An Informed Genetic Algorithm for the High School Timetabling Problem

Rushil Raghavjee
School of Information Systems
University of KwaZulu-Natal
KwaZulu-Natal, South Africa
raghavjee@ukzn.ac.za

Nelishia Pillay
School of Computer Science
University of KwaZulu-Natal
KwaZulu-Natal, SouthAfrica
pillayn32@ukzn.ac.za

ABSTRACT
The high school timetabling problem differs drastically from one school to another and from country to country. The South African high school problem has not been researched. This paper presents a genetic algorithm (GA) to solve this problem for a particular high school. A two-phase approach is taken. The first phase uses a GA to evolve a timetable that meets the hard constraints of the problem. During the second phase a GA improves the quality of the solutions found during the first phase by reducing the soft constraint cost of the timetable. Domain knowledge, in the form of low-level construction heuristics, is used to guide the search during the first phase. The study experiments with the effect of using different low-level construction heuristics for this purpose. Each GA iteratively refines an initial population from one generation to the next by the processes of evaluation, selection and regeneration. The paper also reports on the performance of different mutation operators tested for regeneration.

Categories and Subject Descriptors
I.2 [**Computing Methodologies**]: Artificial Intelligence.

General Terms
Algorithms, Theory.

Keywords
School timetabling, genetic algorithms.

1. INTRODUCTION
The high school timetabling problem essentially involves allocating class, teacher, and venue tuples to timetable slots so as to meet the hard constraints of the problem and minimize the soft constraints [1]. The hard constraints must be met in order for the timetable to be operable. A timetable satisfying the hard constraints is referred to as a feasible timetable.

A common hard constraint is that there must be no class, teacher and venue clashes, i.e. a class, teacher and venue should not be scheduled more than once during a single period. Another example of a hard constraint is that certain classes must take place in certain venues, e.g. science, computer classes. Soft constraints are characteristics that we would like the timetable meet, e.g. teachers prefer certain periods, mathematics should be taught in the morning. It is usually not possible to meet all soft constraints and the soft constraint cost is minimized. The soft constraint cost is treated as a measure of the quality of the timetable. Thus, each problem is described in terms of a set of requirements, e.g. the number of class-teacher meetings, hard constraints and soft constraints.

Various techniques have been used to solve the school timetabling problem. These include simulated annealing ([2], [3]), evolutionary algorithms ([4], [5], [6], [7], [8], [9], [10]). a Hopfield neural network ([11]), Tabu search ([12], [13], [14]), integer programming ([15], [16]), constraint programming [17], GRASP [18], and tiling algorithms [19].

The education system and thus the high school timetabling problem varies drastically from one country to another ([20], [21]). In some versions of the problem room allocation is part of the problem, i.e. the venue is not part of the tuple to be allocated. Some schools extend over more than one site. Furthermore, the number and duration of lessons for different grades differs from school to school. The high school timetabling problem has been researched for a number of countries [1] and data sets are available for Australia, Brazil, Finland, Greece, Netherlands, and Italy [21]. The study presented in this paper investigates solving the high school problem for a South African high school. An informed genetic algorithm (IGA) is applied to the problem. The algorithm is referred to as an "informed" GA as domain knowledge in the form of heuristics is used to guide the search process. The main contribution of the study is the evaluation of an informed genetic algorithm to solving the South African high school problem.

The following section provides an overview of genetic algorithms and discusses previous work applying evolutionary algorithms to the school timetabling problem. The South African high school problem that the IGA is used to solve is described in section 3. Section 4 presents the informed genetic algorithm. Section 5 specifies technical details of the implementation. Section 6 analyzes the performance of the IGA in solving this problem. The findings of the study and future work are summarized in section 7.

2. GENETIC ALGORITHMS AND PREVIOUS WORK
Genetic algorithms were introduced by John Holland in the seventies [22]. These algorithms are based on Darwin's theory of evolution. They begin with an initial population of potential solutions which are iteratively refined from one generation to the next until a solution is found or a preset number of generations have been completed. Each generation involves selecting parents and applying genetic operators to the chosen parents to create the offspring of the generation. Fitness proportionate selection or tournament selection is used to

SAICSIT '10, October 11–13, 2010, Bela Bela, South Africa.

choose parents. Genetic operators usually used for recreation are reproduction, crossover and mutation.

Genetic algorithms and other evolutionary algorithms have been applied to the school timetabling problem. Abramson et al. [4] apply a parallel genetic algorithm to nine highly constrained timetabling problems. Calderia et al. [5] use a genetic algorithm to solve a randomly generated school timetabling problem. A similar approach is taken by Filho et al. [6] in which a constructive genetic algorithm is used to solve the school timetabling problem for two Brazilian high schools. Wilke et al. [7] apply a genetic algorithm to the German high school timetabling problem. Raghavjee et al. [8] apply a genetic algorithm to five difficult school timetabling problems presented by Smith et al. [11]. A variation of tournament selection is used to choose the parents of each offspring. Beligiannis et al. [9] use an adaptive evolutionary algorithm to solve the Greek high school timetabling problem. Linear ranking together with elitism is used to select parents to which a mutation operator is applied. Raghavjee et al. [10] apply a genetic algorithm to the Greek school data set used by Beligiannis et al. [9]. A two-phase approach is taken to solving the problem. An initial GA is used to evolve feasible solutions. The quality of these solutions is then improved using a second GA.

3. A SOUTH AFRICAN HIGH SCHOOL TIMETABLING PROBLEM

This section describes the high school timetabling problem that the IGA is used to solve The school consists of 30 classes, grades 8 to 12, 40 teachers and a total of 44 subjects. There are a total of 42 periods, i.e. 7 periods 6 times a week. School timetabling problems are defined in terms of the problem requirements, hard constraints and soft constraints. Rooms are pre-allocated and thus the problem does not require venues to be scheduled. The requirements of the problem specify:

- The number of meetings required for each class-teacher pair.
- Which classes should be split and re-grouping of classes. For example, classes of a particular grade may be split and regrouped. Teachers involved are also specified.

The hard constraints of the problem are:

- No teacher or class clashes. i.e. a teacher cannot be allocated to more than one class at a time, unless this is during a split period. A class cannot be allocated to more than one teacher at a particular time.
- Split periods – All split period requirements must be met. When classes are broken up and placed into different groups, the single split occurrence must occur during a single period.

The soft constraints of the problem are:

- Normal or class constraint – This involves the scheduling or non-scheduling of a class in a particular period on the timetable.
- Teacher constraint – This involves the scheduling or non-scheduling of a teacher in a particular period on the timetable.
- Combination constraint – This involves the scheduling or non-scheduling of a combination or split of classes in a particular period on the timetable.

Table 1 and Table 2 list the number of occurrences of each type of hard constraint and soft constraint in the problem requirements respectively. Note that these values were obtained from the timetable requirements provided by the school.

Table 1. Hard constraint occurrences

Hard Constraints	Number of Occurrences
Possible teacher and class clashes	212
Number of split periods	66
Number of class-teacher meetings to schedule	278

Table 2. Soft constraint occurrences

Soft Constraints	Number of Occurrences
Number of classes to be scheduled in a particular period.	8
Number of teachers to be scheduled in a particular period.	1
Number of split/combination classes to be scheduled in a particular period.	6

4. THE INFORMED GENETIC ALGORITHM

This section describes the informed genetic algorithm approach used to solve the school timetabling problem for a South African high school. The problem is described in section 3. The research conducted by Pillay et al. [23] in the domain of examination timetabling has revealed that a single GA solving both the hard and soft constraints of the problem is not very successful and a two-phase approach is more appropriate. Thus a two-phase approach is taken in solving the school timetabling problem. During the first phase a GA is applied to produce feasible solutions. A GA is applied in the second phase to improve the quality of the feasible solutions. Both phases employ the generation control model [23]. The overall genetic algorithm basically involves creating an initial population. This population is then evolved by performing the processes of evaluation, selection of parents (using tournament selection) and the application of the mutation operator to parents, iteratively until a present number of generations have been completed. These processes are described in the subsections that follow.

4.1 IGA for Phase I

Phase I focuses on producing timetables that satisfy the hard constraints of the problem. Thus, it aims to produce a solution with a hard constraint cost of zero. The subsections below describe the processes of initial population generation,

evaluation and selection and recreation of the GA employed during this phase. The processes of evaluation, selection and iteration are repeated for *g* generations, where *g* is a problem specific genetic parameter.

4.1.1 Initial Population Generation

Each element of the population is a matrix representing the timetable. The rows correspond to the periods and the columns to the class that each teacher will be allocated to. Each cell contains a teacher. In some cases there is a need to split and then regroup classes, for example in the case of teaching different languages. In these instances more than one teacher will be assigned to a cell. This will be referred to as a *multiple* class. Standard classes that are not split will be called *single* classes.

The method used to create each timetable is similar to that employed by Pillay et al. [24] for examination timetabling. Domain knowledge in the form of heuristics, which measure the difficulty of allocating each class-teacher tuple, is used during construction of the timetable. Three heuristics are used:

- Class split degree – classes involved in splits are given priority over classes that are not split.
- Saturation degree – class-teacher tuples with the least amount of feasible timeslots on the timetable at the particular point of construction is given priority. This heuristic is dynamic and has to be re-calculated after each allocation to the timetable.
- Teacher split degree – teachers with the most split requirements are given priority.

The requirements of the problem are broken down into a list of class-teacher tuples. If a teacher is meant to meet a class *m* times, the class-teacher tuple occurs *m* times in the requirement list. The list of requirements is sorted in order of difficulty with the class split degree as the primary key, saturation degree as a secondary key and the teacher split degree heuristic as a tertiary key. This ordering was determined during trial runs conducted to determine how best to incorporate the use of domain knowledge. Each tuple in the sorted list is allocated to a feasible slot on the timetable. After each allocation the saturation degree is re-calculated and the list is resorted. If a clash free slot is not found the tuple is allocated to the first free slot.

This initial population is then iteratively refined to further reduce the hard constraint cost in an attempt to find a feasible solution. The following subsection describes the methods of evaluation and selection used.

4.1.2 Evaluation and Selection

A fitness measure is calculated for each element of the population. In this phase the fitness function is the sum of the hard constraints violated and the requirements not allocated. Thus, an element with a lower fitness measure is a fitter individual. A variation of the tournament selection is used to select parents of the next generation. The standard tournament selection first creates a tournament of size *t*. The tournament is created by randomly choosing *t* individuals from the population. The fittest individual is returned as the winner of the tournament, i.e. a parent. At each stage of selection the new element of the tournament is compared with the best element thus far in the tournament, and if it is fitter it is marked as the best element. This was found to be too elitist and a variation in which the best element is not always chosen was used. One of three options is randomly chosen. The first option replaces the current best element of the tournament with the new element if the new element is fitter. The second option leaves the current best element unchanged and the third option replaces the current best element with the new element even if the fitness of the new element is not better. Selection is with replacement and an individual may be chosen more than once as a parent.

4.1.3 Recreation

The mutation operator is used to create each generation. The operator takes a single parent which is chosen using the variation of tournament selection described in the previous subsection. During trial runs different types of mutation were tested to determine which would be the most effective in exploring the solution space for the given domain. These include:

- Simple mutation – A cell, i.e. a period and class is randomly chosen. If the class is a single class, another cell involving a single class is randomly chosen. The contents of the cells are swapped. If the period contains a multiple class, the other classes involved in the split-regroup are also swapped to ensure that both regroups remain in the same period.
- Mutation with one clash – This mutation basically performs the same function as simple mutation, however at least one of the teachers swapped must be involved in a clash.
- Hill climber – This operator finds two teachers involved in a clash. If swapping the two teachers will result in an improvement in the fitness of the individual they are swapped, if not the swap is cancelled. The process of selecting two teachers involved in a clash and evaluating the swap of the classes is repeated until a swap reducing the fitness of the timetable is found. In order to ensure that the method is not too elitist a limit is set on the number of attempts to reduce the fitness. The number of attempts is treated as a genetic parameter and its value is problem dependant. There are two hill climbing operators, one for single classes and another for multiple classes.

These mutation operators were tested separately and in combination with each other. The use of the simple mutation in combination with the hill climber mutation operator for single classes produced the best results. The combined mutation operator is applied *s* times to the individual. *S* is a genetic parameter and its value is problem dependant. The operator randomly chooses whether to apply simple mutation or the single class hill climber. If the offspring is fitter than the parent the offspring replaces the parent and the process is repeated until the total number of mutations *s* is reached.

The following section describes the genetic algorithm for the phase II of the approach.

4.2 GA for Phase II

This phase is aimed at reducing the soft constraint cost of the feasible solution found in phase I. The initial population of the GA in phase II is the population of the final generation of the GA implemented in phase I. Trial runs have shown that all the timetables in the final population of the GA in phase I are feasible. The population is evolved for *g* generations, where *g* is a genetic parameter and its value is thus problem dependant.

The fitness of each timetable is the soft constraint cost, i.e. the sum of the soft constraints violated. The tournament selection is the same method employed by the GA in phase I. The same mutation operator is used as in phase I. The hill climber mutation is a variation of the one used in phase I. Instead of swapping just the teachers, the entire row containing the teacher is swapped. The overall operator is the same as that in the first phase. Note that if the offspring is infeasible the parent is not replaced.

5. EXPERIMENTAL SETUP

The IGA was written in Microsoft C++, Visual Studio 2008 and simulations were run on an Intel Dual-Core 1.86 GHz processor with 2 GB of memory and Windows Vista. The values of the genetic parameters for the GAs for both phases are listed in Table 1. These values were obtained empirically by performing trail runs.

Table 1. Values of the genetic parameters.

Parameter	Value
Initial population size (*p*)	100
Tournament size (*t*)	5
Number of mutation swaps(*s*)	45
Number of generations (*g*)	1000

6. RESULTS AND DISCUSSION

Five runs of the algorithm were performed. The informed genetic algorithm was able to evolve a feasible timetable for the high school problem, i.e. a hard constraint cost of zero. The minimum soft constraint cost obtained by the IGA is 6. This means that 9 soft constraints were satisfied. As we aim to minimize the soft constraint cost and usually do not obtain a soft constraint cost of zero, this cost is reasonable. As this is the first attempt at solving the problem for the particular problem, there are no previous results which the quality of the timetable can be compared to. Future work will investigate whether this cost can be reduced further. One option would be to assign each teacher (or teachers in the case of multiple classes), during timetable construction, to a period with the minimum penalty, i.e. the lowest soft constraint for the timetable at that point of construction, if there is more than one feasible period available. In addition to this, the soft constraint cost can also be taken into consideration when estimating the difficulty of scheduling class-teacher tuples. The average runtime of the IGA is approximately three hours. Future work will also investigate methods for reducing runtime and applying the IGA to additional school timetabling problems.

7. CONCLUSION AND FUTURE WORK

The study presented in this paper solves the school timetabling problem for a South African high school. An informed genetic algorithm is used for this purpose. The study has shown that domain knowledge, in the form of construction heuristics used during initial population for timetable construction, is needed to direct the search. In addition to this the use of hill-climbing during population recreation was found to be more effective than using standard mutation. Furthermore, a two phase approach was needed to solve the problem. The first phase employed a GA to produce a feasible timetable, while the second phase used a GA to improve the quality of the feasible timetables evolved during the first phase. The IGA was able to produce a feasible timetable, i.e. a timetable meeting all the hard constraints of the problem, with a reasonable soft constraint cost for the high school timetabling problem.

Future work will investigate whether it is possible to further reduce the soft constraint cost for the problem. The use of assigning class-teacher tuples to a minimum penalty cost period, and using an additional heuristic, namely one that focuses on soft constraints, in sorting class-teacher tuples for timetable construction will be examined for this purpose. Furthermore, methods for reducing the runtime of the IGA will also be investigated.

8. REFERENCES

[1] Pillay, N. 2010. An Overview of School Timetabling Research. In proceedings of the International Conference on the Theory and Practice of Automated Timetabling (Belfast, United Kingdom, August 10-13, 2010). PATAT 2010. To appear.

[2] Abramson, D. 1991 Constructing School Timetables Using Simulated Annealing: Sequential and Parallel Algorithms. Management Science 37(1), 98-113.

[3] Melicio, F., Calderia, J.P. and Rosa, A. 2006. THOR: A Tool for School Timetabling. In Proceedings of the 6th International Conference on the Practice and Teaching of Automated Timetabling (PATAT 2006), E.K. Burke and H. Rudova , Eds., 532-535, ISBN 80-210-3726-1.

[4] Abramson, D. and Abela, J. 1991. 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, 1-11.

[5] Calderia, J.P. and Ross, A.C. 1997. School Timetabling Using Genetic Search. In the proceedings of the International Conference on the Practice and Theory of Automated Timetabling (PATAT '97), 115-122.

[6] Filho, G.R. and Lorena, L.A.N. 2001. A Constructive Evolutionary Approach to School Timetabling. In Proceedings of the EvoWorkshops on Applications of Evolutionary Computing, Lecture Notes in Computer Science 2037, 130-139, Spring-Verlag.

[7] Wilke, P., Grobner, M. and Oster, N. 2002. A Hybrid Genetic Algorithm for School Timetabling. AI 2002: Advances in Artificial Intelligence, Lecture Notes in Computer Science 2557(2002), 455-464, Springer Berlin.

Raghavjee, R and Pillay, N. 2008. An Application of Genetic Algorithms to the School Timetabling Problem. In Cilliers C, Barnard L, Botha RA (Eds.) Proceedings of SAICSIT 2008, 193-199, ACM Press.

[8] Beligiannis, G.N., Moschopoulos. C.N., Kaperonis, G.P. and Likothanassis, S.D. 2008. Applying Evolutionary Computation to the School Timetabling Problem: The Greek Case. Computers and Operations Research 35, 1265-1280, Elsevier.

[9] Raghavjee, R. and Pillay, N. 2009. Evolving Solutions to the School Timetabling Problem. In Proceedings of the World Conference on Nature and Biologically Inspired Computing (Coimbatore, India, December 2009), NaBIC 2009,1524-1527, IEEE.

[10] Smith, K.A., Abramson, D. and Duke, D. 2003. Hopfield Neural Networks for Timetabling: Formulations, Methods,

and Comparative Results. Computers and Industrial Engineering 44, 283-305, Pergamon.

[11] Santos, H.G., Ochi, L.S. and Souza, M.J.F. 2005. A Tabu Search Heuristic with Efficient Diversification Strategies for the Class/Teacher Timetabling Problem. Journal of Experimental Algorithms 10, 2-9, ACM.

[12] Jacobsen, F., Bortfeldt, A. and Gehring, H. 2006. Timetabling at German Secondary Schools: Tabu Search versus Constraint Programming. In Proceedings of the International Conference on the Practice and Theory of Automated Timetabling (PATAT 2006), E.K. Burke, H. Rudova (Eds.), 439-442, ISBN 80-210-3726-1.

[13] Bello, G.S,, Rangel, M.C., and Boeres, M.C.S. 2008 An Approach for the Class /Teacher Timetabling Problem. In the proceedings of the 7th International Conference on the Practice and Theory of Automated Timetabling (PATAT2008), http://w1.cirrelt.ca/~patat2008/ PATAT_7_ PROCEEDINGS/Papers/Boeres-WA2b.pdf. Last accessed 05/02/10.

[14] Birbas, T., Daskalaki, S. and Housos, E. 2009. School Timetabling for Quality Student and Teacher Schedules. Journal of Scheduling 12(2), 177-197, April 2009, Kluwer Academic Publishers.

[15] Santos, H.G., Uchoa, E., Ochi, L.S. and Maculan, N. 2008/ Strong Bounds with Cut and Column Generation for Class-Teacher Timetabling. In the proceedings of the 7th International Conference on the Practice and Theory of Automated Timetabling (PATAT 2008), Montreal, http://w1.cirrelt.ca/~patat2008/ PATAT_7_ PROCEEDINGS/Papers/Santos-WD2b.pdf, last accessed 12 February 2010.

[16] Valouxis ,C. and Housos, E. 2003 Constraint Programming Approach for School Timetabling. Computers and Operations Research 30, 1555-1572, Pergamon.

[17] Moura, A,V. and Scaraficci, R.A. 2010. A GRASP Strategy for a More Constrained School Timetabling Problem. International Journal of Operational Research 7(2),152-170.

[18] Kingston, J.H. 2004. A Tiling Algorithm for High School Timetabling. In Practice and Theory of Automated Timetabling V, Lecture Notes in Computer Science 3616(2005), 208-225, Springer Berlin/Heidelberg.

[19] Alvarez-Valdes, R., Martin, G. and Tamarit, J. M. 1996 Constructing Good Solutions for the Spanish School Timetabling Problem. Journal of the Operational Research Society 47(10), 1203-1215.

[20] Post, G., Ahmadi, S., Daskalaki, S., Kingston, J.H., Kyngas, J., Nurmi, C., Ranson, D. and Ruizenaar, H. 2008. An XML Format for Benchmarks in High School Timetabling. In the proceedings of the 7th International Conference on the Practice and Theory of Automated Timetabling (Montreal, Canada, 2008). PATAT 2008. http://w1.cirrelt.ca/~patat2008/ PATAT_7_ PROCEEDINGS/Papers/Post-WD2a.pdf, last accessed 12 February 2010.

[21] Holland, J.H. 1975 Adaptation in Natural and Artificial Systems. University of Michigan Press.

[22] Koza, J. R. 1992. Genetic Programming I: On the Programming of Computers by Means of Natural Selection, MIT Press.

[23] Pillay, N. and Banzhaf, W. 2010. An Informed Genetic Algorithm for the Examination Timetabling Problem. Applied Soft Computing 10, 455-467.