Evolving Solutions to the School Timetabling Problem

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

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

***Abstract*—There has been a large amount of research into the development of automated systems for creating school timetables. Methodologies such as constraint programming, simulated annealing, and Tabu search have been applied to many school timetabling problems. The research presented in this paper forms part of work-in-progress aimed at evaluating genetic algorithms as a means of solving the school timetabling problem. In previous work a genetic algorithm was successful applied to solving the school timetabling problem with hard constraints. The paper presents our first attempt at extending this system to cater for school timetabling problems with both hard and soft constraints.**

Keywords-school timetabling;genetic algorithms

I. INTRODUCTION

The paper describes an extension of previous work investigating genetic algorithms as a means of solving the school timetabling problem. In previous work the authors have successfully applied genetic algorithms to solving the school timetabling problem with hard constraints. The performance of the genetic algorithm was found to be comparative to other methodologies applied to the same data sets [1]. This paper discusses our first attempt at extending this genetic algorithm system to deal with both hard and soft constraints. The extended genetic algorithm system was evaluated by applying it to the six school timetabling problems made available by Beligiannis et al. [2] that have both hard and soft constraint requirements. The extended system produced timetables which met all the hard constraints and a reduced soft constraint cost for all six problems. The performance of the genetic algorithm is compared to the system implemented in [2].

The following section defines the general school timetabling problem. Section 3 provides an overview of previous studies used to generate school timetables. Section 4 briefly introduces genetic algorithms and section 5 reports on previous studies that have applied evolutionary algorithms to the school timetabling problem. The genetic algorithm system implemented is described in section 6 and the experimental setup used to evaluate the system is outlined in section 7. Section 8 discusses the results obtained by the system on the set of six problems and identifies areas for improvement of the algorithm. A summary of the conclusions of this study and future extensions of this work are described in section 9.

II. THE SCHOOL TIMETABLING PROBLEM

The school timetabling problem involves allocating teachers to classes and teacher-class pairs to periods and venues so as to ensure that the hard constraints of the problem are satisfied and the soft constraints are minimized. The hard constraints and soft constraints are usually different for each school timetabling problem.

A common hard constraint found in all timetabling problems is that the timetable must be clash-free. A clash occurs if a teacher is scheduled to teach two or more different classes in the same period, a class has been allocated to two different teachers in the same period or a venue has been allocated to two different classes or different teachers in the same period. A feasible timetable is one that meets all the hard constraints of the problem.

The school timetabling problem may also have soft constraints associated with it. Soft constraints are requirements that we would like our timetable to satisfy but if they are not met we still have a feasible timetable. For example, classes should have lessons either in the morning or afternoon, no free periods between lessons, and teacher preferences, just to name a few. It is very difficult to satisfy all the soft constraints so the target is usually to minimize the soft constraint cost. The more the soft constraints are met, the better the quality of the timetable.

III. PREVIOUS STUDIES

Numerous studies have been conducted in the field of school timetabling. This section describes some of these studies. Earlier work in the field includes the study conducted by Valouxis et al. [3] which applied constraint programming to the school timetabling problem. The system was used to generate a feasible timetable for a Greek school.

De Haan et al. [4] used a four-phase approach to construct a school timetable for a Dutch secondary school. The first three phases focused on assigning classes and teachers to timetable slots while the fourth phase used a tabu search to improve the quality of the timetable.

More recent studies include the graph colouring algorithm implemented by by Bello et al. [5]. The study applies Tabucol, a tabu search for graph colouring to a graph representation of the school timetable. The system was tested on two artificially created problems and five instances from Brazilian high schools. This approach produced

978-1-4244-5612-3/09/$26.00 ©2009 IEEE

comparative results to both the Grasp tabu search and standard tabu search on the data sets.

Kingston [6] also takes a graph theory approach to the school timetabling problem. The crux of the approach is a bipartite matching model, namely, global tixel matching. The algorithm was successfully applied to six data sets from Australian high schools.

Santos et al. [7] use integer programming to solve the school timetabling problem. A cut and column generation algorithm was used for this purpose. The system was tested on seven problems. Two of these data sets were artificially created and the remaining seven were obtained from Brazilian high schools.

Wilke et al. [8] compare the effectiveness of tabu search, simulated annealing, genetic algorithms and branch and bound algorithms in solving the school timetabling problem. These methods are applied to induce a timetable for a German high school. Simulated annealing produced the best results in terms of both timetable quality and execution time.

IV. GENETIC ALGORITHMS

A genetic algorithm (GA) is an evolutionary algorithm that is used to solve optimization problems. A GA attempts to mimic the human evolutionary process. The algorithm iteratively refines an initial population of potential solutions until a solution is found. An initial population of solutions is created randomly. These solutions are then evaluated using a fitness function. A selection method, such as tournament selection and roulette wheel selection, is then applied in order to choose a parent. Genetic operators are applied to the chosen parents to create offspring. This process of evaluation, selection and recreation is continued until either a solution has been found or a number of iterations/generations have been reached.

V. EVOLUTIONARY ALGORITHMS AND THE SCHOOL TIMETABLING PROBLEM

There has been a fair amount of research into the use of evolutionary algorithms for the generation of school timetables. Caldiera et al. [9] test a genetic algorithm on an artificially created problem with four classes, four teachers and six periods a week. Beligiannis et al. [2] use an evolutionary algorithm to solve seven Greek school timetabling problems. Results obtained were found to be better than other techniques used on the same data sets. Datta *et al.* [10] employed an evolutionary algorithm to solve the class timetabling problem at IIT Kanpur. This system was found to be an improvement over the current manual system. Rahoual et al. [11] present a timetable system that uses a hybrid of both a genetic algorithm and a tabu search. The system was applied to the data set provided by Abramson et al. [12]. The system produced better results than the standard genetic algorithm without the use of tabu search.

VI. PROPOSED GENETIC ALGORITHM

Initial experimentation revealed that it was more effective to use two separate genetic algorithms for hard constraints and soft constraints instead of using a single genetic algorithm to solve the problems for both soft and hard constraints simultaneously. Section A and section B describe the genetic algorithms for hard and soft constraints respectively.

A. Genetic Algorithm for the Hard Constraints

A sequential construction method (SCM) is used to create the initial population for the genetic algorithm. The SCM produces *n* timetables. The best of the *n* timetables is added to the initial population for the genetic algorithm. Thus, if the initial population is of size *m*, the SCM is run *m* times. The SCM algorithm is illustrated in Figure 1.

```
Procedure SCM
Begin
    Loop x = 1 to SCMSize
    Begin
        Create SCMTimeTable x
        Evaluate SCMTimeTable x
    End
    Y = SCMTimeTable with best evaluation
    Add Y to initial population
End
```

Figure 1. SCM algorithm

The algorithm used by the SCM for constructing each timetable is illustrated in Figure 2. Each timetable is represented as a two-dimensional array with each row corresponding to a period in the timetable and each column a class. If the timetable requirements include subclasses or co-teaching these are dealt with first when constructing a timetable. A subclass occurs when a single class is split into two or more groups that are each taught different subjects in the same period. A co-teaching requirement involves temporarily creating new classes made up of students from some established classes. These temporary classes are taught different subjects, usually depending on student choice. The remaining requirements are then placed on the timetable.

```
Procedure Timetable creation
Begin
    Search for split class instance
    Determine if instance is subclass or co-teaching split
    If subclass is found
        Find Class
        Find participating teachers
        If random position is valid add teacher to class timeslot
        Else find another random position
    Else If co-teaching instance is found
        Find Classes involved
        Find Teachers involved
        If random position is valid add teachers to the class timeslot
        Else find another random position
    Else
    Sort remaining teachers according to number of requirements
    Place teachers in order on the timetable
End
```

Figure 2. Timetable creation algorithm

The remaining teachers are sorted in ascending order according to the number of requirements and allocated to a randomly selected timeslot on the timetable. Note that the hard constraints for classes, i.e. there are no clashes, are automatically satisfied due to the structure used to represent the timetable.

Each timetable is evaluated in terms of the hard and soft constraints of the problem. Thus, the evaluation function used is problem dependant. An example of an evaluation algorithm is illustrated in Figure 3.

```
Procedure EvaluateSCMTimeTable
Begin
   Go Through each timeslot
      Count the number of clashes (A)
      Count the number of teacher availability violations
      (B)
      Count empty periods not in last period of the day (C)
 End
Return(A+B+C)
```

Figure 3. Evaluating a timetable created during SCM

The genetic algorithm employs the generational control model and thus consists of a fixed number of generations with the population size the same from one generation to the next. The algorithm terminates when a feasible timetable is found or a preset number of generations is reached. The fitness of each generation is the number of hard constraints violated. A variation of tournament selection depicted in Figure 4 is used to select parents for the next generation. Note that selection is with replacement.

```
Procedure Tournament Selection
Begin
   Select random member D as winner
   Loop T = 2 to t
      Select random member D
      If random number mod 3 = 0
         winner = fitter of D and winner (or either if fitness is
         equal)
      Else If random number mod 3 = 1
         winner remains the same
      Else
         winner = D
End
```

Figure 4. Tournament selection

The mutation operator is used to create the offspring of each successive generation. The mutation operator works by swapping the contents of two timeslots. The genetic parameter *w* determines the number of swaps that occur. At least one of the slots being swapped is involved in a constraint violation in the timetable.

B. Genetic Algorithm for the Soft Constraints

This algorithm also implements the generational control model. The initial population for this algorithm is the population of the last generation of the genetic algorithm for hard constraints. The algorithm uses the fitness evaluation and selection methods described in section A. The mutation operator works by simply selecting two random timeslots and swapping them. The number of swaps is determined by the genetic parameter *w* and then multiplied by 2. This random mutation could result in many hard constraint violations. This problem is solved by applying the original mutation algorithm for hard constraint violations continuously until no hard constraints are violated.

VII. EXPERIMENTAL SETUP

The genetic algorithm discussed in section VI was evaluated using a set of benchmarks made available by Beligiannis et al. [2]. The six data sets used were based on actual data from Greek schools. Some characteristics of these data sets are listed in Table 1 below.

TABLE 1. CHARACTERISTICS OF DATA SETS

Data Set Number	No. of Teachers	No. of Classes
1	34	11
2	35	11
3	19	6
4	19	7
5	18	6
7	35	13

The hard constraints and soft constraints are described in detail in [2]. Ten runs, each using a different seed value for the random number generator, were performed for each data set. The GA system was written in Microsoft Visual C++, Visual Studio 2008 and simulations were run on an Intel Dual-Core 1.86 GHz processor with 2GB of memory and Windows Vista. The genetic parameters for the hard and soft constraint genetic algorithms are listed in Table 2. These values were obtained by performing initial trial runs.

TABLE II. GENETIC PARAMETERS

Parameter	Value
No. of Timetables Generated by SCM	50
Initial Population Size for both Algorithms	1000
Tournament Size for both GAs	30
No. of swaps during mutation for both GAs	30
No. of Generations for both GAs	30

VIII. RESULTS AND DISCUSSION

The GA system was able to produce feasible timetables for all six data sets. The best soft constraint cost and the time taken by the GA system is displayed in Table 3.

TABLE III. SOFT CONSTRAINT COST AND RUNTIME FOR EACH SEED

Data Set	Time (mins.)	Soft Constraint Cost
HS1	18	355
HS2	16	378
HS3	8	161
HS4	8	178
HS5	10	122
HS7	18	413

The study presented in this paper was our first attempt at extending the genetic algorithm system to cater for both hard and soft constraints. Thus, one of the aims of the study was to identify areas for further improvement. The performance of the mutation operator for the soft constraint genetic algorithm was studied in detail for this purpose. Checks were conducted to compare the fitness of the offspring to that of the parent so as to ascertain whether the mutation operator was actually improving the quality of the timetable. It was found that in a number of cases there was no improvement in the fitness of the offspring. It was decided to test a "non-destructive" version of the mutation operator. This operator compares the fitness of a timetable produced on each mutation swap to the fitness of the parent. If the fitness is better further swaps are not performed and the timetable is returned as the offspring. If there is no improvement all swaps are performed and the offspring returned is the timetable on which all w swaps were performed. The mutation operator for the hard constraints was also adapted to reproduce the parent if the offspring generated was not at least as fit as the parent. It was also found that a decrease in tournament size (from 30 to 10) and an increase in the number of generations and mutations swaps (from 50 to 75) was more effective. The result obtained by the revised system is illustrated in Table 4.

TABLE IV. SOFT CONSTRAINT COST COMPARISON

Data Set	Revised System		Beligiannis et al. [2]	
	Time (mins.)	Soft Constraint Cost	Time (mins.)	Soft Constraint Cost
HS1	58	107	30	121
HS2	55	121	30	148
HS3	22	38	24	50
HS4	21	77	24	90
HS5	17	31	22	32
HS7	60	154	45	215

The "non-destructive" mutation operators have improved the quality of timetables induced drastically. For completeness Table 4 also compares the soft constraint cost to that obtained by the system implemented by Beligiannis et al. [2]. Note this is only included to indicate the contribution made by this study and a direct comparison is difficult as the algorithms were run on different machines with different parameter values. For all six data sets the revised GA system has produced better quality timetables.

IX. CONCLUSION AND FUTURE WORK

The study presented in this paper is our first attempt at extending a genetic algorithm for school timetabling to cater for both hard constraints and soft constraints. The extended system was successfully applied to six real-world data sets. The study revealed that it was necessary to run two separate genetic algorithms to deal with hard constraints and soft constraints instead of simultaneously attempting to minimize both these costs. The use of "non-destructive" mutation operators improved the performance of the GA. Future work will test the revised system on additional data sets including data obtained from a local primary and high school.

REFERENCES

[1] R. Raghavjee, and N. Pillay, "An application of Genetic Algorithms to the School Timetabling Problem", Proccedings of SAICSIT '08, ACM Press, October 2008, pp. 193-199.

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

[3] C. Valouxis, and E. Housos, "Constraint Programming Approach for School Timetabling",Computers and Operations Research, vol. 30, 2003, pp. 1555-1572.

[4] P. De Haan, R. Landma, G. Post, and H. Ruizenaar, "A Four Phase Approach to a Timetabling Problem in Secondary Schools", Proceedings of PATAT '06 (Practice and Theory of Automated Timetabling), 2006, pp. 423-425.

[5] G.S. Bello, M.C. Rangel, and M.C. S. Boeres, "An Approach for the Class/Teacher Timetabling Problem using Graph Colouring", in Proceedings of of PATAT '08, 2008.

[6] J. H. Kingston, "Resource Assignment in High School Timetabling", Proceedings of PATAT '08, 2008.

[7] H.G. Santos, E. Uchoa, L.S. Ochi, and N. Maculan, "Strong Bounds with Cut and Column Generation for Class-Teacher Timetabling", Proceedings of PATAT '08, 2008.

[8] P. Wilke, and J. Ostler, "Solving the School Timetabling Problem Using Tabu Search, Simulated Annealing, Genetic and Branch & Bound Algorithms", Proceedings of PATAT 2008.

[9] J. P. Caldeira., and A. C. Rosa, "School Timetabling Using Genetic Search", Proceedings of PATAT '97(Practice and Theory of Automated Timetabling), 1997.

[10] D. Datta and K. Deb, "Solving Class Timetabling Problem of IIT Kanpur using Multi-Objective Evolutionary Algorithm", KanGAL Report, 2006.

[11] M. Rahoual and R. Saad R., "Solving Timetabling Problems by Hybridizing Genetic Algorithms and Tabu Search", Proceedings of PATAT '06, 2006, pp. 467-472.

[12] J. E. Beasley, OR-Library, February 2008, http://people.brunel.ac.uk/~mastjjb/jeb/orlib/tableinfo.html