcan and Ersoy [17] in the implementation of the final exam scheduler (FES) system. The FES system employs the use of a memetic algorithm that combines a genetic algorithm and violated directed hill-climbing (VDHC). The chromosome representing each timetable is comprised of a number of genes, one for each examination. Each gene stores the timeslot and day the examination has been allocated to. The genetic algorithm uses one-point crossover and mutation to improve timetables. The mutation operators change the timeslot and/or day for an examination. This system was applied to a number of datasets from the Faculty of Engineering and Architecture of Yeditepe University.

Preliminary studies in the field have assessed the performance of genetic algorithms in this domain by applying GAs to test problems. For example, Chu and Fang [9] use a genetic algorithm to generate a solution for a test problem involving the allocation of 10 examinations for 50 students to 12 examination periods, i.e. 4 periods over 3 days. Each timetable is an array with each cell representing an examination. The cell stores the examination period the examination has been allocated to. The crossover and mutation operators are applied sequentially to create the next generation. The crossover operator randomly selects two crossover points in the chosen parents and swaps the fragments at these points. The mutation operator randomly selects two examinations and swaps their timeslots. This evolutionary process is repeated until an optimal timetable is found or the maximum number of generations has been completed. A similar study conducted by Shebani [20] implements a genetic algorithm to evolve a solution to a test problem requiring 11 examinations to be scheduled over two days with two examination sessions held on each day. Each timetable is represented as an array of examination periods with a group of examinations allocated to each timeslot. A crossover operator, which swaps the contents of one or more cells is used to create offspring for each generation. Burke et al. [3] also apply a genetic algorithm to test problems. However, in this study the performance of the GA is tested on the capacitated version of the ETP. The length of timetables is not fixed. The mutation and crossover operators are applied to selected parents to create offspring. The mutation operator makes random changes to the period and room an examination is scheduled to. The crossover operator swaps the early and late examinations in two of the selected parents.

Ross et al. [19] apply a genetic algorithm to the capacitated version of the Carter benchmarks. This version of the problem has the added constraint that a student should not sit for consecutive examinations on the same day. A timetable is represented as an array of timeslots with each index of the array corresponding to an examination. The value stored at an index is the timeslot that the examination has been allocated to. Timetables are penalized for both clashes and near-clashes, i.e. a student has consecutive examinations on the same day. The system was also used to generate a timetable for the University of Edinburgh for over 1000 examinations and 9000 students.

Wong et al. [21] use a genetic algorithm to induce an examination timetable for Ecole de Technologie Superieure. The evolved timetable was clash-free with no student having to sit three consecutive examinations and only 0.8% of the students had two consecutive examinations. Furthermore, only about 11% of the students had to sit examinations separated by 1 or 2 periods with the rest of the student body having more than two periods between examinations. Each timetable is represented by a vector with each index representing an examination and each element of the vector representing the timeslot that the examination has been allocated to. Binary tournament selection is used to choose parents for each generation. The crossover and mutation operators are applied to the chosen parents to create offspring.

Based on the work by Falkenauer [15], Erben [13] uses a steady-state grouping genetic algorithm to evolve examination timetables. Tournament selection is used for selecting parents.

Each chromosome is essentially an array of genes. Each gene is comprised of a group of examinations that do not clash. The index of each array cell corresponds to the timeslot that the group of examinations has been allocated to. Two mutation operators are defined. The first operator deletes one or more randomly selected genes and reschedules the examinations contained in the gene/genes using a first fit algorithm. The second mutation operator swaps the positions of two randomly chosen genes. The crossover operator randomly selects one or more genes in each of the parents and swaps these groups of examinations. Excess examinations are removed and missing examinations are allocated to timeslots using a first fit algorithm. A later version of this system [14] gives priority to unscheduled examinations with a larger number of conflicts by swapping these with scheduled examinations and then applying the first fit algorithm to the new list of unscheduled examinations. This version also includes an additional mutation operator that reduces near-clashes, i.e. consecutive examinations with common students. This system was tested on the Carter benchmark set but the results are not reported in the paper [14]. However, the authors state that the performance of this system was not comparable to other methodologies applied to this domain.

Paquete and Fonseca [18] and Cote et al. [10] apply multi-objective evolutionary algorithms to the ETP. Paquete and Fonseca use a multi-objective evolutionary algorithm to induce a timetable for the Unit of Exact and Human Sciences of the University of Algarve. This problem requires 249 examinations to be scheduled in 30 timeslots. Each constraint corresponds to an objective that must be met by the evolved timetable. Elements of the population are essentially chromosomes consisting of a set of possible examination slots for each exam. Pareto ranking is used to measure the fitness of each timetable. The mutation operator is used to improve timetables.

Table 1
Carter benchmarks.

Problem	Institution	Periods	No. of examinations	No. of students	Density of conflict matrix
car-f-92 I	Carleton University, Ottawa	32	543	18419	0.14
car-s-91 I	Carleton University, Ottawa	35	682	16925	0.13
ear-f-83 I	Earl Haig Collegiate Institute, Toronto	24	190	1125	0.27
hec-s-92 I	Ecole des Hautes Etudes Commerciales, Montreal	18	81	2823	0.42
kfu-s-93	King Fahd University of Petroleum and Minerals, Dharan	20	461	5349	0.06
lse-f-91	London School of Economics	18	381	2726	0.06
pur-s-93 I	Purdue University, Indiana	43	2419	30029	0.03
rye-s-93	Ryerson University, Toronto	23	486	11483	0.08
sta-f-83 I	St Andrew's Junior High School, Toronto	13	139	611	0.14
tre-s-92	Trent University, Peterborough, Ontario	23	261	4360	0.18
uta-s-92 I	Faculty of Arts and Sciences, University of Toronto	35	622	21266	0.13
ute-s-92	Faculty of Engineering, University of Toronto	10	184	2749	0.08
yor-f-83 I	York Mills Collegiate Institute, Toronto	21	181	941	0.29

The hybrid multi-objective algorithm (hMOEA) implemented by Cote et al. [10] maintains both a population and an archive of non-dominated timetables on each iteration of the overall algorithm. The population and archive are initialized to contain randomly created timetables. The algorithm then performs a number of iterations each consisting of four phases. The first phase applies two local search operators to both the population and the archive. The first operator eliminates hard constraint violations while the second operator reduces the proximity cost of the soft constraints. The first operator implements a Tabu search while the second operator uses a Variable Neighborhood Descent (VND) search. The second phase performs a Pareto ranking of the population and archive. This is followed by an "update" phase during which the weaker elements of the archive are replaced. The last phase mutates the population and applies tournament selection with a tournament size of 2 to create the next generation. The mutation operator changes the timeslot of one or more randomly selected examinations. The system was applied to the Carter benchmarks [7]. The authors report that the performance of the multi-objective approach was comparable to other optimization techniques used for this purpose.

Eley [12] applies a Max–Min ant system (MMAS) to the uncapacitated version of the examination timetabling problem. The algorithm consists of n cycles during which m ants generate feasible timetables. Hill-climbing is applied to the best timetable constructed during each cycle to further reduce the cost of the timetable. The hill-climbing search is terminated if the timetable can no longer be improved. The examination to be scheduled next is chosen from a list of preferred exams. This list is comprised of a percentage of the examinations with the lowest saturation degree. The use of this list was found to reduce the computational time and improve the quality of timetables.

A majority of the studies have used evolutionary algorithms to induce timetables for a particular institution with specific hard and soft constraints. Two systems, namely, the hMOEA implemented by Cote et al. [10] and the ant system employed by Eley [12], have generated solutions for the Carter benchmarks. A comparison of the performance of these systems and that of the IGA is presented in Section 7.

5. Experimental methodology

The main aim of this study is to determine the performance of genetic algorithms on the uncapacitated ETP. The IGA system was applied to the 13 Carter benchmarks listed in Table 1.

This dataset has the following hard and soft constraint requirements:

- Hard constraint – A feasible timetable is one in which there are no clashes, i.e. no student must be scheduled to write more than one examination at the same time.
- Soft constraint – The examinations must be well spaced for students. This cost is defined in terms of how well examinations with common students are spaced. The proximity cost defined by Carter et al. [7] is used to calculate this cost. This cost is calculated using the following equation:

$$\frac{\sum w(|ei - ej|)Nij}{S} \tag{1}$$

where $|e_i - e_j|$ is the distance between the periods of each pair of examinations (e_i,e_j) with common students. N_{ij} is the number of students common to both examinations. S is the total number of students. $w(1) = 16$, $w(2) = 8$, $w(3) = 4$, $w(4) = 2$ and $w(5) = 1$, i.e. the smaller the distance between periods the higher the weight allocated.

The overall approach of the IGA presented in this paper is comprised of two phases. In the first phase a genetic algorithm is implemented to evolve feasible timetables. The second phase employs a genetic algorithm to optimize the soft constraint cost of the feasible timetables evolved during phase 1. The genetic parameter values used for phase 1 are tabulated in Table 2. Test runs were performed to find suitable values for these parameters. Decreasing the population size resulted in the system taking longer to converge to a feasible timetable. However, increasing the size of the population to beyond 1000 did not speed up convergence. The algorithm generally converged within the first 15–20 generations and hence a higher number of generations did not in any way improve the performance of the system. Increasing the limit on the number of mutation iterations tended to impede the convergence of the algorithm and increased the overall runtime of the system.

Table 3 lists the parameter values used for phase 2. As in the case of phase 1, test runs were also performed to find suitable

Table 2
Parameters for phase 1.

Population size	1000
Maximum generations	20
Tournament size	10
Maximum mutation iterations	3

Table 3
Parameters for phase 2.

Population size	500
Maximum generations	1000
Tournament size	10
Maximum mutation iterations	3

```
procedure iga()
begin
  initialize the array *timetables* to store the *timetables* meeting the hard constraints
  for *count* → 1 to *n*
    begin
      do
        *timetables*[*count*]= call the method *phase1* to create a timetable meeting the hard constraints
        while ( *timetables*[*count*] already exists in *timetable and* the limit on the number of attempts to
                find a different timetable is not exceeded)
  endfor
  *best_timetable* = invoke the method *phase2(timetables)* to optimize the soft constraint costs
                  and return the timetable with the lowest soft constraint cost
  return *best_timetable*
end
```

Fig. 1. Overall IGA algorithm.

parameter values for this phase. A population size of 500 allowed for sufficient variety in the initial population, however increasing the population size to 1000 did not increase the variety much more. Runs were performed with the maximum number of generations increased to 2000 and 3000. This did not provide any improvements as the algorithm basically converged within a 1000 generations.

The system was implemented in Java using JDK 1.4.2 and simulations were run on an Apple iMac with a 2.16 Mhz Intel Core 2 Duo processor and 1 gigabyte of memory.

6. Proposed system

This section describes the IGA implemented to evolve solutions to the examination timetabling problem. The overall approach consists of two phases. The first phase focuses on evolving timetables that meet the hard constraints while the second phase improves the timetables generated in the first phase by minimising the soft constraint values. The algorithm is illustrated in Fig. 1.

In both phases genetic algorithms have been implemented to evolve and improve timetables. Each timetable is represented as a linear structure with each cell representing a timetable period. Each cell stores a string of integer values representing the examinations allocated to that period. Fig. 2 illustrates an example of the representation used.

Both phases of the algorithm are described in detail in the following sections.

6.1. Phase 1

This phase implements a genetic algorithm to induce a timetable that meets the hard constraints. While the main focus of this phase is to produce timetables that satisfy the hard constraints, it also attempts to minimise the soft constraint cost. However, this minimisation is not at the expense of violating any

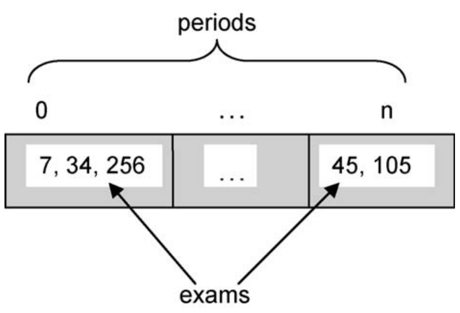


Fig. 2. Representation of a timetable.

hard constraints. The overall algorithm employs the generational control model. An initial population of timetables is created and iteratively refined during each generation using mutation until a feasible timetable is found.

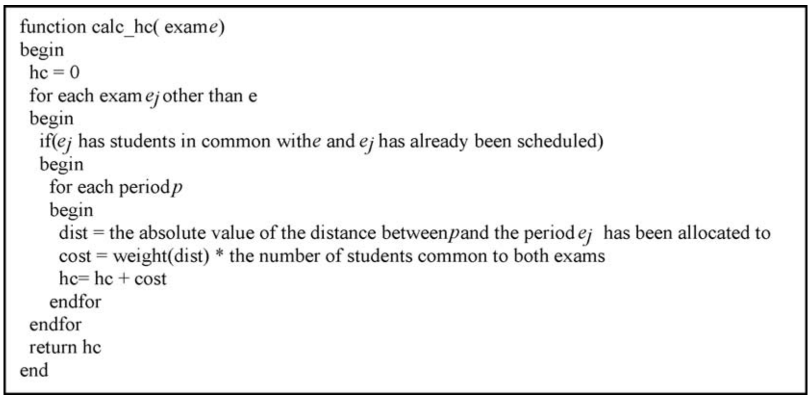
6.1.1. Initial population generation

The genetic algorithm implemented in this phase differs from that employed in previous studies applying GAs to the ETP in that it uses domain specific knowledge, in the form of heuristics, during initial population generation to guide the evolutionary process. Section 6.1.1.1 presents an analysis of the heuristics used and Section 6.1.1.2 describes the overall initial population generation process.

6.1.1.1. Using heuristics. Section 3 describes a number of methodologies that use low-level heuristics to order examinations during the timetable construction process. The low-level heuristics generally used are the largest degree, largest weighted degree, largest enrolment and saturation degree. In previous studies applying genetic algorithms to the ETP timetables are constructed by the random allocation of examinations to timeslots. The study presented in this paper investigated the effect of using low-level heuristics in the construction of timetables. The low-level heuristic is used to assess the difficulty of examinations. Instead of randomly choosing the next examination to allocate, the examinations are scheduled sequentially according to their difficulty. This process is described in Section 6.1.1.2. The following low-level heuristics have been tested for this purpose.

- Largest degree (LD) – The number of conflicts an examination is involved in. The examination with the largest number of conflicts is the most difficult.
- Largest enrolment (LE) – The examination with the largest student enrolment is the most difficult to schedule.
- Largest weighted degree (LWD) – The examination with the largest number of conflicting students is the most difficult.
- Saturation degree (SD) – The number of periods that an examination can be allocated to without causing a clash, i.e. the number of feasible periods. A smaller value indicates an examination that is more difficult to schedule.
- Highest cost (HC) – This is the cost of scheduling an examination *e* in terms of its distance from examinations that it has students in common with. Fig. 3 illustrates the function used to calculate this cost.

Three of the Carter datasets, with varying conflict matrix densities, namely, *hec-s-92*, *sta-f-83*, and *ute-s-92*, were used to test the effect of using heuristics in constructing timetables. The following versions of the GA for phase 1 were implemented for comparison purposes:



```
function calc_hc( exame)
begin
 hc = 0
 for each exam ej other than e
 begin
  if(ej has students in common withe and ej has already been scheduled)
  begin
   for each period p
   begin
    dist = the absolute value of the distance between p and the period ej  has been allocated to
    cost = weight(dist) * the number of students common to both exams
    hc= hc + cost
   endfor
 endfor
 return hc
end
```

Fig. 3. Pseudo code for calculating the HC heuristic for examination *e*.

- Random – Both the examination to schedule next and the examination period it is allocated to, are randomly chosen.
- Best-slot – In this version the examination to be scheduled next is randomly chosen but it is allocated to the minimum cost period. Fig. 5 in Section 6.1.1.2 defines the algorithm to determine the minimum cost period.
- Heuristic – At each stage of the construction process the examinations are ordered according to their difficulty and the most difficult examination is scheduled next. Each examination is allocated to the minimum cost period.

Thirty runs per dataset were performed for each version of phase 1. These simulations were run on a PC with an Intel Pentium III 1000 Mhz processor and 256 MB of memory. The values used for the GA parameters for these runs are listed in Table 4. The values of these parameters were obtained empirically by performing test runs. A lower value for the population size increased the time it took for the algorithm to converge while a higher value did not any anyway improve the convergence of algorithm. The algorithm was found to converge within 50 generations. A value lower than 10 for the tournament size did not provide sufficient selection pressure while higher values resulted in the algorithm converging quicker, however it often led to premature convergence. An increase in the number of mutation iterations did not in anyway improve the performance of the algorithm and resulted in longer runtimes.

Table 5 lists the average time and soft constraint cost for each of the datasets. The soft constraint cost is calculated using Eq. (1) (Section 5). It also lists the number of runs which produced infeasible timetables. It is evident from Table 5 that the allocation of an examination to the minimum cost slot and the use of heuristics affect both the soft constraint cost of the timetable and the runtime needed to produce a feasible timetable. The allocation of randomly selected examinations to the minimum cost slot reduces both the runtime of phase 1 and the soft constraint cost of the timetable. Note that in some cases the random version of the system was unable to evolve a feasible timetable within 50 generations. In these instances the cost for the run was calculated to be 2 times the maximum proximity cost for all the runs producing feasible timetables. The heuristic version of the system produced the best results. With the exception of the largest degree heuristic all the other low-level heuristics used resulted in a reduction in both the soft constraint cost and the time taken to induce a feasible timetable. The saturation degree heuristic produced the lowest runtimes while the highest cost heuristic produced the lowest soft constraint cost. Hypothesis tests, namely *Z* tests, were performed to test the statistical significance of these results. The decision rules and hypotheses are specified in the tables that follow.

Table 6 lists the levels of significance and the corresponding critical values and decision rules. The hypotheses and corresponding *Z* values for cost comparisons are listed in Table 7 and the hypotheses and corresponding *Z* values for the time comparisons are listed in Table 8.

It can be seen from Table 7 that, with an exception of the largest degree heuristic for *sta-f-83*, in all other cases the *Z* value at a 0.01 significance level is greater than the critical value. Thus, in these cases the null hypothesis is rejected and the results are significant.

Table 4
Parameters for heuristic experiments.

Population size	1000
Maximum generations	50
Tournament size	10
Maximum mutation iterations	3

Table 5
Average proximity cost and average runtimes.

		hec-s-92	*sta-f-83*	*ute-s-92*
Random	Mean cost	17.86	207.90	41.03
	Mean time (s)	466.83	837.40	1001.10
	No. of infeasible timetables	1	6	1
Best-slot	Mean cost	14.30	171.02	34.79
	Mean time (s)	77.33	38.23	41.67
	No. of infeasible timetables	0	0	0
LD	Mean cost	15.13	189.20	34.98
	Mean time (s)	1.63	0.21	5.07
	No. of infeasible timetables	0	0	0
LWD	Mean cost	12.75	170.32	29.88
	Mean time (s)	25.77	0.38	35.17
	No. of infeasible timetables	0	0	0
LE	Mean cost	12.94	170.41	29.00
	Mean time (s)	21.10	0.34	31.63
	No. of infeasible timetables	0	0	0
SD	Mean cost	13.84	168.84	31.65
	Mean time (s)	**0.42**	0.28	0.32
	No. of infeasible timetables	0	0	0
HC	Mean cost	**12.63**	**165.78**	**28.95**
	Mean time (s)	29.33	**0.17**	10.09
	No. of infeasible timetables	0	0	0

The bold values indicate the best soft constraint cost obtained and best time.

Table 6
Levels of significance, critical values and decision rules.

p	Critical value	Decision rule
0.01	2.33	Reject H_o if $Z > 2.33$
0.05	1.64	Reject H_o if $Z > 1.64$
0.10	1.28	Reject H_o if $Z > 1.28$

Table 7
Hypotheses and Z values for cost comparisons of the random version of the GA with the best-slot and heuristic versions.

Hypothesis	Z value		
	hec-s-92	*sta-f-83*	*ute-s-92*
H_o: $\mu_{Random\_cost} = \mu_{Best\_slot\_c}$ H_a: $\mu_{Random\_cost} > \mu_{Best\_slot\_c}$	3.55	2.60	3.91
H_o: $\mu_{Random\_cost} = \mu_{LD\_cost}$ H_a: $\mu_{Random\_cost} > \mu_{LD\_cost}$	2.75	1.32	3.72
H_o: $\mu_{Random\_cost} = \mu_{LWD\_cost}$ H_a: $\mu_{Random\_cost} > \mu_{LWD\_cost}$	5.12	2.65	7.04
H_o: $\mu_{Random\_cost} = \mu_{LE\_cost}$ H_a: $\mu_{Random\_cost} > \mu_{LE\_cost}$	4.94	2.64	7.64
H_o: $\mu_{Random\_cost} = \mu_{SD\_cost}$ H_a: $\mu_{Random\_cost} > \mu_{SD\_cost}$	4.02	2.75	5.94
H_o: $\mu_{Random\_cost} = \mu_{HC\_cost}$ H_a: $\mu_{Random\_cost} > \mu_{HC\_cost}$	5.25	2.97	7.67

Table 8
Hypotheses and Z values for time comparisons of the random version of the GA with the best-slot and heuristic versions.

Hypothesis	Z value		
	hec-s-92	*sta-f-83*	*ute-s-92*
H_o: $\mu_{Random\_time} = \mu_{Best\_slot\_t}$ H_a: $\mu_{Random\_time} > \mu_{Best\_slot\_t}$	5.10	14.35	8.33
H_o: $\mu_{Random\_time} = \mu_{LD\_time}$ H_a: $\mu_{Random\_time} > \mu_{LD\_time}$	6.25	15.03	8.65
H_o: $\mu_{Random\_time} = \mu_{LWD\_time}$ H_a: $\mu_{Random\_time} > \mu_{LWD\_time}$	5.92	15.03	8.39
H_o: $\mu_{Random\_time} = \mu_{LE\_time}$ H_a: $\mu_{Random\_time} > \mu_{LE\_time}$	5.98	15.03	8.42
H_o: $\mu_{Random\_time} = \mu_{SD\_time}$ H_a: $\mu_{Random\_time} > \mu_{SD\_time}$	3.30	15.03	8.69
H_o: $\mu_{Random\_time} = \mu_{HC\_time}$ H_a: $\mu_{Random\_time} > \mu_{HC\_time}$	5.87	15.03	8.61

At the 0.05 and 0.10 levels of significance all the Z values are greater than the critical values and hence all the results are statistically significant at these levels. In the case of time comparisons all the Z values are greater than the critical values for all datasets at all levels of significance. Thus, we can conclude that the time differences in Table 5 are significant.

Hypothesis tests were also performed to test the significance of the differences between the best-slot and heuristic versions of phase 1. The hypotheses and Z values for the soft constraint cost and time comparisons are listed in Table 9 and Table 10, respectively.

From Table 5 it appears that the best-slot version evolves timetables with a better soft constraint cost than the heuristic version of the system with the largest degree heuristic. This result was found to be significant at all levels of significance for all datasets except *ute-s-92*. Note that to test this hypothesis, the negative of the critical values in Table 6 is used and the decision rule rejects the null hypothesis if the Z value is less than the critical value. For all other low-level heuristics the heuristic version generated better quality timetables than the best-slot version. It is evident from Table 9 that these results are significant for the datasets *hec-s-92* and *ute-s-92* at all levels of significance. For the *sta-f-83* dataset these results are only significant at all levels of significance for the saturation degree and highest cost heuristics.

The Z values listed in Table 10 for all the datasets are greater than the critical value for all levels of significance. Thus, in all cases

Table 9
Hypotheses and Z values for cost comparisons of the best-slot and heuristic versions of the GA for phase 1.

Hypothesis	Z value		
	hec-s-92	*sta-f-83*	*ute-s-92*
H_o: $\mu_{Best\_slot\_c} = \mu_{LD\_cost}$ H_a: $\mu_{Best\_slot\_c} < \mu_{LD\_cost}$	−6.21	−21.71	−0.38
H_o: $\mu_{Best\_slot\_c} = \mu_{LWD\_cost}$ H_a: $\mu_{Best\_slot\_c} > \mu_{LWD\_cost}$	9.10	0.91	15.26
H_o: $\mu_{Best\_slot\_c} = \mu_{LE\_cost}$ H_a: $\mu_{Best\_slot\_c} > \mu_{LE\_cost}$	9.45	0.76	22.12
H_o: $\mu_{Best\_slot\_c} = \mu_{SD\_cost}$ H_a: $\mu_{Best\_slot\_c} > \mu_{SD\_cost}$	2.66	2.35	10.99
H_o: $\mu_{Best\_slot\_c} = \mu_{HC\_cost}$ H_a: $\mu_{Best\_slot\_c} > \mu_{HC\_cost}$	10.64	5.71	22.03

Table 10
Hypotheses and Z values for time comparisons of the best-slot and heuristic versions of the GA for phase 1.

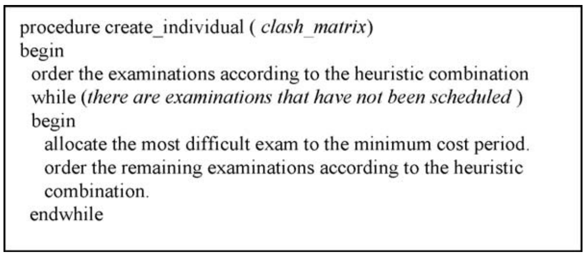
Hypothesis	Z value		
	hec-s-92	*sta-f-83*	*ute-s-92*
H_o: $\mu_{Best\_slot\_t} = \mu_{LD\_time}$ H_a: $\mu_{Best\_slot\_t} > \mu_{LD\_time}$	4.49	32.14	27.42
H_o: $\mu_{Best\_slot\_t} = \mu_{LWD\_time}$ H_a: $\mu_{Best\_slot\_t} > \mu_{LWD\_time}$	3.05	31.94	5.02
H_o: $\mu_{Best\_slot\_t} = \mu_{LE\_time}$ H_a: $\mu_{Best\_slot\_t} > \mu_{LE\_time}$	3.32	31.98	8.71
H_o: $\mu_{Best\_slot\_t} = \mu_{SD\_time}$ H_a: $\mu_{Best\_slot\_t} > \mu_{SD\_time}$	4.56	32.05	36.09
H_o: $\mu_{Best\_slot\_t} = \mu_{HC\_time}$ H_a: $\mu_{Best\_slot\_t} > \mu_{HC\_time}$	2.85	32.17	13.51

the null hypothesis is rejected proving that the time comparisons are significant for all heuristics.

Based on these results we can conclude that the best-slot and heuristic versions of phase 1 definitely perform better than the random version. The heuristic version takes less time in converging to a feasible timetable than the best-slot version irrespective of the heuristic used. However, when comparing the soft constraint cost of both versions the performance of the heuristic version is dependant on the heuristic used. The heuristic version of phase 1 using the saturation degree and highest cost heuristics outperform the best-slot version with respect to the soft constraint cost.

The results of these experiments have revealed the effectiveness of using heuristics to allocate examinations during timetable construction. The saturation degree heuristic has lower runtimes while the timetables induced using the highest cost heuristic have the lowest soft constraint cost. Thus, it was decided to allocate examinations using a combination of both these heuristics. During the process of sorting examinations according to their level of difficulty when constructing a timetable, examinations are compared by performing a Pareto comparison of the saturation degree and highest cost heuristics for the examinations. The following section describes the overall process of creating an individual and the population for phase 1.

6.1.1.2. Overall process. Each element of the initial population is created using the algorithm in Fig. 4. The examinations are firstly ordered according to their difficulty. As discussed in Section 6.1.1.1, a Pareto comparison of the saturation degree and highest cost heuristics is used to order examinations.



```
procedure create_individual ( clash_matrix)
begin
  order the examinations according to the heuristic combination
  while (there are examinations that have not been scheduled )
  begin
    allocate the most difficult exam to the minimum cost period.
    order the remaining examinations according to the heuristic
    combination.
  endwhile
```

Fig. 4. Algorithm for creating a timetable.

The function used to determine the minimum cost slot, is illustrated in Fig. 5.

6.1.2. Fitness calculation and selection

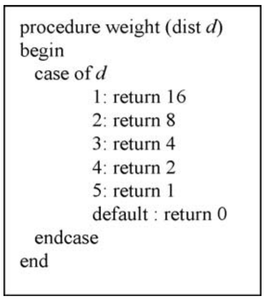
The raw fitness of an individual is calculated to be the number of clashes in the timetable. Tournament selection is used to choose the parents of the next generation.

This selection method randomly selects n elements, called the tournament, from the population and returns the timetable with the fewest number of clashes as the winner. If two timetables have the same number of clashes their soft constraint cost is then compared. The soft constraint cost is calculated using Eq. (1) (Section 5).

6.1.3. Mutation

The mutation operator reschedules one or more examinations in the chosen timetable. The number of examinations to be rescheduled is randomly selected to be between one and the maximum number of changes permitted. Only examinations that are involved in a clash are selected for rescheduling. Each examination is allocated to the minimum cost slot which is identified using the algorithm in Fig. 5. The following mutation and crossover operators were also tested for this domain:

- The random deletion and re-allocation of the examinations scheduled in a randomly chosen timeslot.



```
procedure weight (dist d)
begin
  case of d
          1: return 16
          2: return 8
          3: return 4
          4: return 2
          5: return 1
          default : return 0
  endcase
end
```

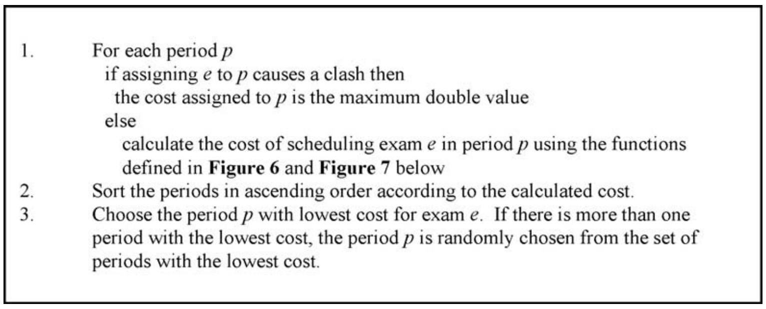
Fig. 7. Pseudo code for the weight function.

- Two slots are randomly chosen in a parent and the examinations scheduled in these slots are swapped.
- The examinations stored at randomly selected slots in two chosen parents are swapped. A repair mechanism is applied to the offspring to remove duplicate examinations and randomly allocate missing examinations.
- The examinations stored at randomly selected slots in two similar parents are swapped. A similarity index is used to determine the likeness of the parents. A repair mechanism is applied to the offspring to remove duplicate examinations and randomly allocate missing examinations.

However, test runs revealed that none of these operators improved the performance of the system any further and thus only the mutation operator described above was used.

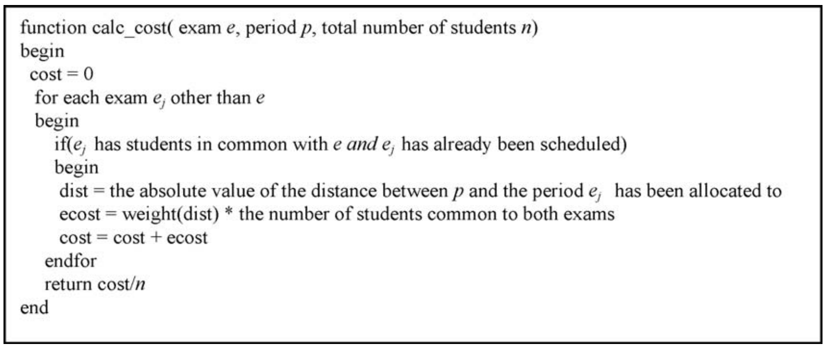
6.2. Phase 2

The main aim of this phase is to minimise the soft constraint costs of the feasible timetables constructed during phase 1. As illustrated in Fig. 1 the genetic algorithm of phase 1 is invoked n times to create an initial population of size n for phase 2. During phase 2 the genetic algorithm refines this initial population over a set number of generations using mutation. The refinement process focuses on reducing the soft constraint cost of the timetables.



```
1.    For each period p
        if assigning e to p causes a clash then
          the cost assigned to p is the maximum double value
        else
          calculate the cost of scheduling exam e in period p using the functions
          defined in Figure 6 and Figure 7 below
2.    Sort the periods in ascending order according to the calculated cost.
3.    Choose the period p with lowest cost for exam e. If there is more than one
      period with the lowest cost, the period p is randomly chosen from the set of
      periods with the lowest cost.
```

Fig. 5. Algorithm for finding the minimum cost period p for examination e.



```
function calc_cost( exam e, period p, total number of students n)
begin
  cost = 0
  for each exam $e_j$ other than e
  begin
    if($e_j$ has students in common with e and $e_j$ has already been scheduled)
    begin
      dist = the absolute value of the distance between p and the period $e_j$ has been allocated to
      ecost = weight(dist) * the number of students common to both exams
      cost = cost + ecost
    endfor
    return cost/n
end
```

Fig. 6. Pseudo code for the function used to calculate the cost of scheduling examination e in period p.

The raw fitness of each individual is the proximity cost defined in Eq. (1) (Section 5). The tournament selection method is also used during this phase to select parents of the next generation. As in the previous phase this method selects a tournament of size *n*. The element of the tournament with lowest soft constraint cost is reported as the winner.

A mutation operator, similar to that employed in phase 1, is applied to create the next generation. One or more examinations are randomly re-allocated so as not to cause any clashes, i.e. the offspring produced is a feasible timetable. This process is repeated until the offspring is at least as fit as its parent, i.e. the proximity cost of the offspring is less than or equal to that of the parent. While this results in the algorithm converging faster it could lead to premature convergence of the algorithm. Hence, a limit is set on the number steps of improvement. If an offspring with fitness as good as its parent cannot be found within a maximum number of steps, the current offspring is returned as the result of mutation.

7. Results and discussion

This section examines the performance of the IGA system on the 13 Carter benchmarks. Ten runs were performed for each dataset. The combination of the saturation degree and highest cost heuristics (SD-HC) were used to guide the search during initial population. Experiments were also conducted to test the effect of using the largest degree, largest weighted degree and largest enrolment heuristics to break ties when performing a Pareto comparison of examinations during the process of sorting the examinations according to difficulty. Table 11 lists the best cost, the mean cost and the standard deviation for each combination. The combination of the saturation degree and highest cost heuristics, with the use of the largest weighted degree and largest enrolment heuristics to break ties, have produced the best results. Table 12 lists the runtimes of the best results obtained by the IGA. The runtimes for each of the seeds were found to be more or less the same as the number of generations for the refinement process is fixed. For a majority of the datasets the runtime is less than an hour. For the larger datasets, namely, *car-f-92*, *car-s-91* and *uta-s-92*, the runtimes are just over an hour. The only dataset for which the runtime is high is *pur-s-92*. This is the largest dataset with 2419 examinations and 30,029 students. The best solutions generated by the IGA are accessible from http://titan.cs.unp.ac.za/∼nelishiap/et/iga.htm.

The performance of the IGA has also been compared to other methodologies applied to the same version of the Carter benchmarks. The performance of the methods is measured in terms of the soft constraint cost of the best timetable evolved by the method. A comparison of runtimes is not presented as some of the studies do not cite the runtime of the simulations and simulations have been run on machines of varying computing power for the different studies. We firstly compare the performance of the IGA with that of other evolutionary algorithms. The literature does not cite any other studies applying genetic algorithms to the Carter benchmarks. Thus a comparison of the IGA with other genetic algorithms is not possible. Hence, the results obtained by the IGA system are compared to the performance of other evolutionary

Table 11
IGA performance on the Carter benchmarks for the different heuristic combinations.

Dataset	SD–HC	SD–HC (LD)	SD–HC (LWD)	SD–HC (LE)
car-f-92 I	Best: 4.25 Mean: 4.32 S.D.: 0.044	Best: 4.24 Mean: 4.30 S.D.: 0.033	Best: 4.23 Mean: 4.30 S.D.: 0.047	Best: **4.22** Mean: 4.28 S.D.: 0.039
car-s-91 I	Best: 4.99 Mean: 5.08 S.D.: 0.042	Best: 5.01 Mean: 5.06 S.D.: 0.027	Best: **4.92** Mean: 5.05 S.D.: 0.060	Best: 4.97 Mean: 5.05 S.D.: 0.053
ear-f-83 I	Best: 36.83 Mean: 38.28 S.D.: 0.961	Best: 35.98 Mean: 36.53 S.D.: 0.664	Best: 35.92 Mean: 36.68 S.D.: 0.510	Best: **35.87** Mean: 36.49 S.D.: 0.411
hec-s-92 I	Best: **11.5** Mean: 11.69 S.D.: 0.109	Best: 12.22 Mean: 12.23 S.D.: 0.004	Best: 11.87 Mean: 11.99 S.D.: 0.047	Best: 11.71 Mean: 11.96 S.D.: 0.098
kfu-s-93	Best: 14.93 Mean: 15.23 S.D.: 0.188	Best: 14.43 Mean: 14.85 S.D.: 0.258	Best: **14.37** Mean: 14.42 S.D.: 0.032	Best: 14.4 Mean: 14.43 S.D.: 0.027
lse-f-91	Best: 10.91 Mean: 10.95 S.D.: 0.046	Best: 10.99 Mean: 11.03 S.D.: 0.028	Best: 10.97 Mean: 11.00 S.D.: 0.031	Best: **10.89** Mean: 10.95 S.D.: 0.0
pur-s-93 I	Best: 4.67 Mean: 4.71 S.D.: 0.020	Best: 4.66 Mean: 4.70 S.D.: 0.033	Best: **4.65** Mean: 4.70 S.D.: 0.037	Best: 4.7 Mean: 4.71 S.D.: 0.013
rye-s-93	Best: 9.34 Mean: 9.47 S.D.: 0.099	Best: 9.81 Mean: 10.01 S.D.: 0.093	Best: **9.3** Mean: 9.41 S.D.: 0.081	Best: **9.3** Mean: 9.45 S.D.: 0.111
sta-f-83 I	Best: 159.37 Mean: 159.45 S.D.: 0.070	Best: 157.83 Mean: 158.07 S.D.: 0.200	Best: 158.03 Mean: 158.24 S.D.: 0.204	Best: **157.81** Mean: 158.07 S.D.: 0.177
tre-s-92	Best: 8.46 Mean: 8.49 S.D.: 0.036	Best: 8.61 Mean: 8.77 S.D.: 0.123	Best: 8.43 Mean: 8.47 S.D.: 0.034	Best: **8.38** Mean: 8.45 S.D.: 0.038
uta-s-92 I	Best: 3.43 Mean: 3.50 S.D.: 0.047	Best: **3.35** Mean: 3.39 S.D.: 0.028	Best: **3.35** Mean: 3.40 S.D.: 0.032	Best: 3.37 Mean: 3.40 S.D.: 0.025
ute-s-92	Best: 28.07 Mean: 28.10 S.D.: 0.016	Best: 28.49 Mean: 28.57 S.D.: 0.092	Best: **27.24** Mean: 27.45 S.D.: 0.366	Best: 28.04 Mean: 28.05 S.D.: 0.014
yor-f-83 I	Best: 40.17 Mean: 41.37 S.D.: 0.471	Best: 39.86 Mean: 39.97 S.D.: 0.081	Best: 39.72 Mean: 40.76 S.D.: 0.487	Best: **39.33** Mean: 39.74 S.D.: 0.329

The bold values indicate the best soft constraint cost obtained.

Table 12
Runtimes for the best results.

Dataset	Runtime for best result
car-f-92 I	1 h 11 min
car-s-91 I	1 h 38 min
ear-f-83 I	12 min 31 s
hec-s-92 I	7 min 28 s
kfu-s-93	52 min 48 s
lse-f-91	47 min 43 s
pur-s-93 I	31 h 36 min
rye-s-93	1 h 12 min
sta-f-83 I	7 min 49 s
tre-s-92	18 min 41 s
uta-s-92 I	1 h 39 s
ute-s-92	11 min 3 s
yor-f-83 I	9 min 12 s

Table 13
The best results obtained by the IGA and other evolutionary algorithms.

Problem	IGA	hMOEA	MMAS
car-f-92 I	**4.2**	**4.2**	4.8
car-s-91 I	**4.9**	5.4	5.7
ear-f-83 I	35.9	**34.2**	36.8
hec-s-92 I	11.5	**10.4**	11.3
kfu-s-93	14.4	**14.3**	15.0
Lse-f-91	**10.9**	11.3	12.1
pur-s-93 I	**4.7**	–	5.4
rye-s-93	9.3	**8.8**	10.2
sta-f-83 I	157.8	**157.0**	157.2
Tre-s-92	**8.4**	8.6	8.8
uta-s-92 I	**3.4**	3.5	3.8
ute-s-92	27.2	**25.3**	27.7
yor-f-83 I	39.3	**36.4**	39.6

The bold values indicate the best soft constraint cost obtained.

Table 14
The best results obtained by IGA and other optimization methods applied to the Carter benchmarks.

Dataset	IGA	Di Gaspero and Schaerf	Caramia et al.	Casey and Thompson	Abdullah et al.	Ayob et al.	Burke et al.
car-f-92 I	4.22	5.2	6.0	4.4	4.36	4.51	**3.74**
car-s-91 I	4.92	6.2	6.6	5.4	5.21	4.90	**4.42**
ear-f-83 I	35.87	45.7	**29.3**	34.8	34.84	36.28	32.76
hec-s-92 I	11.5	12.4	**9.2**	10.8	10.28	11.06	10.15
kfu-s-93	14.37	18.0	13.8	14.1	13.46	14.74	**12.96**
lse-f-91	10.89	15.5	**9.6**	14.7	10.24	12.08	9.83
pur-s-93 I	4.65	–	**3.7**	–	–	4.66	–
rye-s-93	9.3	–	**6.8**	–	8.74	10.67	–
sta-f-83 I	157.81	160.8	158.2	**134.9**	159.28	157.32	157.03
tre-s-92	8.38	10.0	9.4	8.7	8.13	8.92	**7.75**
uta-s-92 I	3.35	4.2	3.5	–	3.63	3.58	**3.06**
ute-s-92	27.24	29.0	24.4	25.4	**24.21**	26.36	24.82
yor-f-83 I	39.33	41.0	36.2	37.5	36.11	38.97	**34.84**

The bold values indicate the best soft constraint cost obtained.

algorithms that have been applied to the same version of the Carter benchmarks. These studies include the hMOEA implemented by Cote et al. [10] and the MMAS implemented by Eley [12]. These studies are described in Section 4. Table 13 lists the best results obtained by the IGA and that obtained by hMOEA and MMAS.

It is clear from Table 13 that the results obtained by the IGA are comparable to the performance of other evolutionary algorithms. For six of the datasets the IGA has outperformed the other evolutionary methods. The performance of the IGA is also compared to those techniques that have been cited as making a contribution to the field. These studies are described in detail in Section 3. The performance of the IGA is compared to that of:

- The Tabu search implemented by Di Gaspero and Schaerf [11].
- The sequential heuristic construction methods employed by Caramia et al. [6].
- The GRASP system implemented by Casey and Thompson [8].
- The Ahuja–Orlin large neighbourhood search used by Abdullah et al. [1].
- The IRVNS implemented by Ayob et al. [2].
- The Flex–Deluge algorithm employed by Burke and Bykov [5].

Table 14 lists the best results obtained by the IGA and these methodologies. Although the IGA has not produced the best result for any of the datasets, it is evident from Table 14 that the performance of the IGA is comparable to the other methodologies applied to these benchmarks and in some cases has outperformed these methodologies.

8. Conclusion

The study presented in this paper implements an IGA to evolve examination timetables. This system differs from a standard GA in that domain knowledge, in the form of heuristics, is used to guide the search during initial population generation. It is evident from the results presented in the paper that the use of heuristics in scheduling examinations improves the performance of the genetic algorithm on the examination timetable problem. This study also introduces a new heuristic, namely, highest cost, which has been found to produce better quality timetables than other low-level heuristics used to assess the difficulty of examinations.

The performance of the IGA was found to be comparable to other evolutionary algorithms when applied to the Carter set of benchmarks and has outperformed these methods on a number of the datasets. Furthermore, the results obtained by IGA are within the range of the best results cited for this set of benchmarks and has produced better results than some of the other optimization techniques, such as Tabu search, applied to this set of benchmarks.

This study has provided a starting point for the application of GAs to the uncapacitated examination timetabling problem. Future work will focus on bettering the performance of GAs in this domain. For example, the effect of using the steady-state control model instead of the generational control will be examined as a means of improving the convergence time of the algorithm. This study has also illustrated the effectiveness of using a combination of low-level heuristics to guide the evolutionary process. Thus, investigations into using genetic programming as a means of evolving heuristic combinations, i.e. hyper-heuristics, that are tailored to a particular problem will be conducted.

Acknowledgement

The authors would like to thank the reviewers for their insightful comments and suggestions.

References

[1] S. Abdullah, S. Ahmadi, E.K. Burke, M. Dror, Investigating Ahuja–Orlin's large neighbourhood search for examination timetabling. Technical Report NOTTCS-TR-2004-8, School of CSiT, University of Nottingham, UK, 2004.

[2] M. Ayob, E.K. Burke, G. Kendall, in: E.K. Burke, H. Rudova (Eds.), An Iterative Re-Start Variable Neighbourhood Search for the Examination Timetabling Problem, PATAT, 2006, pp. 336–344, ISBN 80-210-3726-1.

[3] E.K. Burke, D. Elliman, R. Weare, A genetic algorithm based university timetabling system, in: Proceedings of the 2nd East-West International Conference on Computers in Education, Crimea, Ukraine, 19th–23rd September 1994, vol. 1, (1994), pp. 35–40.

[4] E.K. Burke, J.P. Newall, R.F. Weare, A memetic algorithm for university exam timetabling, in: E.K. Burke, P. Ross (Eds.), Practice and Theory of Automated Timetabling: Selected Papers from the 1st International Conference, Lecture Notes in Computer Science, vol. 1153, Spring-Verlag, Berlin, Heidelberg, 1996, pp. 241–250.

[5] E.K. Burke, Y. Bykov, Solving exam timetabling problems with the Flex-Deluge algorithm, in: E.K. Burke, H. Rudova (Eds.), Proceedings of PATAT 2006, 2006, pp. 370–372.

[6] M. Caramia, P. Dell'Olmo, G. Italiano, New algorithms for examination timetabling, in: S. Naher, D. Wagner (Eds.), Algorithm Engineering 4th International Workshop, WAE 2000, Lecture Notes in Computer Science, vol. 1982, Springer, Berlin, 2001, pp. 230–241.

[7] M.W. Carter, G. Laporte, S.Y. Lee, Examination timetabling: algorithmic strategies and applications, The Journal of the Operational Research Society 47 (3) (1996) 373–383.

[8] S. Casey, J. Thompson, GRASPing the examination scheduling problem, in: E.K. Burke, P. De Causmaecker (Eds.), The Practice and Theory of Automated Timetabling IV: Proceedings of the 4th International Conference on the Practice and Theory of Automated Timetabling, vol. 2740, Springer, Berlin, 2003, pp. 232–246.

[9] S.C. Chu, H.L. Fang, Genetic algorithms vs. Tabu search in timetable scheduling, in: Proceedings of the 3rd International Conference on Knowledge-Based Intelligent Information Engineering Systems, 31st August–1st September 1999, IEEE Press, Adelaide, Australia, 1999, pp. 492–495.

[10] P. Cote, T. Wong, R. Sabourin, Application of a hybrid multi-objective evolutionary algorithm to the uncapacitated exam proximity problem, in: E.K. Burke, M. Trick (Eds.), Practice and Theory of Timetabling V, 5th International Conference, PATAT 2004, Pittsburgh, PA, USA, 18–20 August, vol. 3616, Springer, Berlin, 2004 , pp. 294–312 (Revised Selected Papers, ISBN: 978-3-540-30705-1).

[11] L. Di Gaspero, A. Schaerf, Tabu search techniques for examination timetabling, in: Selected Papers from the 3rd International Conference on the Theory and Practice of Automated Timetabling, Lecture Notes in Computer Science, vol. 2079, Springer, Berlin, 2000, pp. 104–117.

[12] M. Eley, Ant algorithms for the exam timetabling problem, in: 6th International Conference PATAT 2006, Springer, Berlin, 2006, , pp. 167–180, ISBN: 80-210-3726-1.
[13] W. Erben, A grouping genetic algorithm for graph colouring and exam timetabling, in: E.K. Burke, M. Trick (Eds.), Selected Papers from the Third International Conference on the Practice and Theory of Automated Timetabling III, Lecture Notes in Computer Science, vol. 2079, Springer-Verlag, 2000, pp. 132–158, ISBN: 3-540-42421-0.
[14] W. Erben, P.Y. Song, A hybrid grouping genetic algorithm for examination timetabling, in: E.K. Burke, M. Trick (Eds.), PATAT '04 Proceedings of the 5th International Conference on the Practice and Theory of Automated Timetabling, Springer-Verlag, 2004, pp. 487–490.
[15] E. Falkenauer, Genetic Algorithms and Grouping Problems, John Wiley and Sons, Chichester, 1998.
[16] J.H. Holland, Adaptation in Natural and Artificial Systems, University of Michigan Press, 1975.
[17] E. Ozcan, E. Ersoy, Final exam scheduler – FES, in: Proceedings of the 2005 IEEE Congress on Evolutionary Computation, vol. 2, 2005, pp. 1356–1363.
[18] L.F. Paquete, C.M. Fonseca, A study of examination timetabling with multiobjective evolutionary algorithms, in: Proceedings of the 4th Metaheuristics International Conference (MIC 2001), 2001, pp. 149–154.
[19] P. Ross, E. Hart, D. Corne, Some observations about GA-based exam timetabling, in: Practice and Theory of Automated Timetabling II: Selected Papers from the Second International Conference, PATAT'97, Lecture Notes in Computer Science, vol. 1408–1998, Springer-Berlin Heidelberg, 1998, ISSN 0302-9743.
[20] K. Sheibani, An evolutionary approach for the examination timetabling problems, in: E.K. Burke, P. De Causmaecker (Eds.), The Proceedings of the 4th International Conference on Practice and Theory of Automated Timetabling, 21–23 August 2002, KaHo St.-Lieven, Gent, Belgium, (2002), pp. 387–396.
[21] T. Wong, P. Cote, P. Gely, Final exam timetabling: a practical approach, in: IEEE Canadian Conference on Electrical and Computer Engineering (CCECE 2002), vol. 2, 2002, 726–731.