School of Computer Science and Information Technology
University of Nottingham
Jubilee Campus
NOTTINGHAM NG8 1BB, UK

Computer Science Technical Report No. NOTTCS-TR-1995-8

A Hybrid Genetic Algorithm for Highly Constrained Timetabling Problems

Rupert Weare, Edmund Burke and Dave Elliman

First released: January 1995

A Hybrid Genetic Algorithm for Highly Constrained Timetabling Problems

Technical Report NOTTCS-TR-95-8, January 1995

Rupert WEARE, Edmund BURKE and Dave ELLIMAN
R.Weare@cs.nott.ac.uk
Department of Computer Science
University of Nottingham
Nottingham, UK

Abstract

In this paper, we present a powerful hybrid genetic algorithm based around a heuristic timetabling framework. This combines a direct representation of the timetable with heuristic crossover operators to ensure that the most fundamental constraints are never violated. We explain how the population is seeded so as to produce a solution which cannot be outperformed by the heuristic method alone. We also see how the algorithm is guaranteed to always produce a feasible solution by hard coding constraints which must not be broken. Hybrid crossover operators are used, which propogate the most desirable features of the timetable to produce good quality solutions, even for large and highly constrained problems. Finally, we present the results from applying the algorithm to a particularly difficult problem, and demonstrate the variety of possible timetables that can be generated, depending upon the requirements of the user.

Introduction

Recently, there has been a lot of attention paid to the problem of automating the construction of university course and examination timetables. Many different methods have and are being tried, including Simulated Annealing [10], Constraint Logic Programming [1,15], Linear Programming [19] and Graph Colouring Heuristics[4]. A growing number of researchers are now turning to Evolutionary Algorithms, or evolutionary hybrids, as a powerful method of solving difficult timetabling problems[3,6,8,16,18]. Various alternatives on the evolutionary theme have been applied successfully to real situations and some are now used regularly to produce course and examination schedules [8,18].

At its simplest, timetabling is the problem of scheduling a set of events (for example: exams, lectures, tutorials) to specific time slots such that no person or resource is expected to be in more than one location at the same time and that there is enough space available in each location for the number of people expected to be there. These two main fundamental constraints, and many others, combine to make timetabling a classically hard problem to solve.

The algorithm we present here uses a heuristic timetabling framework presented by the authors in [4].

Constructing an Examination Timetable

Several major problems confront anyone wishing to produce an examination timetable. The first is the actual size of the problem. The examination timetable for the University of Nottingham consists of over eight hundred exams scheduled over a period of two weeks, repeated twice a year. Each exam, on average, conflicts with about thirty others while the most conflicting exam, an Inter-Faculty Language Course, conflicts with over two hundred other exams. By conflicts, we mean the number of other exams which cannot be scheduled in the same period because they have at least one student in common. The biggest single exam, a first year maths course, has over five hundred people taking it.

Clearly a solution exists and can be found because the problem is regularly solved by universities, though not without many problems and a huge amount of administrative effort. Current approaches in British universities typically involve an heuristic assignment algorithm with a large manual input, if automation is used at all. (The authors are currently compiling a survey of timetabling problems and solution methods in British Universities.) Using one of these heuristics in a hybrid Genetic Algorithm will aid the search for a good timetable and ensure that we do not produce worse solutions than are already found.

A second problem that appears in all complex timetabling problems is the variety and large number of, sometimes non-compatible, constraints. Heuristic assignment algorithms do not easily allow for the possibility of search and strongly limit the type of constraints that may be catered for. By having a separate evaluation function, the timetabler can encode and alter any (or all) of these to produce the timetable required rather than one restricted by the assignment algorithm. Clearly there are still problems with this approach, particularly that some constraints are viewed as hard, and therefore cannot be violated, and others, viewed as soft, which may be violated if absolutely necessary. It is very possible that, if encoding all constraints in the fitness function, the hard constraints may be swamped by the large number of soft constraints just because of the way that they are represented. This may be overcome this by redefining the search space in such a way as to not allow infeasible timetables i.e. those with exam clashes or with more exams scheduled than there is space to house them.

Consequently, we must also redefine the crossover and mutation operators so they do not produce infeasible timetables and the initial population must be specially generated, in this case using a graph colouring algorithm.

The problem of finding a colouring of a graph with a minimal number of colours is NP-complete[13]. This problem is equivalent to the problem of finding a timetable in a minimum number of periods. It is also true that the problem of finding a k-colouring of the same graph (a timetable in k periods) is combinatorially hard so it is an NP-complete problem to decide how long the exam period should be in order to be able to schedule all the exams. Of course, this is usually decided by experience or by other time constraints such as the amount of time required for marking. Within U.K. universities, there has been a drive towards a modular structure and a large increase in student numbers. This has led to considerably more complicated timetabling problems; the University of Nottingham recently had to change from two to three exam periods a day to allow for this. The method to be described allows the length of the timetable to vary. The random sequential graph colouring algorithm used to generate the starting population uses on average about twice as many periods as the optimal amount [11] giving an estimation as to the hardness of the problem. The genetic algorithm then evolves new timetables, possibly reducing the length, if required. This approach guarantees a feasible timetable while not potentially creating a search space in which no solution exists. It may be the case that no solution does exist in the specified number of periods but at least when this situation occurs, it will become obvious and the timetabler may seek to make more time available. Where the opposite is true (i.e. in the rare event that there is more than enough time) a constraint causing exams to be scheduled early for marking purposes may be traded off against a constraint to *spread out* students' exams, assigning them evenly across the whole examination timetable.

Also to be dealt with is the problem of Epistasis, where the recombination of two potential solutions leads to a new solution worse than either of the original ones. This ties up with the problem of propagating good features of the timetable onto future generations. Both are dealt with in later sections.

A Heuristic Timetabling Framework

The timetabling methodology described in figure 1 combines two heuristic algorithms, Algorithm 1 finds a non-conflicting set of exams and Algorithm 2 assigns these selected exams to rooms. The process is repeated for each period until all exams have been scheduled to a period and a room without conflict or overcrowding. The unselected exams may be taken as a basis for the search of exams for the next period since, they will have no conflicts between them, but also will not conflict with any exams from the previous period. This will reduce the number of times a student has to sit two exams in a row.

The framework may also be used as the basis for a hybrid crossover operator which maintains a population of feasible timetables (see figure 2). Hybrid genetic operators are introduced because normal domain independent operators will hardly ever produce a feasible timetable. Instead of considering all the unscheduled exams in the search for those to be placed in the next period, this can be restricted to the exams in the parent timetables that are currently scheduled in that period. Once those exams scheduled in that period by both parents have been selected, Algorithm 1 (whatever that might be) can complete the set of non-conflicting exams using only those currently available. Any unscheduled exam may be passed on to the next period for future consideration.

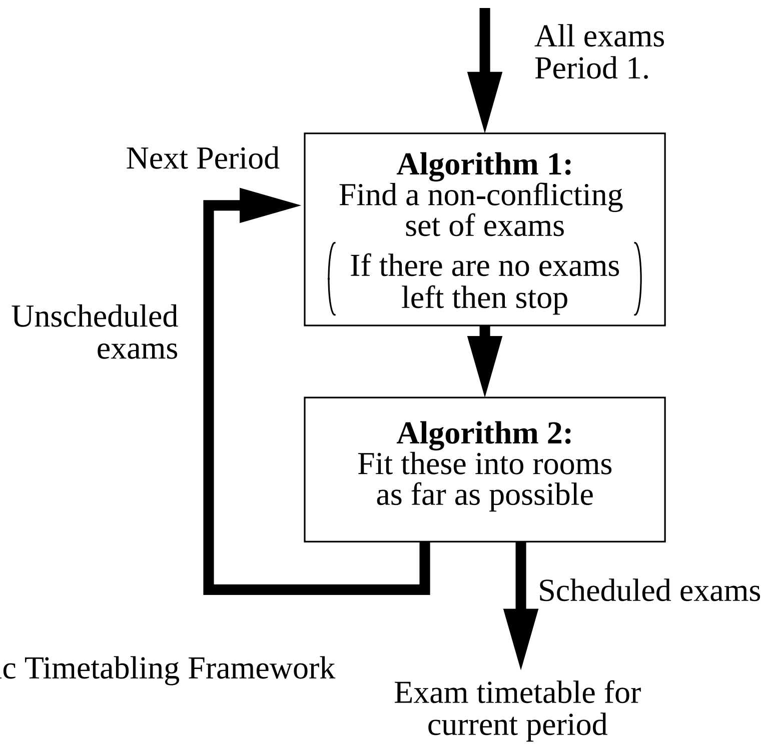
All exams
Period 1.
Next Period
Algorithm 1:
Find a non-conflicting set of exams
(If there are no exams left then stop)
Unscheduled exams
Algorithm 2:
Fit these into rooms as far as possible
Scheduled exams
Exam timetable for current period

Figure 1. A Heuristic Timetabling Framework

Whatever algorithms are used, it is clear that the hybrid crossover operator will satisfy the properties of *respect* and *assortment* given by Radcliffe [21]. Respect is the property that if an exam is timetabled to the same period in both parents then it will be scheduled to that period in the child. Assortment is the property that the operator can generate a child such that if *Exam 1* is scheduled to *Period 1* in the first parent and *Exam 2* is scheduled to *Period 2* in the second parent then the child may have *Exam 1* in *Period* 1 and *Exam 2* in *Period 2* providing these are compatible. The operator is not, however, *strictly transmitting*. This means that in the child timetable, it could be the case that an exam is not scheduled to the same period as in either of the parents. The fact that this property may not be satisfied is a consequence of the choice of search space and thus requires a rule to deal with any unplaced exams, such as passing them on to the next period.

A Hybrid Evolutionary Algorithm For Timetabling

The evolutionary algorithm described searches only in the domain of feasible timetables. The requirement for specific rooms to be scheduled has been temporarily waved for the purposes of experimentation and is replaced by the rule that no more than a set number of students may be scheduled to any particular period. This is the rule that is used in the current University of Nottingham Examination Scheduling System.

The Creation of an Initial Population

The following random algorithm is used to generate conflict-free graphs which form the initial population.

```
For each population member:
    Generate a random ordering of exams
    Taking each exam in turn according to that ordering:
        Find the first period in which the exam may be placed without conflict
        and so that the number of students does not go above a set maximum.
        Place the exam in that period.
```

This algorithm can quickly produce large populations of random feasible exam timetables. Clearly, it will not produce timetables of minimum length, nor will it produce extremely long drawn out timetables containing unused or little used periods. The assumption is that the problem to be solved is reasonably constrained, i.e. that the amount of time and space available is limited.

Crossover Operators

Figure 2 shows how the crossover operator works for period *i*. The operator starts by looking at the first period. It takes exams scheduled in that period (in both parents) and then uses an algorithm to select other exams so that none clash with those already scheduled and the limit on the number of spaces is not violated. Once this is completed, the crossover looks at period two and so on until all exams are placed.

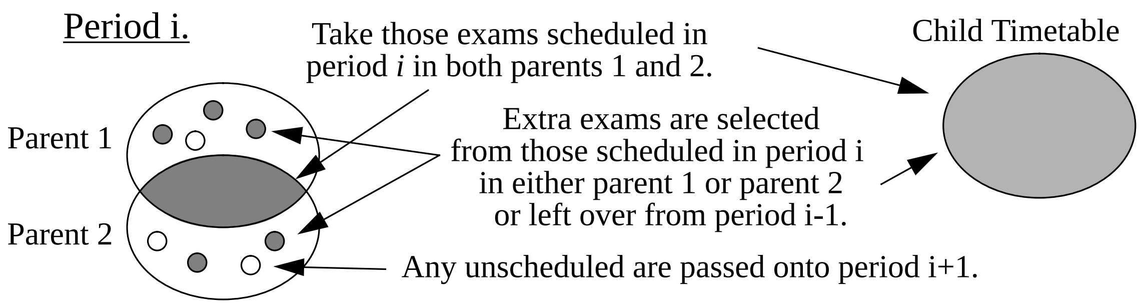
Period i.
Take those exams scheduled in period *i* in both parents 1 and 2.
Child Timetable
Parent 1
Extra exams are selected from those scheduled in period i in either parent 1 or parent 2 or left over from period i-1.
Parent 2
Any unscheduled are passed onto period i+1.

Figure 2. A Heuristic Hybrid Crossover Operator

The normal domain independent uniform crossover operator, when given a choice, takes alleles randomly from one parent or the other. It is not possible, as has been said in the previous section, to have strictly transmitting operators within our chosen domain. However, the action of the standard operator may be partially mimicked by choosing an appropriate exam selection algorithm (which we called Algorithm 1). However, as well as ensuring that each offspring is feasible, hybrid operators may also use non-payoff information to guide the genetic algorithm towards good solutions; this being information other than the single value returned by the evaluation function.

We may, therefore, construct a number of different operators based on the same framework but using alternative selection algorithms. An experiment doing exactly this is discussed below. The operators we use are described in figure 3. In each case, once an exam is selected, all other exams that clash with it are labelled as unscheduled for that period. The algorithms continue to select exams until there are none left to chose from.

•*Random (rand)*

Exams are selected at random. This is the closest to the standard uniform crossover.

•*Largest Degree (large)*

Exams are selected according to the number of other exams they conflict with. This is a widely known graph colouring heuristic[22].

•*Most Similar Exams (similar)*

Exams are selected according to how many conflicting exams they have in common with those already scheduled. This is also a graph colouring heuristic[4].

•*Latest Scheduled in other Parent (late)*

Exams are selected according to where the same exam is scheduled in the other parent. Since unplaced exams are passed on to the next period, this increases the chances of shortening the length of the timetable.

•*Least Conflicting with Previous Period (spread)*

Exams are selected so as to minimise the number of conflicts with exams in the previously scheduled period.

•*Both late and spread together (E)*

The two operators are used together, the ratio of the number used of each also evolves according to which is producing the better solutions at the time.

Figure 3. Hybrid Genetic Operators for Evolutionary Timetabling.

The Mutation Operator

Mutation, like crossover, must also ensure that a timetable remains feasible after its action. It cannot therefore take any exam and shift it to another period at random, since this may cause a conflict between the moved exam and ones already scheduled. Instead, we chose to incorporate mutation into the crossover algorithm. This is done by adding exams to the current search that would otherwise not be considered until a later period

Fitness Calculation and Selection

The evaluation function can clearly be made up of any timetabling related factors. For instance, if it were hoped that larger exams appeared earlier on for marking purposes then it would be possible to include that in the function. In the experiment discussed below, we concentrate on two particular common requirements:

•*The length of the timetable.*

•*The number of conflicts between exams in adjacent periods.*

Although many different alternatives are available for epitomising a number of semi-dependant criteria such as may be found in a timetabling problem, Corne [8], argues that a simple linear penalty-weighted sum is sufficient for the timetabling problem. Whether this holds for the slightly changed search neighbourhood described here remains the subject of future work. It is certainly sufficient to illustrate the workings of the algorithm. If we wish to produce a system to be used, the workings of the penalty function needs to be sufficiently clear for the user to be able to alter it usefully.

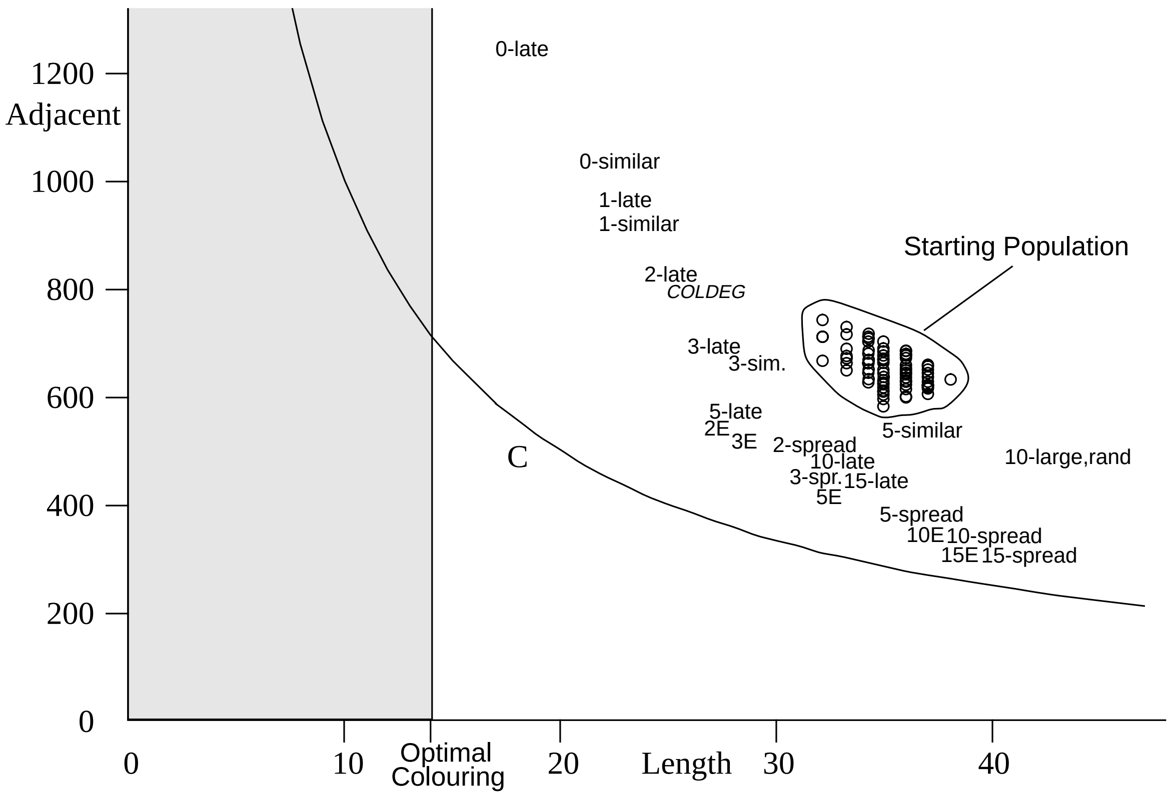

Figure 4. Timetable Length in periods vs Number of times conflicting exams are scheduled in adjacent periods

Timetables Produced by The Algorithm

The above operators were incorporated into a generational Genetic Algorithm with population size 200, using exponential rank based selection with selection pressure 0.986. A random set of constraints was generated with known minimal colouring [17] as a test problem. This has two hundred exams, perhaps typical of a faculty scheduling problem, but is considerably more constrained The probability of any two exams conflicting is $^1/_2$ (which is equivalent to 10,000 constraints). A minimal schedule for this set (i.e. a timetable which uses the least number periods) would be fourteen periods long. The penalty function used was 100 times the length of the timetable plus a variable *x* times the number of conflicting exams scheduled in adjacent periods. In figure 4, solutions are marked as the value of *x* followed by the particular operator that produced the solution. *COLDEG* marks the solution produced by the heuristic graph colouring algorithm [5] which chooses vertices according to which has the smallest number of possible values; this is generally quoted as being one of the most effective of known heuristic algorithms [16]. All solutions took in the order of fifteen minutes to produce on a Sun Sparc Classic.

On this highly constrained problem, neither the *random* operator nor the *largest degree* operator perform very well. In fact, only when *x* is 10 do they even improve on the starting population but then they are both dominated by timetables produced by both the *spread* and *late* crossovers. This casts doubt on the ability of the naive operators to cope with the pontentially highly epistatic problems that are often found in the field of timetabling. The *similar* crossover outperformed the others when *x* was set to one but otherwise also performs poorly. This is perhaps surprising as this heuristic produces good colourings when used by itself.

The two operators that consistently out-performed the others were *spread* and *late*. *Spread* was particularly good at enlarging the timetable whilst minimising the number of adjacent conflicts. The main strength of the *late* operator was in shortening the length of the timetable; These are two criteria by which timetables are often judged. In the case where the only criteria is the length of the timetable, *late* finds a timetable using only three more periods than is optimal.

Late and *spread* were then used together, the proportion of each being evolved according to which is producing the best timetables. This new combined version outperformed all the individual operators for all values of *x* tried.

The curve C is given by 10,000 (number of constraints)/Length. It is neither an upper nor lower bound for the quality of the timetables but shows the trade-off we might expect between length and the number of adjacent conflicting exams. Of course, we cannot produce any timetables in the shaded region to the left of the vertical line given by “length = 14”. However, given that we can actually find a timetable, this function gives the expected number of adjacent conflicts if the exams are evenly spread between all periods.

Conclusion

Evolutionary algorithms show great promise in the area of educational timetabling, particularly in their ability to consider, and optimize, the wide variety of different constraints that may be found in universities. We have shown here that by considering hybrid crossover operators, incorporating already known graph colouring techniques, we can produce good quality timetables, even from extremely constrained problems.

By representing the timetable directly, and not as a set of instructions as to how to produce it, we ensure that the chromosomes that are selected really are the best ones. By hard-coding the fundamental, unbreakable, constraints into the algorithm, we increase the probability that an appropriate solution can be found (if one exists) and we make certain that no number of extra *soft* constraints might cause an infeasible timetable to be found. The genetic algorithm is given a *head start* over other approaches by seeding the initial population with already feasible timetables.. By using (a combination of) specially developed heuristic operators, the search can be guided towards the best sector of the solution space and hence the best timetables.

We are now continuing to develop the hybrid genetic algorithm, and the different operators, to include other hard and soft constraints. Currently, the operators continue picking exams for the current period until there are no more suitable exams left. This means that currently, it is highly unlikely that a solution of length 33 and no adjacent conflicting exams (which may be constructed by taking the periods from the solution *0-late* and putting an extra empty period in between each) will be found any other way. Using the hybrid method means that it is very easy to include an extra hard constraint to ensure that none of the exams picked for a particular period will clash with those already scheduled in the previous period. This would produce timetables with a more *even* number of exams in each period while definitely not having any adjacent conflicting exams at all.

We are also looking at ways of how the search time may be decreased with minimal loss of final timetable quality by using features specific to the structure of timetable graphs.

References

[1] Boizumault P., Gueret C. and Jussien N. (1994) “Efficient Labelling and Constraint Relaxation for Solving Time Tabling Problems” *International Logic Programming Symposium - Workshop on Constraint Languages and their use in Problem Modelling.* pp116-130

[2] Bruns R. (1993) “Knowledge-Augmented Genetic Algorithm for Production Scheduling”, *IJCAI ‘93 Workshop on Knowledge based Production Planning, Scheduling and Control.*

[3] Burke E.K., Elliman D.G. and Weare R.F. (1994) “A Genetic Algorithm Based University Timetabling System” East-West Conference on Computer Technologies in Education, Crimea, Ukraine pp35-40.

[4] Burke E.K., Elliman D.G. and Weare R.F. (1993) “A University Timetabling System Based on Graph Colouring and Constraint Manipulation”, *To appear in the Journal of Research on Computing in Education.* Vol. 26. issue 4.

[5] Brelaz D.(1979) “New Methods to Color the Vertices of a Graph” *Comm. A.C.M.* 7, pp494-498.”

[6] Colorni, A., Dorigo, M., Maniezzo, V. (1990) “Genetic Algorithms and Highly Constrained Problems; The Timetable Case”, *Parallel Problem Solving from Nature I*, Springer-Verlag, pp.55-59.

[7] Carter M.W. (1986) “A Survey of Practical Applications of Examination Timetabling Algorithms” *OR Practice* 34, pp 193-202.

[8] Corne, D., Ross,P.,Fang H-L (1994) “Fast Practical Evolutionary timetabling”, *Lecture Notes in Computer Science 865* , Springer-Verlag, pp250-263.

[9] Davis L. (1991) “Handbook of Genetic Algorithms” Van Nostrand Reinhold

[10] Davis L. and Ritter F. (1987) “Schedule Optimization with Probabilistic Search” *Proceedings of the 3rd IEEE Conference on Artificial Intelligence Applications*, February 1987, Orlando, Florida, USA, pp231-236.

[11] Grimmet G.R. and Mcdiarmid C.J.H. (1975) “On Colouring Random Graphs“, *Math. Proc. Camb. Phil. Soc.* 77, pp313-324.

[12] Hancock P.J.B. (1994) “An empirical Comparison of Selection Methods in Evolutionary Algorithms”, *Lecture Notes in Computer Science 865,* Springer-Verlag, pp80-94

[13] Johnson D.S. (1974) “Worst Case Behaviour of Graph Coloring Algorithms” *In Proc. 5th SE Conf. on Combinatorics, Graph Theory and Computing.* Utilitas Mathematica Publishing, Winnipeg, Canada. pp513-528.

[14] Johnson D.S., Aragon C.R., McGeoch L.A. and Schevon C. (1991) “Optimization by Simulated Annealing: An Experimental Evaluation; Part II, Graph Coloring and Number Partitioning” *Operations Research* 39, pp378-406.

[15] Kang L. and White G.M. (1992) “A Logic Approach to the Resolution of Constraints in Timetabling” *European Journal of Operational Research* 61 pp306-317.

[16] Ling, S.E. (1992) “Integrating Genetic Algorithms with a Prolog Assignment Problem as a Hybrid Solution for a Polytechnic Timetable Problem”, *Parallel Problem Solving from Nature II*, Elsevier Science Publisher. pp321-329.

[17] Manvel B. (1981) “Coloring Large Graphs”, *Congressus Numerantium,* Vol 33, pp197-204

[18] Paechter B., Cumming A., Luchian H. and Petriuc M. (1994) “Two Solutions to the General Timetable Problem Using Evolutionary Methods” *To appear in IEEE Conference of Evolutionary Computing*

[19] Tripathy A. (1984) “School Timetabling - A case in Large Binary Integer Linear Programming” *Discrete Applied Mathematics* 35.3 pp313-323.

[20] Radcliffe N.J. and Surry P.D. (1994) “Formal Memetic Algorithms”, *Lecture Notes in Computer Science 865*, Springer-Verlag, pp1-16.

[21] Radcliffe N.J. (1993)“Genetic Set Recombination”, *Foundations of Genetic Algorithms 2*, Ed: L.Darrell Whitley, Morgan K. pp203-219.

[22] Welsh D.J.A. and Powell M.B. (1967) “An Upper bound for the Chromatic Number of a Graph and Its Application to Timetabling Problems” *Comp. Jrnl.* 10, pp85-86.