A Study of Genetic Algorithms to Solve the School Timetabling Problem

Rushil Raghavjee and Nelishia Pillay

School of Mathematics, Statistics and Computer Science, University of KwaZulu-Natal, Pietermaritzburg Campus, KwaZulu-Natal, South Africa
{raghavjee,pillayn32}@ukzn.ac.za

Abstract. This paper examines the use of genetic algorithms (GAs) to solve the school timetabling problem. The school timetabling problem falls into the category of NP-hard problems. Instances of this problem vary drastically from school to school and country to country. Previous work in this area has used genetic algorithms to solve a particular school timetabling problem and has not evaluated the performance of a GA on different problems. Furthermore, GAs have not previously been applied to solving the South African primary or high school timetabling problem. The paper presents a two-phased genetic algorithm approach to solving the school timetabling problem and provides an analysis of the effect of different low-level construction heuristics, selection methods and genetic operators on the success of the GA approach in solving these problems with respect to feasibility and timetable quality. The GA approach is tested on a benchmark set of "hard" school timetabling problems, the Greek high school timetabling problem and a South African primary and high school timetabling problem. The performance of the GA approach was found to be comparable to other methods applied to the same problems. This study has also revealed that different combinations of low-level construction heuristics, selection methods and genetic operators are needed to produce feasible timetables of good quality for the different school timetabling problems. Future work will investigate methods for the automatic configuration of GA architectures of both phases.

Keywords: Timetabling, genetic algorithms, combinatorial optimization, evolutionary computation.

1 Introduction

Genetic algorithms have been successfully applied to solving combinatorial optimization problems such as university course and examination timetabling problems[1], the travelling salesman problem [2], and the bin packing problem [3] amongst others. Given the success in these domains, this paper presents an investigation of genetic algorithms in solving the school timetabling problem.

The school timetabling problem (STP) involves the scheduling of resources, or combinations of resources, to timetable slots in such a manner that the hard constraints of the problem are met and the soft constraints minimized [4]. Resources for this problem include classes, teachers and venues, amongst others. The requirements

F. Castro, A. Gelbukh, and M. González (Eds.): MICAI 2013, Part II, LNAI 8266, pp. 64–80, 2013.

of the problem include a specification of the number of times a particular teacher must meet a class. Some versions of the problem do not require venue allocations to be made while others include this constraint. Resources are allocated as class-teacher (or class-teacher-venue) tuples to the different timetable periods. The hard constraints of a problem are constraints that must be satisfied by a timetable in order for it to be operable. A timetable meeting all the hard constraints of the problem is said to be feasible. Examples of hard constraints include all class-teacher meetings must be scheduled the required number of times in the specified venue; no clashes, i.e. a resource, namely, a teacher, class, or venue, must not be scheduled more than once in a timetable period. The soft constraints on the other hand measure the quality of the timetable. These constraints define characteristics that we would like the timetable to possess but which may not always be possible. The aim is to minimize the number of soft constraints violated and this value is a measure of the quality of the timetable, i.e. the fewer soft constraints violated the better the timetable quality. A common soft constraint is daily limits on the number of lessons taken by a class on a particular subject and the number of lessons taught by a particular teacher. The hard and soft constraints differ from one timetabling problem to the next to such an extent that in some cases what may be defined as a hard constraint for one problem is a soft constraint for another and vice versa.

Genetic algorithms take an analogy from Darwin's theory of evolution. The standard genetic algorithm presented by Goldberg [5] implements the processes of initial population, evaluation, selection and regeneration by means of genetic operators. Elements of the population are represented as binary strings and each element, called a chromosome, is randomly created. A measure of how close a chromosome is to the solution is referred to as the fitness of a chromosome. The fitness is used to select parents to create offspring of the successive generation. Fitness proportionate or roulette wheel selection is traditionally used to choose parents. The reproduction, mutation and crossover operators are usually used to create the offspring of each generation. As the field has developed, variations of the standard genetic algorithm have emanated. These include the representation of chromosomes which now range from binary strings and character strings to matrices, depending on the problem domain. The effectiveness of tournament selection over fitness proportionate selection has also been established. In addition to this, instead of probabilities of each genetic operator being attached to each chromosome, application rates are set globally and applied in the creation of each generation, e.g. 40% of each generation will be created using mutation and 60% by means of crossover. Furthermore, implementation of genetic algorithms with just mutation has also proven to be effective [6].

Various methods have been applied to solving different versions of the school timetabling problem including tabu search, integer programming, constraint programming and constraint satisfaction methods, simulated annealing, neural networks, GRASP, tiling algorithms, the walk down jump up algorithm, bee algorithms and the cyclic transfer algorithm [4]. Hybrid approaches have also been applied to solving the school timetabling problem. Successful combinations of methods include randomized non-ascendant search (RNA) and tabu search, tabu search and the Floyd-Warshall algorithm, tabu search and graph colouring algorithms, beam search and branch and bound techniques, simulated annealing and very large neighbourhood search [4].

The school timetabling problem differs from school to school and country to country. However previous work has used genetic algorithms to find a solution to a specific school timetabling problem. The study presented in this paper evaluates genetic algorithms over different types of school timetabling problems. A two-phased approach, employing a GA in the first phase to evolve feasible timetables and a GA in the second phase to improve the quality of timetables generated in the first phase, is evaluated in solving the school timetabling problem.

The GA approach was tested on four different types of school timetabling problems, namely, the set of "hard" artificial timetabling problems made available by [6], the Greek high school timetabling problem, a South African primary and high school timetabling problem. It was found that combinations of different construction heuristics, selection methods and mutation operators were needed to generate feasible timetables of good quality for different problems. Hence, there appears to be a need for the automatic configuration of the GA architectures of both phases for the school timetabling problem. This will be examined as part of future work.

The contributions made by the study presented in the paper are: an evaluation of genetic algorithms over a set of different problems with varying characteristics, the identification and evaluation of low-level construction heuristics for this domain, and an evaluation of GAs in solving the South African school timetabling problems. The following section provides an overview of previous work using evolutionary algorithms to solve the school timetabling problem. The two-phased GA approach is presented in section 3. The methodology used to evaluate this approach is outlined in section 4 and section 5 discusses the performance of this approach in solving the different school timetabling problems. A summary of the findings of the study and future extensions of this work are presented in section 6.

2 Genetic Algorithms and School Timetabling

There has been a fair amount of research into using genetic algorithms to solve different types of school timetabling problems including generated problems [7, 8, 9], the Italian [10], Brazilian [11], German [12], Turkish [13], Greek [14] and Bosnian [15] school timetabling problem. Each element of the population is generally a two-dimensional array representing the timetable [8, 10, 13]. The fitness of an individual is the number of constraint violations [15] or the weighted sum of the constraint violations [10, 14]. Either fitness proportionate selection [8, 9, 12, 14] or tournament selection [15] is used to choose parents for each generation. The genetic operators applied to create the offspring of each generation are reproduction, mutation and crossover.

GAs have also been used in combination with other techniques to obtain solutions to school timetabling problems. The h-HCCA genetic algorithm is used by Nurmi et al. [16] to evolve timetables for Finnish schools. This GA incorporates the use of hill-climbing in the mutation operator and simulated annealing to select timetable periods to allocate tuples to. The GA implemented by Zuters et al. [17] uses a neural network to calculate the fitness of the population. A combination of genetic algorithms and a non-random ascent method (RNA) produced better results in solving a set of high school timetabling problems than applications of these methods separately [18].

3 The Two-Phased GA Approach

A two-phased approach is taken in solving the school timetabling problem. The first phase uses a genetic algorithm to produce feasible timetables (Phase I), the quality of which is improved in the second phase by a second genetic algorithm (Phase II). Trial runs conducted revealed that a two-phased approach, with different GAs dealing with hard and soft constraints, was more effective than using a single GA to evolve both feasible and good quality timetables. Previous work [1] applying genetic algorithms to solving the examination timetabling problem has also revealed the effectiveness of a two-phased approach, with each phase employing different GAs to optimize hard and soft constraints.

Both GAs begin by creating an initial population of individuals, i.e. timetables, which are iteratively improved over successive generations with respect to either feasibility or quality. The number of individuals remains constant over all generations. Each successive generation involves evaluation of the population, selecting parents and applying mutation operators to the parents to create the next generation. The stopping criterion for both GAs is a set number of generations. The processes of initial population generation, evaluation, selection and regeneration are described in the following subsections.

3.1 Initial Population Generation

A majority of the studies in section 2 have used a matrix representation for each chromosome. Thus, in this study each element of the population is also a matrix representing a school timetable with each row corresponding to a timetable period and each column a class to be taught. The teacher teaching the class in the particular period (and the venue in which the lesson is to be taught if venue allocation is part of the problem) is stored at the intersection of each row and column.

The requirements, i.e. class-teacher meetings of a problem are defined in terms of class-teacher or class-teacher-venue (if venue allocation is included) tuples. For example, (C1,T4) is a tuple indicating that teacher T4 must teach class C1 and (C3,T1,V1) specifies that class C3 must be taught by teacher T1 in venue V1. If teacher T4 has to meet with class C1 five times in the school week, (C1, T4) will occur five times in the list of tuples to be allocated.

Initially, the timetables of the population of the first generation of the GA for Phase I were created by randomly allocating tuples to timetable periods. However, this is not very effective as the search space represented by the initial population was too large. This led to the derivation of a sequential construction method (SCM) to create each element of the initial population. The SCM creates n timetables. The most appropriate value for n is problem dependant. Each timetable is created by sorting the tuples to be allocated to the timetable according to the difficulty of scheduling the tuple. Low-level construction heuristics are used to assess this difficulty. Each tuple is scheduled in a feasible timetable period, i.e. a period to which the tuple can be allocated without resulting in any hard constraint violations. If there is more than one feasible period available the tuple is allocated to the minimal penalty period, i.e. the period which produces the lowest soft constraint cost. If more than one minimal penalty period exists, a period is randomly selected from these. If there are no feasible

periods available the tuple is scheduled in a randomly selected slot. Each timetable is evaluated and its fitness is determined. In Phase I the fitness is the number of hard constraints violated. The SCM returns the fittest of the *n* timetables. If there is more than one timetable with the same fitness, the soft constraint cost is used as a secondary measure.

One of the contributions of this work is the identification of a set of low-level construction heuristics that can be used to measure the difficulty of scheduling a tuple. Low-level construction heuristics generally used for the university examination and course timetabling problems are the graph colouring heuristics largest degree, largest colour degree, largest weighted degree, largest enrollment and saturation degree [1]. Due to the differences in these problems and the school timetabling problem the largest colour degree, largest weighted degree and largest enrollment are not relevant to the STP. The largest degree and saturation degree have been adapted for the STP and other low-level construction heuristics have been identified for this domain. The following low-level heuristics have been defined for this purpose:

- Random – In this case a construction heuristic is not used and tuples to be allocated are randomly chosen from the list of unscheduled tuples.
- Largest degree – Tuples with a larger number of class-teacher meetings are scheduled first. Once a tuple is allocated the largest degree of the remaining tuples with the same class and teacher (and venue if applicable) is reduced by one. For example, suppose that teacher T3 is required to meet class C1 in venue V4 four times a week. There will be four occurrences of the tuple (C1, T3, V4) in the list of tuples to be allocated and all four occurrences will have a largest degree of 4. Suppose one occurrence is scheduled, leaving three occurrences in the list of unscheduled tuples. The largest degree of three remaining tuples will be reduced by one giving each occurrence a largest degree of 3.
- Saturation degree – The saturation degree of a tuple is the number of feasible, i.e. a period that will not result in hard constraint violations if the tuple is scheduled in it, timetable periods which the tuple can be scheduled in at the current point of the construction process. Tuples with a lower saturation degree are given priority. At the beginning of the timetable construction process all tuples have the same saturation degree, i.e. the number of timetable periods for the problem. For example, suppose that the tuple (C1,T3) has been allocated. The saturation degree of all tuples containing either C1 and/or T3 will be reduced by one.
- Class degree – Tuples containing a class that is involved in the most class-teacher meetings is given priority.
- Teacher degree – Tuples containing the teacher involved in the most number of class-teacher meetings are given priority.
- Consecutive periods – Tuples that need to be scheduled in consecutive periods, i.e. doubles and triples, are given priority and scheduled first.
- Sublclass/co-teaching degree – Tuples that have co-teaching or subclass requirements are given priority and allocated to the timetable before the other tuples.

- Period preferences – Tuples that have to be scheduled in specific periods are scheduled first and hence given priority over the other tuples. For example, if all Mathematics lessons must be scheduled within the first four periods for certain grades all the tuples for these lessons will be given priority.
- Teacher availability – Tuples containing teachers that are available for the least number of days are given priority.

One of these low-level heuristics is usually used to sort tuples. Alternatively, a combination of low-level heuristics can be applied to sort the list of tuples. In this case a primary heuristic and one or more secondary heuristics can be used for sorting purposes. For example, if saturation degree is employed as a primary heuristic and period preferences as a secondary heuristic, the tuples will firstly be sorted in ascending order according to the saturation degree. If two tuples have the same saturation degree, the tuples with a larger number of period preferences will be scheduled first. The initial population of the GA in Phase II is the population of the last generation of Phase I. All the timetables in this population are usually feasible.

3.2 Evaluation and Selection

Evaluation of the population on each generation involves calculating a fitness measure for each individual, i.e. timetable. The fitness of a timetable is the number of hard constraint violations in Phase I and the number of soft constraint violations in Phase II. Thus, in both phases we aim to minimize the fitness of an individual. The fitness of the elements of the population is used by the selection method to choose the parents of the next generation.

The tournament selection method is used to select parents. This method randomly selects t elements of the population where t is referred to as the tournament size. The element of the tournament with the best fitness, i.e. the lowest fitness measure, is returned as a parent.

During trial runs a variation of the tournament selection method, called a sports tournament method, proved to be more effective in the evolution of solutions to the school timetabling problem than the standard tournament selection method. The pseudo code for the sports tournament selection is depicted in Figure 3. The selection method takes an analogy from sport such as cricket where the best team may not always win. Instead of always returning the fittest element of the tournament this method firstly randomly selects the first element of the tournament and in comparing the successive elements of the tournament randomly decides to leave the *current_champion* unchanged, replace the *current_champion* with the contender, even if the contender is not fitter, or replace the *current_champion* with the contender if the contender is fitter (standard tournament selection). The two-phased GA approach will use either the tournament or sports tournament selection for both GAs of both phases and the choice of selection method is problem dependant.

3.3 Regeneration

One or more mutation operators are applied to chosen parents to create the offspring for each generation. Section 3.3.1 presents the mutation operators used by the GA in

Phase 1 and section 3.3.2 those used by the GA in Phase 2. A certain percentage of mutation operations are usually reduced to reproduction, i.e. the offspring is a copy of the parent. Thus the reproduction operator is not used to reduce the possibilities of cloning. Previous studies have found the use of a crossover operator usually results in violation of the problem requirements, e.g. allocation of the same tuple to the same period. Thus, application of the crossover operator is usually followed by a repair mechanism being applied to rectify the side effects [7, 9]. This is time consuming and results in an increase in runtimes. Hence, Bedoya et al. [8] do not implement a crossover operator. The same approach is taken in this study.

3.3.1 Phase 1 Operators

The following three mutation operators are available for the GA for Phase 1:

- Double violation mutation (2V) – This operator locates two tuples assigned to periods which have resulted in hard constraint violations and swaps these tuples. This swap may result in no change in the fitness of the timetable, i.e. the swap has not removed the violations or may improve the fitness by resulting in one or both of the violations being eliminated.
- Single violation mutation (1V) – This mutation operator selects a tuple causing a hard constraint violation and swaps it with a randomly selected tuple. This could result in a further violation worsening the fitness. Alternatively, the swap may remove the constraint violation improving the fitness of the timetable or have no effect.
- Random swap – This operator selects two tuples or two sets of consecutive tuples randomly and swaps the locations of the tuples or sets in the timetable.

Each of these operators performs s swaps and the best value for s is problem dependant. Versions of these operators incorporating hill-climbing is also available. The hill-climbing versions of these operators continue mutating the parent until an offspring fitter than the parent is produced. In order to prevent premature convergence of the GA and long runtimes, a limit l is set on the number of attempts at producing a fitter individual. If this limit is reached the last offspring created is returned as the result of the operation. The performance of the different mutation operators with and without the incorporation of hill-climbing will be tested for the different school timetabling problems. This is discussed in section 4.

3.3.2 Phase 2 Operators

This section describes the four mutation operators that are used by the GA in Phase 2 of the approach. As in the first phase, each mutation operator performs s swaps, with the best value for s being problem dependant. Swaps producing hard constraint violations are not allowed. The four mutation operators for Phase 2 are:

- Random swap – This operator randomly selects two tuples and swaps their positions in the timetable.
- Row swap - Two rows in the timetable are randomly selected and swapped, changing the period that the tuples in both the rows are scheduled in.
- Double violation mutation – Two tuples causing soft constraint violations are chosen and swapped. This can have no effect on the fitness or improve the fitness by eliminating one or both of the violations.

- Single violation mutation – The position of a tuple causing a soft constraint violation is swapped with that of a randomly selected tuple. As in the first phase this could result in a further violation, have no effect or remove the soft constraint violation.
- Subclass+co - teaching row swap (1VSRS) – The row containing a tuple that is violating a subclass or co-teaching constraints is swapped with another row.

As in the first phase, versions of these operators including the use of hill-climbing are also implemented. In this case the mutation operator is applied until an offspring at least as fit as the parent is produced. Again to prevent premature convergence and lengthy runtimes a limit is set on the number of attempts at producing such an offspring.

4 Experimental Setup

This section describes the school timetabling problems that the GA approach presented in the previous section is evaluated on, the genetic parameter values used and the technical specifications of the machines the simulations were run on.

4.1 School Timetabling Problems

The school timetabling problem varies from school to school due to the different educational systems adopted by different countries. Thus, there are different versions of the school timetabling problem. In order to thoroughly test the two-phased GA approach and to evaluate it in a South African context, the approach was applied to four school timetabling problems:

- A set of hard benchmark school timetabling problems
- The Greek high school timetabling problem
- A South African primary school timetabling problem
- A South African high school timetabling problem

Each of these problems is described in the following subsections.

4.1.1 Benchmark Timetabling Problems

Abramson [7] has made available five artificial timetabling problems [19]. These problems are "hard" timetabling problems (hence the *hdtt*) as all periods must be utilized with very little or no options for each allocation. The characteristics of the problems are listed in Table 1. Each school week is comprised of five days with six periods a day with a total of 30 timetable periods.

Table 1. Characteristics of the artificial school timetabling problems

Problem	Number of teachers	Number of Venues	Number of Classes
hdtt4	4	4	4
hdtt5	5	5	5
hdtt6	6	6	6
hdtt7	7	7	7
hdtt8	8	8	8

All five problems have the following hard constraints:

- All class-teacher-venue tuples must be scheduled the required number of times.
- There must be no class clashes, i.e. a class must not be scheduled more than once in a period.
- There must be no teacher clashes, i.e. a teacher must not be scheduled more than once in a period.
- There must be no venue clashes, i.e. a venue must not be allocated more than once to a timetable period.

4.1.2 The Greek School Timetabling Problem

The GA approach is applied to two Greek school timetabling problems, namely, that made available by Valouxis et al. [20] and Beligiannis et al. [21]. The problem presented by Valouxis et al. involves 15 teachers and 6 classes. There are 35 weekly timetable periods, i.e. 5 days with 7 periods per day. The hard constraints of the problem are:

- All class-teacher meetings must be scheduled.
- There must be no class or teacher clashes.
- Class free/idle periods must be scheduled in the last period of the day.
- Each teacher's workload limit for a day must not be exceeded.
- Class-teacher meetings must be uniformly distributed over the school week.

The soft constraints for the problem are:

- The number of free periods in the class timetable must be minimized.
- Teacher period preferences must satisfied if possible.

The GA approach is also tested on six of the problems made available by Beligiannis et al. [21]. The characteristics of these problems are depicted in Table 2. There are 35 timetable periods per week.

Table 2. Characteristic of the Beligiannis Problem Set

Problem	Number of Teachers	Number of Classes	Number of Co-Teaching/Sublcass Requirements
HS1	11	34	18
HS2	11	35	24
HS3	6	19	0
HS4	7	19	12
HS5	6	18	0
HS6	13	35	20

The hard constraints for the problem are:

- All class-teacher meetings must be scheduled.
- There must be no class or teacher clashes.
- Teachers must not be scheduled to teach when they are not available.

- Class free/idle periods must be scheduled in the last period of the day.
- Co-teaching and subclass requirements must be met.

The problem soft constraints are:

- The number of idle/free periods for teachers must be minimized.
- Free periods must be equally distributed amongst teachers.
- The workload for a teacher must be uniformly distributed over the week.
- Classes should not be taught the same subject in consecutive periods or more than once in a day if possible.

4.1.3 South African Primary School Problem

This problem involves 19 teachers, 16 classes and 14 subjects. There are a maximum of 11 weekly timetable periods. However, different grades have a different number of daily periods ranging from 9 to 11. The hard constraints for the problem are:

- All required class-teacher meetings must be scheduled.
- There must be no class or teacher clashes.
- Certain subjects must be taught in specialized venues, e.g. Technology in the computer laboratory.
- Mathematics must be taught in the mornings (specified in terms of valid periods).
- All co-teaching requirements must be met.
- All double period requirements must be met.

The problem has one soft constraint, namely, the lessons per class must be uniformly distributed throughout the school week.

4.1.4 South African High School

The South African high school problem that the GA approach is applied to involves 30 classes, 40 teachers and 44 subjects. The hard constraints for the problem are:

- All required class-teacher meetings must be scheduled.
- There must be no class or teacher clashes.
- All sub-class and co-teaching requirements must be met.

The soft constraints for the problem are:

- Teacher period preferences must be met if possible.
- Period preferences for classes must be met if possible.

4.2 Genetic Parameter Values

Trials runs were conducted to determine the most appropriate values for the following genetic parameters:

- SCM population size (n) – The SCM is used to create each element of the population. It creates n timetables, the fittest of which is included in the GA population of Phase I.
- GA population size
- Number of generations
- Tournament size
- Number of mutation swaps
- Number of generations

Table 3 lists the values tested for each of these parameters.

Table 3. Ranges for each parameter value

Parameter	Tested range	Note:
SCM size	1 to 100	Only applicable in Phase 1
Population size	200 to 1000	Constant population size adopted for every generation
Tournament size	5 to 20	Applicable to tournament selection for Phase 1 and Phase 2
Swaps	20 to 200	Applicable to mutation operators for Phase 1 and Phase 2
Generations	20 to 75	Applicable to Phase 1 and Phase 2

When testing each parameter value, 30 runs were performed. In order to test the impact that each parameter has on the performance of the genetic algorithm, all other parameter values, construction heuristics, selection methods and genetic operators were kept constant. The most appropriate values found for each problem are listed in Table 4.

Table 4. Parameter values for each data set

Problem	SCM	Population Size	Tournament Size	Swaps per Mutation	Genera-tions
HDTT4	50	1000	10	200	50
HDTT5	50	1000	10	200	50
HDTT6	50	1000	10	200	50
HDTT7	50	1000	10	200	50
HDTT8	50	1000	10	200	50
Valouxis	50	1000	10	100	50
HS1 – HS4, HS6	25	750	15	200	50
HS5	50	750	10	20	75
Lewitt	20	500	10	200	50
Woodlands	20	750	10	150	75

4.3 Technical Specifications

The GA system was developed using Visual C++ 2008. The random number generator function available in C++ is used to generate random numbers. A different seed is

used for each run of the genetic algorithm approach. Simulations (trial and final) were run on several machines:

- Intel Core 2 Duo CPU @ 2.40 GHz, 2.00 GB RAM, Windows XP, Windows 7 Enterprise OS.
- Intel Core I7 870 CPU @ 2.93 GHz, 4.00 GB RAM, Windows 7 64-bit OS.
- Intel Core I7 860 CPU @ 2.80 GHZ, 4.00 GB RAM (3.49 Usable), Windows 7 32-bit OS.
- Pentium Dual Core @ 2GHZ, 2.00 GB RAM, Windows XP.

5 Results and Discussion

The two-phased genetic algorithm approach was able to evolve feasible solutions of good quality for all problems. Different combinations of construction heuristics, selection method and genetic operators were found to produce the best quality solution for each problem. The GA approach was run using different combinations of these components. In order to test the impact that each component has on the performance of each genetic algorithm, all other genetic algorithm components and parameter values are kept constant. Thirty runs were performed for each component. The statistical significance of the performance of the different construction heuristics, selection methods and genetic operators was ascertained using hypothesis tests[1] (tested at the 1%, 5% and 10% levels of significance). The combination producing the best result for each problem is listed Table 5. Note that if hill-climbing was used with the genetic operator this is indicated by HC and if it was not used by NH.

The use of saturation degree as a primary heuristic produced the best results for all except one problem. A secondary heuristic was needed for all of the real world problems especially problems involving subclass and co-teaching constraints. For the Abramson data set double violation mutation without hill-climbing appears to be the most effective during Phase 1. For the real world problems single violation mutation with hill-climbing produced the best results for a majority of the problems. Hill-climbing was not needed in Phase 2 to produce the best soft constraint cost for any of the problems with single violation mutation proving to be the most effective for a majority of the problems. The sports tournament selection method appears to be effective in the GA implemented in Phase 1 focused on optimizing the hard constraint costs while the standard tournament selection appears to have produced better results in Phase 2, which improves the quality of timetables, for most of the problems. It is evident from Table 5 that different combinations of low-level constructive heuristics, selection method and mutation operators is needed to solve each problem. Future work will investigate whether there is a correlation between the architecture of the GAs of each phase and the characteristics of the different problems as well as methods for the automatic configuration of the GA architectures of both phases for the school timetabling problem.

[1] Throughout the paper hypothesis tests conducted test that the means are equal and the Z test is used.

Table 5. Summary of best heuristics, methods and operators for each data set

	PHASE 1				PHASE 2	
Problem	**Primary Heuristic**	**Secondary Heuristics**	**Selection Method**	**Genetic Operators**	**Selection Method**	**Genetic Operator**
HDTT4	Saturation Degree	None	Std/Sports	2VNH	N/A	N/A
HDTT5	Saturation Degree	None	Sports	2VNH	N/A	N/A
HDTT6	Saturation Degree	None	Sports	2VNH	N/A	N/A
HDTT7	Saturation Degree	None	Sports	2VNH	N/A	N/A
HDTT8	Saturation Degree	None	Standard	2VNH	N/A	N/A
Valouxis	Saturation Degree	Teacher Degree Teacher availability	Sports	1VHC	Sports	Random Swap
HS1	Saturation Degree	SubClass/Co-Teaching degree	Sports	1VHC	Standard	Single Violation
HS2	Saturation Degree	SubClass/Co-Teaching degree	Sports	1VHC	Standard	Single Violation
HS3	Saturation Degree	SubClass/Co-Teaching degree	Sports	1VHC	Standard	Single Violation
HS4	Saturation Degree	SubClass/Co-Teaching	Sports	1VHC	Sports	Single Violation
HS5	Largest Degree	SubClass/Co-Teaching degree	Sports	1VNH	Standard	Random Swap
HS7	Saturation Degree	SubClass/Co-Teaching degree	Sports	1VHC	Standard	Single Violation
Lewitt	Saturation Degree	Consecutive Periods	Standard	Hybrid (2VHC, 1VHC, Random Swap)	Sports	Random Swap
Woodlands	Saturation Degree	SubClass/Co-Teaching degree	Standard	1VHC	Standard	1VSRS

The performance of the GA approach was compared to other methods applied to the same set of problems. For the first set of problems, namely, the benchmark hard problems made available by Abramson [7], the GA approach was compared to the following:

- SA1 – A simulated annealing method implemented by Abramson et al. [22].
- SA2 – A simulated annealing algorithm implemented by Randall [23].
- TS – A tabu search employed by Randall [23].
- GS – The greedy search method used by Randall [23].
- NN-T2 – A neural network employed by Smith et al. [24].
- NN-T3 – A second neural network employed by Smith et al. [24].

The hard constraints for this set of problems are listed in section 4. The minimum (best cost - BC) and average (average cost – AC) hard constraint costs for each of these methods and the GA approach is listed in Table 6. In this study the average is taken over thirty runs. The best results are highlighted in bold. The GA approach has produced the minimum for all of the problems and the best average for three of the problems. For the remaining two problems, the average obtained is very close to the best results.

Table 6. Comparison for the Abramson Data Set

Method	HDTT4	HDTT5	HDTT6	HDTT7	HDTT8
SA1	BC: Unknown AC: Unknown	BC: 0 AC: 0.67	BC: 0 AC: 2.5	BC: 2 AC: 2.5	BC: 2 AC: 8.23
SA2	**BC: 0** **AC: 0**	BC: 0 AC: 0.3	BC: 0 AC: 0.8	BC: 0 AC: 1.2	BC: 0 AC: 1.9
TS	BC: 0 AC: 0.2	BC: 0 AC: 2.2	BC: 3 AC: 5.6	BC: 4 AC: 10.9	BC: 13 AC: 17.2
GS	BC: 5 AC: 8.5	BC: 11 AC: 16.2	BC: 19 AC: 22.2	BC: 26 AC: 30.9	BC: 29 AC: 35.4
HNN1	BC: 0 AC: 0.1	BC: 0 AC: 0.5	BC: 0 AC: 0.8	BC: 0 AC: 1.1	BC: 0 AC: 1.4
HNN2	BC: 0 AC: 0.5	BC: 0 AC: 0.5	BC: 0 AC: 0.7	**BC: 0** **AC: 1**	**BC: 0** **AC: 1.2**
GA approach	**BC: 0** **AC: 0**	**BC: 0** **AC: 0**	**BC: 0** **AC: 0**	BC: 0 AC: 1.067	BC: 0 AC: 1.733

The GA approach was also applied to the school timetabling problem presented by Valouxis et al. [20]. In the study conducted by Valouxis et al. constraint programming was used to solve this problem. The timetables induced by both methods were run through an evaluator developed by the authors which assessed the hard and soft constraint costs. Feasible timetables were produced by both methods. The timetable produced by constraint programming had 45 soft constraint violations while that produced by the GA approach had 35.

The timetables generated by the evolutionary algorithm implemented by Beligiannis et al. [21] are compared to those produced by the GA approach. Again an evaluator developed by the authors was used to assess the hard and soft constraint cost of all timetables for comparison purposes. Both methods produced feasible timetables for the 6 problems tested. The soft constraint costs of the timetables are listed in Table 7.

Table 7. Comparison with the Beligannis data set [21]

Problem	Evolutionary Algorithm	GA Approach
HS1	139	96
HS2	175	99
HS3	61	34
HS4	102	59
HS5	43	40
HS6	226	117

The timetable used by the South African primary school is induced by a package. The timetable produced by the package is manually changed to meet the hard and soft constraints. The current timetable used by the school does not meet all the double period requirements while the best timetable evolved by the GA approach satisfies these. The best timetable produced by the GA for the South African high school problem is a feasible timetable and has the same soft constraint cost, namely a cost of two, as the timetable currently being used by the school. From the above comparisons it is evident that the performance of the GA approach is comparable and in some cases better, than other methodologies applied to the same problems.

6 Conclusion and Future Work

This study has presented a two-phased genetic algorithm approach for solving the school timetabling problem. In previous work a genetic algorithm was developed to solve a particular problem whereas this study has evaluated genetic algorithms as a means of solving different school timetabling problems. The paper has also defined low-level construction heuristics for this domain. The performance of a methodology on a variety of problems is important as the school timetabling problem varies drastically from one school to the next. The two-phased genetic programming approach was tested on four different types of problem sets involving a total of 13 different problems. This approach was able to produce feasible timetables for all problems. The soft constraint cost of these timetables were found to be comparable to and in some cases better than other methodologies applied to the same problems. Different combinations of genetic algorithm components, namely, construction heuristics, selection methods and genetic operators were needed to produce the best results for the different problems. Thus, future work will focus on identifying the correlation between different combinations and problem characteristics and methods for the automatic configuration of the GA architecture for both phases of the GA approach in solving the school timetabling problem. This research will investigate the use of case-based reasoning and an evolutionary algorithm, to explore a space of strings representing the GA components to find the optimal combination, as options for automatic GA architecture configuration. The study has also revealed that GAs can successful solve both the South African primary and high school timetabling.

References

1. Pillay, N., Banzhaf, W.: An Informed Genetic Algorithm for the Uncapacitated Examination Timetabling Problem. Applied Soft Computing 10, 45–67 (2010)
2. Larranaga, P., Kuijpers, C.M.H., Murga, R.H., Inza, I., Dizdarevic, S.: Genetic Algorithms for the Travelling Salesman Problem: A Review of Representations and Operators. Artificial Intelligence Review 11(2), 129–170 (1999)
3. Ponce-Perez, A., Perez-Garcia, A., Ayala-Ramirez, V.: Bin-Packing Using Genetic Algorithms. In: Proceedings of CONIELECOMP 2005: 15th International Conference on Electronics, Communications and Computers, pp. 311–314. IEEE Press (2005)
4. Pillay, N.: A Survey of School Timetabling. Annals of Operations Research (February 2013), doi:10.1007/s10479-013-1321-8
5. Goldberg, D., Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley Longman Publishing Co. (1989).
6. Beasley, D., Bull, D.R., Martin, R.R.: An Overview of Genetic Algorithms: Part 1 and Part 2, Research Topics. University Computing 15(4), 170–181 (1993)
7. Abramson, D., Abela, J.: A Parallel Genetic Algorithm for the Solving the School Timetabling Problem. In: Proceedings of the Fifteenth Australian Conference: Division of Information Technology, C.S.I.R.O. pp. 1–11 (1991)
8. Bedoya, C.F., Santos, M.: A Non-Standard Genetic Algorithm Approach to Solve Constrained School Timetabling Problems. Eurocast, 26–37 (2003)
9. Calderia, J.P., Ross, A.C.: School Timetabling Using Genetic Search. In: The Proceedings of the International Conference on the Practice and Theory of Automated Timetabling (PATAT 1997) pp. 115-122 (1997)
10. Colorni, A., Dorigo, M., Maniezzo, V.: Metaheuristics for High School Timetabling. In: Computational Optimization and Applications, vol. 9, pp. 275–298. Kluwer Academic Publishers (1998)
11. Filho, G.R., Lorena, L.A.N.: A Constructive Evolutionary Approach to School Timetabling. In: Boers, E.J.W., Gottlieb, J., Lanzi, P.L., Smith, R.E., Cagnoni, S., Hart, E., Raidl, G.R., Tijink, H. (eds.) EvoWorkshop 2001. LNCS, vol. 2037, pp. 130–139. Springer, Heidelberg (2001)
12. Wilke, P., Gröbner, M., Oster, N.: A Hybrid Genetic Algorithm for School Timetabling. In: McKay, B., Slaney, J.K. (eds.) AI 2002. LNCS (LNAI), vol. 2557, pp. 455–464. Springer, Heidelberg (2002)
13. Yigit, T.: Constraint-Based School Timetabling Using Hybrid Genetic Algorithms. In: Basili, R., Pazienza, M.T. (eds.) AI*IA 2007. LNCS (LNAI), vol. 4733, pp. 848–855. Springer, Heidelberg (2007)
14. Beligiannis, G.N., Moschopoulos, C.N., Likothanassis, S.D.: A Genetic Algorithm Approach to School Timetabling. Journal of the Operational Research Society 60(1), 23–42 (2009)
15. Srndic, N., Dervisevic, M., Pandzo, E., Konjicija, S.: The Application of a Parallel Genetic Algorithm to Timetabling of Elementary School Classes: A Coarse Grained Approach. In: Proceedings of ICAT 2009 -2009 22nd International Symposium on Information, Communication and Automation Technologies, pp. 1–5. IEEE (2009)
16. Nurmi, K., Kyngas, J.: A Framework for School Timetabling Problem. In: Proceedings of the 3rd Multidisciplinary International Scheduling Conference: Theory and Application (2007)

17. Zuters, J.: Neural Networks to Enrich Fitness Function in a GA-Based School Timetabling Model. Proceedings of WSEAS Transactions on Information Science and Application 4(2), 346–353 (2007)
18. Cedeira-Pena, A., Carpente, L., Farina, A., Seco, D.: New Approaches for the School Timetabling Problem. In: Proceedings of the 7th Mexican Conference on Artificial Intelligence (MICAI 2008), pp. 261–267 (2008)
19. Beasley, J.F.: OR Library, `http://people.brunel.ac.uk/mastjjb/jeb/orlib/tableinfo.html` (accessed May 25, 2011)
20. Valouxis, C., Housos, E.: Constraint Programming Approach for School Timetabling. Computers and Operations Research 30, 1555–1572 (2003)
21. Beligiannis, G.N., Moschopoulos, C.N., Kaperonis, G.P., Likothanassis, S.D.: Applying Evolutionary Computation to the School Timetabling Problem: The Greek Case. Computers and Operations Research 35, 1265–1280 (2008)
22. Abramson, D., Dang, H.: School Timetable: A Case Study in Simulated Annealing. In: Applied Simulated Annealing Lecture Notes in Economics and Mathematical Systems, ch. 5, pp. 103–124 (1993)
23. Randall, M.: A General Meta-Heuristic Based Solver for Combinatorial Optimization Problems. Computational Optimization and Applications 20(2), 185–210 (2000)
24. Smith, K.A., Abramson, D., Duke, D.: Hopfield Neural Networks for Timetabling: Formulations, Methods, and Comparative Results. Computers and Industrial Engineering 44, 285–305 (2003)