Successful Lecture Timetabling with Evolutionary Algorithms

Peter Ross, Dave Corne, Hsiao-Lan Fang
Department of Artificial Intelligence
University of Edinburgh
80 South Bridge, Edinburgh EH1 1FN
email: {peter|dave|hsiaolan}@aisb.ed.ac.uk

Abstract

Arranging a lecture/tutorial/lab timetable in a large university department or school is a hard problem faced continually in educational establishments. We describe how this problem has been solved in one institution via the use of evolutionary algorithms. The technique extends easily and straightforwardly to any lecture timetabling problem. Although there may be more effective ways to handle particular instances of the general lecture timetabling problem, we note that the combination of speedy, good results and ease of development for the particular application in hand make the EA-based technique we present potentially widely useful in general.

1 Introduction

Lecture timetabling is the problem of assigning times and places to a many separate lectures, tutorials, etc ..., to satisfy several constraints concerning capacities and locations of available rooms, free-time needs and other such considerations for lecturers, and relationships between particular courses. The most prominent overall constraint (central to all timetabling problems) is that there should be no *clashes*; that is: any pair of lectures (or tutorials, etc ...) which are expected to share common students or teachers should not be scheduled simultaneously.

Typically, this is addressed by drawing up an initial draft timetable, followed by perhaps weeks of redrafting as complaints about the most recent draft flow in from various sources. The space of possible timetables can be nightmarish to traverse, and there have been several attempts to find useful AI/OR approaches to aid the process [2, 15, 4, 11, 3]. Success has been recently reported for using evolutionary algorithms (EAs) for timetabling [1, 6, 14, 7, 8, 17, 16]. Here we describe one such EA approach, and present illustrative results on some real lecture timetabling problems.

In presenting results on some real problems, we augment similar work which also reports results on real problems [6, 7, 8, 17, 16], but we also present a fuller and deeper discussion of the general approach, clarifying how it may be used on a much wider range of problems than that studied. Some comparison is also made between the use of different EA selection schemes. Also, in reporting EA-based results in comparison with the independently and 'expertly' calculated timetables for four real lecture timetabling problems we show clearly how this approach can yield very beneficial improvements.

Overview

We first describe the kind of problem addressed in more detail in section 2. Description of our EA based approach follows in section 3, and notes on implementing the approach then appear in section 4. Illustrative experiments on real problems appear in section 5, followed by general discussion in section 6.

2 Lecture Timetabling Problems

The basic element of a lecture timetabling problem is a set of events $E = \{e_1, e_2, \ldots, e_v\}$. Each member of E is a unique event requiring assignment of a time and a place. That is, it may be a lecture, a tutorial, a lab session, or some other event which plays a part in the term timetable. We could alternatively formulate this, for example, in terms of a set of subjects S, each of which has associated numbers of lectures, tutorials, lab sessions, etc ... per week. However it is simpler to take as our starting point the set E as described, which is easily generated from the latter kind of data. Eg, two separate Lisp Programming lectures and five Lisp Programming tutorials will constitute seven members of E.

Each event e_i has an associated length l_i (how long the event is in, say, minutes), and an associated size s_i, which is either known or an estimate of the number of students expected to attend that event. There is also a set of 'agents' $A = \{a_1, a_2, \ldots, a_t\}$; these are lecturers, tutors, technicians, etc ... — people with some kind of distinguished role to play in an event. Finally, there is a set of places $P = \{p_1, p_2, \ldots, p_q\}$, and a set of times $T = \{t_1, t_2, \ldots, t_s\}$. An *assignment* is a four-tuple (a, b, c, d), in which $a \in E, b \in T, c \in P, d \in A$, with the interpretation "event a starts at time b in place c and is taught (lectured, tutored, ...) by agent d. A lecture timetable is simply a collection of n assignments, one for each event.

Such problems are beset by many kinds of constraint. A fuller presentation of these appears in [18], and various more simplified treatments have been presented elsewhere [2, 1, 3]. The following briefly notes the necessary aspects of the approach discussed in [18] which are needed for understanding the rest of this paper. Following this we describe the constraints which need to be met in a *specific* series of lecture timetabling problems; this serves as a background case-study against which we can illutrate some general aspects of the implementation of the approach to an arbitrary timetabling problem.

3 Evolutionary Timetabling

Assuming familiarity with the basic processes in EAs, it suffices to describe our approach by reference only to the chromosome representation and the fitness function. Those unfamiliar with the basics of EAs can consult good texts such as [12, 10].

The Chromosome Representation

A 'timetable' chromosome is a vector of symbols of total length $3v$ (recall: v is the number of events), divided into contiguous three-gene chunks. The three alleles in the ith chunk, where $1 <= i <= v$, represent the time, place, and agent assignments of ith event. Naturally, the sets of possible alleles at time, place, and agent genes are respectively identified with the sets T, P, and A. The simple example chromosome "abcdef" represents a timetable in which event e_1 starts at time a in place b, involving agent c, and event e_2 starts at time d in place e, involving agent f.

This constitutes a 'direct' representation, as opposed to the more indirect style common in EA-based job shop scheduling work, and recently implemented for timetabling in =citePaechter+94. Relative advantages and disadvantages of these two styles are beyond the scope of this paper, but are a central point of interest. Suffice to say here that comparison of the two is beset by complications, but it so far seems that the direct approach *enhanced* with the introduction of intelligent mutation operators (which take great advantage of the directness of the representation, and hence cannot be feasibly constructed for use with the indirect style) [8], vies on equal terms with an ingenious version of the indirect style [16]; these observations are yet to be properly backed up empirically.

3.1 The Fitness Function

A maximally fit timetable is clearly one which satisfies all of the imposed constraints. Also, it seems reasonable to distinguish between timetables in terms of fitness based on the numbers and kinds of different constraints violated. A choice of fitness function which meets this behaviour is as follows, where C is the set of constraints in the problem, P_i is a penalty associated with constraint i, and $v_i(g) = 1$ if timetable g violates constraint i, and 0 otherwise:

$$f(g) = 1/(1 + \sum_{i \in C} P_i v_i(g)) \tag{1}$$

The relative penalty values may be chosen to reflect intuitive judgement of the relative importance of satisfying different kinds of constraint. Further discussion of this kind of objective function and other possibilities is beyond the scope of this paper, but appears elsewhere, eg: [8, 19, 13]. In general, however, a penalty function as above, using a rough choice of penalty settings derived from the course organisers' notion of relative importance of different constraints, appears adequately robust for many problems.

4 A Specific Lecture Timetabling Problem

An MSc course in the Department of Artificial Intelligence, University of Edinburgh (EDAI) involves eight taught course modules spread over two terms. The course is organised into 'themes', each involving a particular combination of 8 modules, of which some are compulsory and some optional. As well as choices from among the 30+ modules available in the AI Department, students may also choose modules from the Computer Science (CS) Department and others. A complicating factor here is that the CS Dept is an inconvenient bus ride away from the AI Dept.

T comprises 80 start times, 16 per day on each day of a five day week. Each day's slots are at half-hourly intervals from 9am to 4:30pm. E comprises a large collection of lectures, tutorials, and lab sessions, mostly an hour long, but sometimes two hours long. A student enrolled on a course must attend all the lectures in E involving the course, but only one of the tutorials or labs (E will include several tutorials or labs for each course). Separate courses are pre-assigned to either term 1 or term 2. Hence, in one academic year there is a separate lecture timetabling problem for each term involving roughly half of the modules available on the course as a whole. The full set of constraints which need to be faced are as follow:

Options : Student's options should be kept open as far as possible. No pair of lectures in the same theme should overlap in the timetable. More generally, lectures x and y should not overlap if there is expectation that one or more students may wish to take both courses x and y.

Event Spread : The individual timetable for any student must be spread out fairly evenly. Eg: A student should not have to sit through four lectures in a single day. Rather, the events an individual student must attend should be evenly spaced out during the week. Also, different lectures on the same topic (eg: there may be 2 Prolog lectures per week) should occur on different days.

Travel Time : A student should have at least 30 minutes free for travel between events in the CS Dept and events in the AI Dept.

Slot Exclusions : CS lectures should occur in morning slots, and AI lectures in afternoon slots (this arises from an inter-departmental agreement). Also, lectures at lunchtime (starting at 1:00pm or 1:30pm) should be avoided if possible. In a similar vein, various constraints of the form "event e cannot start at time t" arise owing to other commitments of the staff involved.

Slot Specifications : Various constraints are given in the form "event e must start at time t", arising for various reasons.

Capacity : The size of a lecture or tutorial should not exceed the capacity of the room it occurs in. Also, a room can only cope with one event (lecture, tutorial, or lab) at a time (this is the main difference between lecture and examination timetabling).

Room Exclusions : Many constraints on room assignments for particular events can easily be derived from the Capacity constraints, along with information about the expected sizes of events. In addition however, there are other considerations which lead to several *a priori* constraints of the form "event e cannot occur in room r". For example, event e may demand disabled access, or certain audio-visual requirements unavailable in room r.

Room Specifications : Similarly, several constraints are apparent of the form "event e must occur in room r".

Juxtaposition : Preferably, all tutorials or laboratory sessions for any course should occur later in the week than the week's first lecture on that course. Sometimes this is particularly necessary, since a tutorial or lab session may be based on the lectures which were held (hopefully) earlier in the week. In other cases this is desirable but not vital.

Translating the Constraints into a Fitness Function

In this case, it so happens that every lecture's agent (ie: lecturer) is predetermined, and individual lecturers have already decided in advance (providing details in the style of exclusion constraints) which slots they are not available for. Tutors for tutorials and lab sessions are not pre-determined in this way, but for these there is no point in incorporating them into the timetable at this stage (usually well in advance of term), since we simply do not know for sure who will be available and when. For this problem we therefore need not consider the set A, and hence can use chromosomes of length $2v$. The above constraints can be dealt with as follows:

Keeping Options Open

The Options constraint is handled by interpreting it as a large collection of binary constraints, each involving a distinct pair of events expected to share students. For convenience, we derive a set of 'virtual' students, each of whom takes a distinct one of the set of possible four or five-module options for the term. This set of virtual students is then used to derive the collection of distinct binary constraints between events. Here, such a binary constraint occurs between every pair of distinct lectures taken by some virtual student. A similar constraint also holds between distinct lectures on the same module, and between lectures and tutorials on the same module. No such constraint is needed between different tutorials or labs for the same course, or even between tutorials and labs on different courses. Such may be scheduled simultaneously, and often are; in due course this gives rise to constraints which affect each students' choice from the set of tutorial and/or lab sessions available for each module.

This amounts to a collection of binary constraints of the form "e_1 must not overlap in time with e_2"; the fitness function must check for violation of each of these in turn. Similarly, it should be clear how the basic problem of avoiding clashes in any other lecture or exam timetabling problem can be dealt with. Notice too that we can incorporate the Travel Time constraint here. If e_1 and e_2 are timetabled with a break of, say, k minutes between them, but are assigned to rooms which take more than k minutes to travel between, then any Options constraint between them is effectively violated. Hence, the violation check for Options constraints can, simply via accessing the 'place' genes for e_1 and e_2 and a given travel-time matrix for the places, also account for Travel Time constraints.

Event Spread constraints

Event spread constraints can be handled in a number of ways. Eg, to even out the event spread for individual students we might explicitly calculate some measure or measures of event spread for each virtual student. Alternatively, we could reasonably approximate this by examining some measure or measures of the spread of the timetable as a whole. Both would seem to offer the same overall effect; the latter will typically be computationally cheaper, but the former approach would seem to offer more potential for control and tradeoff of different aspects of the event spread for individual students.

The method used in the experiments detailed later is as follows: the fitness function notes, for each virtual student, the number of instances of the following two 'offenses': a) four events are scheduled in one day for this virtual student; b) five or more events are scheduled in one day for this virtual student. A different penalty term is associated with each, and the penalty weighted sum of instances of these offenses, summed over virtual students, makes up the contribution to (or, rather, detraction from ...) fitness of the overall event spread constraint.

Finally, 'different-day' constraints can clearly be handled in the same way as Options constraints. For any pair of lectures on the same module, we simply check directly from the chromosome whether or not their assigned slots are on the same day. If they are, then an appropriate penalty is added.

Exclusions and Specifications

It is easy to see how exclusion constraints can be directly translated into one or more simple violation-check functions, given the chromosome representation in use. Notice however that we can just as simply pre-arrange it so that chromosomes never violate these constraints in the first place. We can doctor the allele range of each gene so that it is always the specified allele (if any), or only ranges over the non-excluded alleles.

Choosing between such pre-satisfaction of exclusion and specification constraints, and the option of penalising violations of them, is not always straightforward. If many such constraints exist, the 'pre-satisfied' space may well lack excellent timetables which violate a few exclusions, for example, but make up for this in other ways. On the other hand, pre-satisfaction speeds up evaluation and promises to speed up search via reducing the search space. The full ramifications of this choice are beyond the scope of this paper, but it suffices to point out here that either option should be available in a system which implements this technique. In the experiments discussed later, most exclusion and specification constraints were prespecified. The only 'penalised' such constraint was that for lunchtime lectures. According to the EDAI MSc course organisers, it is preferable to avoid these, but acceptable to trade these off against other constraint violations.

Capacity constraints

To check that a room's capacity isn't exceeded, we must first translate the overall room capacity constraint into constraints of the form "room r should contain no more than r_{cap} students in timeslot t", for each room r and timeslot t, where r_{cap} is the student capacity of room r. During evaluation, the system simply precomputes from the current candidate timetable the student load for each room in each slot, and then runs through this list of constraints checking each in turn and accumulating penalties for violations. Evidently, the same technique can be applied for a very wide range of similar problems, and also applies to constraints concerned with 'teaching loads' constraints in problems for which agents must be considered in the representation.

Juxtaposition Constraints

Finally, it is evident how the ordering constraints between given groups of events can be incorporated. We first derive a collection of binary constraints from those given. Eg, "all lisp tutorials should occur later than the first lisp lecture of the week" is translated into a collection of binary constraints of the form "lisp_l must be before lisp_t3". Checking for violations of such constraints is then straightforward . Similarly, it should be clear how any juxtaposition constraint (eg: "there should be at least two days between event e_1 and event e_2") can be similarly handled.

5 Experiments

Experimental Setup

We address the EDAI MSc lecture/tutorial problems for both terms of the academic years 92/93 and 93/94, respectively involving 76, 73, 82, and 73 events. Other features of these problems are as described in section 4, and full details are available from the authors.

In all cases, the EA used a population size of 50, uniform crossover, gene-by-gene mutation, and elitist generational reproduction. The crossover and mutation rates p_C and p_M were dynamically altered as follows. p_C started at 0.8 and was decreased by 0.001 after each generation (ie: after every 50 evaluations), with a lower limit of 0.6, while p_M started at 0.003 and increased by 0.0003 each generation, with an upper limit of 0.02. Each trial was run for 200 generations (10000 evaluations). Separate experiments are recorded for each problem for each of three different selection strategies: fitness-proportionate (FIT), GENITOR-style rank-based with bias 2 (RANK) [20], and tournament selection with tournament size 10 (TOUR).

The result of a trial was a maximally fit timetable found during the trial. From this we record a vector of violations $V = \{c, j, p, l, e, d_{5+}, d_4, t\}$, which respectively denote violations of Options constraints (clashes), *important* juxtaposition constraints, *desirable* juxtaposition constraints, lunchtime constraints, slot-exclusion constraints, cases where a 'virtual student' faced more than four events in a a day, cases where a 'virtual student' faced four events in a single day, and, finally, travel time constraints. Violations of all other constraints mentioned above (eg: room exclusions, capacity constraints) are not recorded, since they were fully satisfied in all cases.

Ten trials were run for each experiment, and results for each problem record the best V found overall, and the mean of V over the ten trials, for each of three selection schemes. We also present V for the timetables produced by the course organisers for each problem, and which were the actual timetables used (or in use), since the EA system itself was not yet in regular use. 'Best' means relative to the penalty-weighted sum of violations. The fixed penalty values used in these trials for the various violations were, in the order in which they appear in the tables: 500, 300, 30, 30, 10, 5, 1, 1. Hence, violations are listed in order of decreasing importance, as judged by the course organisers.

5.1 Results

Problem	c	j	p	l	e	d_{5+}	d_4	t
92/93_term1								
Course Organisers	0	0	4	4	14	0	2	65
FIT / Best of 10	0	0	0	0	0	0	0	8
FIT / Mean of 10	0	0.1	0.2	1.2	0	0	0.9	16.6
RANK / Best of 10	0	0	0	0	0	0	0	17
RANK / Mean of 10	0	0	0	1.6	0	0	0.6	17.2
TOUR / Best of 10	0	0	0	0	0	0	0	0
TOUR / Mean of 10	0	0	0	1	0	0	0.4	0.7
92/93_term2								
Course Organisers	0	1	9	2	0	1	4	49
FIT / Best of 10	0	0	0	0	0	0	1	0
FIT / Mean of 10	0	0	0.1	0.5	0	0	2.1	9.2
RANK / Best of 10	0	0	0	0	0	0	3	9
RANK / Mean of 10	0	0	0.5	1	0	0	2.4	13.4
TOUR / Best of 10	0	0	0	0	0	0	0	0
TOUR / Mean of 10	0	0	0.3	0.6	0	0	0.7	0.1

Table 1: Comparative performance on the 93/94 problems

Tables 1 and 2 clearly show that the EA approach leads to much better timetables in each case. One course-organiser produced timetable failed to keep all reasonable options open (ie: had clashes), while many failed to fully keep lectures and tutorials away from various restricted slots, and all failed constraints at least as important as the need to avoid lunchtime events. Problems caused involve lecturers and tutors being forced to work during timeslots previously designated for other things (eg: regular weekly seminars), forced to give up free afternoons or mornings designated for research, students facing excessively demanding days, and so on.

On the other hand, for each problem, tournament selection found either a perfect timetable or one with a single travel-time violation in at least one of ten trial runs. For each selection scheme, mean and best results compared very favourably with the experts' efforts. Each EA trial was completed within 5 minutes on a sun SPARC. Occasionally, a EA solution, was worse in terms of some attribute (eg: violations of desirable juxtaposition constraints) than the course organisers' solution, but better overall in terms of the penalty-weighted sum of violations. This suggests that the EA was more successful at trading off the relative occurrences of violations of different importance.

Tournament selection appears to be the best choice, with rank-based selection of the style used in [20] and fitness proportionate selection, in that order, being next best. This relative performance of different selection schemes cannot strictly be taken as read from these results without further experiments using different tournament sizes, biases, and so on; however, much EA literature backs up this ordering of relative performance.

Problem	c	j	p	l	e	d_{5+}	d_4	t
93/94_term1								
Course Organisers	0	0	0	5	7	0	2	84
FIT / Best of 10	0	0	0	2	0	0	0	15
FIT / Mean of 10	0	0	2.6	2.3	0	0	0.7	28.6
RANK / Best of 10	0	0	1	2	0	0	0	23
RANK / Mean of 10	0	0	2.2	2.5	0	0.1	0.3	31.5
TOUR / Best of 10	0	0	0	0	0	0	0	1
TOUR / Mean of 10	0	0	0.6	1.4	0	0	0.2	0.6
93/94_term2								
Course Organisers	2	0	3	3	1	0	0	50
FIT / Best of 10	0	0	1	0	0	0	0	11
FIT / Mean of 10	0	0	0.2	1.4	0	0	1.4	14
RANK / Best of 10	0	0	0	1	0	0	1	13
RANK / Mean of 10	0	0	0.6	1.7	0	0	1.3	13.6
TOUR / Best of 10	0	0	0	0	0	0	0	0
TOUR / Mean of 10	0	0	0	0.7	0	0	0.2	0.2

Table 2: Comparative performance on the 93/94 problems

6 Discussion

The results clearly indicate the benefits of using a penalty-function based EA approach on this problem, and by implication suggest similar utility for the same approach on similar problems. In designing the penalty-weighted fitness function itself for these experiments, several of the design decisions were *ad hoc*. For example, penalty values were chosen according to a rough judgement of relative importance. Also, there were several other possibilities, as discussed earlier, for dealing with the event-spread constraints. Even the underlying EA itself was far from optimal in terms of parameter settings and general configuration. Better choices of selection scheme, for example, are spatially-oriented schemes as presented in [5] and [9], while better overall choices for the EA are certainly possible.

The 'rough-and-ready' aspect of the experimental configurations used in this paper, coupled with the good results reported and the ease of implementing the approach strongly suggests a promising future for both further research and also practical use of EAs on general timetabling problems. Naturally, there are a considerable number of theoretical and practical issues that need to be answered. An illustrative collection of these follow:

Scaling Up

How does this approach scale up to larger and more tightly constrained problems? The real problems addressed here, and similarly those addressed in [6, 14, 7], are similar in size or larger than a large proportion of the timetabling problems faced in many institutions. Hence the usefulness of this approach seems justified, inasmuch as we can expect the beneficial results displayed here to carry over to different timetabling problems of similar or smaller size. How the approach scales with increasing size and/or complexity is a harder question, which is the subject of continuing research. Initial indications in unpublished work are that the basic approach scales well, but suffers from a problem common to EA-based optimisation: that is, solutions near optimal regions are rapidly found on large complex problems, but further evolution towards optima becomes considerably slow, and may stop altogether. Fortunately this difficulty is readily aided by the use of smart hillclimbing mutation operators. As detailed in [8], use of such operators helps to vastly increase the scope of the approach in terms of problem size. Similarly, an alternative chromosome representation used in [16] is also found to significantly improve on solution quality when compared with the basic approach as presented here.

Generalising Across

How does the approach perform on other timetabling problems? The general nature of the approach suggests that it would be just as well employed on many similar problems. A key aspect which matters here is speed of evaluation. As long as the numbers of constraints which need checking coupled with the computational ease of checking them make for a relatively speedy evaluation function, it seems safe to suggest that useful performance is promised. The kinds of individual constraints that usually occur in timetabling problems are computationally quick to check when using the direct chromosome representation. The general prospects for EA-based timetabling in this respect are in any case illustrated by the variety of real problems so far successfully addressed.

Different Approaches

Different representations, use of domain specific recombination operators, and hybridisation of the EA with other techniques are all candidates for refinement of this approach. Much further research in this vein is in order. It is also interesting and important to compare EA approaches with other methods such as branch & bound search, simulated annealing, and so on. This endeavour is complicated by the differences between the techniques themselves. Eg, the promise of the EA-based approach is most strongly manifest in its robustness across a very wide range of different timetabling problems. Comparison with rule-based approaches to test this claim on the same variety of problems would then necessitate the lengthy and difficult development process of building rule-based systems with similarly wide applicability. Comparison with simulated annealing is a more likely prospect, and such is planned in due course.

Practice

Some common needs are not met by the approach as discussed here. Eg: timetablers may wish to generate several distinct timetables to choose from. Such considerations require refinements and extensions, although the basic approach we have discussed remains a useful partial tool for such requirements. More relevantly, space prevents us from properly covering here the general process and the many possible choices involved in interpreting the constraints of a problem into a particular choice of fitness function. It is rarely apparent how best to do this, and further work is required to assess the various possibilities. Experience shows, however, that there is unlikely to be a major difference in performance between different such choices for problems of the size and type addressed here; hence, we feel that natural and/or arbitrary choices may be made with impunity for penalty settings, pre-specifications, event-spread constraint handling, and so on ..., at least for problems of the size and type found in small or medium sized university departments.

Acknowledgements

We would like to thank Bob Fisher and Alan Smaill for their help in describing the lecture timetabling problems used here and providing data. Thanks also to the UK Science and Engineering Research Council for support of Dave Corne via a grant with reference number GR/J44513.

References

[1] D. Abramson and J. Abela, 'A parallel genetic algorithm for solving the school timetabling problem', Technical report, Division of Information Technology, C.S.I.R.O., (April 1991).

[2] Alan M. Barham and John B. Westwood, 'A simple heuristic to facilitate course timetabling', *Journal of the Operational Research Society*, **29**, 1055–1060, (1978).

[3] Mirjana Cangalovie and Jan A.M. Schreuder, 'Exact colouring algorithm for weighted graph applied to timetabling problems with lectures of different lengths', *European Journal of operations research*, **51**, 248–258, (1991).

[4] Michael W. Carter, 'A survey of pratical applications of examination timetabling algorithms', *Operations Research*, **34**(2), 193–202, (March-April 1986).

[5] Robert J. Collins and David R. Jefferson, 'Selection in massively parallel genetic algorithms', in *Proceedings of the Fourth International Conference on Genetic Algorithms*, eds., R.K. Belew and L.B. Booker, pp. 249–256. San Mateo: Morgan Kaufmann, (1991).

[6] Alberto Colorni, Marco Dorigo, and Vittorio Maniezzo, 'Genetic algorithms and highly constrained problems: The time-table case', in *Parallel Problem Solving from Nature*, eds., G. Goos and J. Hartmanis, 55–59, Springer-Verlag, (1990).

[7] Dave Corne, Hsiao-Lan Fang, and Chris Mellish, 'Solving the module exam scheduling problem with genetic algorithms', in *Proceedings of the Sixth International Conference in Industrial and Engineering Applications of Artificial Intelligence and Expert Systems*, eds., Paul W.H. Chung, Gillian Lovegrove, and Moonis Ali, 370–373, Gordon and Breach Science Publishers, (1993).

[8] Dave Corne, Peter Ross, and Hsiao-Lan Fang, 'Fast practical evaolutionary timetabling', in *Proceedings of the AISB Workshop on Evolutionary Computation*, (1994).

[9] Yural Davidor, 'A naturally occuring niche & species phenomenon: The model and first results', in *Proceedings of the Fourth International Conference on Genetic Algorithms*, eds., R.K. Belew and L.B. Booker, pp. 257–263. San Mateo: Morgan Kaufmann, (1991).

[10] *Handbook of Genetic Algorithms*, ed., L. Davis, New York: Van Nostrand Reinhold, 1991.

[11] K. A. Dowsland, 'A timetabling problem in which clashes are inevitable', *Journal of operations research society*, **41**, 907–918, (1990).

[12] David E. Goldberg, *Genetic Algorithms in Search, Optimization & Machine Learning*, Reading: Addison Wesley, 1989.

[13] Sami Khuri, Thomas Bäck, and Jörg Heitkötter, 'An evolutionary approach to combinatorial optimization problems', in *Proceedings of the 1994 Computer Science Conference (CSC94)*, Phoenix, Arizona, (March 1994). ACM Press. to appear.

[14] Si-Eng Ling, 'Intergating genetic algorithms with a prolog assignment problem as a hybrid solution for a polytechnic timetable problem', in *Parallel Problem Solving from Nature, 2*, eds., R. Manner and B. Manderick, 321–329, Elsevier Science Publisher B.V., (1992).

[15] N. K. Mehta, 'The application of a graph coloring method to an examination scheduling problem', *Interfaces*, **11**, 57–64, (1981).

[16] B. Paechter, H. Luchian, A. Cumming, and M. Petruic, 'Two solutions to the general timetable poblem using evolutionary methods', in *Proceedings of the IEEE Conference on Evolutionary Computation*, (1994).

[17] Ben Paechter, 'Optimising a presentation timetable using evolutionary algorithms', in *Proceedings of the AISB Workshop on Evolutionary Computation*, (1994).

[18] Peter Ross, Dave Corne, and Hsiao-Lan Fang, 'Timetabling by genetic algorithms: Issues and approaches', Technical Report AIGA-006-94, Department of Artificial Intelligence, University of Edinburgh, (1994). revised version to appear in Applied Intelligence.

[19] Alice E. Smith and David M. Tate, 'Genetic optimisation using a penalty function', in *Proceedings of the Fifth International Conference on Genetic Algorithms*, ed., S. Forrest, pp. 499–503. San Mateo: Morgan Kaufmann, (1993).

[20] Darrell Whitley, 'The GENITOR algorithm and selection pressure', in *Proceedings of the Third International Conference on Genetic Algorithms*, ed., J. D. Schaffer, 116–121, San Mateo: Morgan Kaufmann, (1989).