Improving Evolutionary Timetabling with Delta Evaluation and Directed Mutation

Peter Ross, Dave Corne, and Hsiao-Lan Fang

DAI Research Paper No. 707

To appear in Parallel Problem Solving from Nature III, Springer Verlag, 1994.

Improving Evolutionary Timetabling with Delta Evaluation and Directed Mutation

Peter Ross, Dave Corne, Hsiao-Lan Fang

Department of Artificial Intelligence, University of Edinburgh, 80 South Bridge, Edinburgh EH1 1HN, U.K.

Abstract. Researchers are turning more and more to evolutionary algorithms (EAs) as a flexible and effective technique for addressing timetabling problems in their institutions. We present a class of specialised mutation operators for use in conjunction with the commonly employed penalty-function based EA approach to timetabling which shows significant improvement in performance over a range of real and realistic problems. We also discuss the use of *delta evaluation*, an obvious and recommended technique which speeds up the implementation of the approach, and leads to a more pertinent measure of speed than the commonly used 'number of evaluations'. A suite of realistically difficult benchmark timetabling problems is described and made available for use in comparative research.

1 Introduction

Recent research suggests that Evolutionary Algorithms (EAs) are a viable and effective method for addressing timetabling problems [2, 1, 7, 3, 5, 8]. Following the general success shown in these endeavours, the general importance and ubiquity of timetabling problems now warrants systematic attempts to assess the general success of applying EAs to them. That is, there is a need for some standard problem definitions, benchmark problems, benchmark results, and standardised techniques for reference. Towards this end, we discuss a common penalty-function based approach to timetabling, outline a working definition for general timetabling problems, and presents several benchmark problems. In particular some domain-specific operators are described, generally applicable to all timetabling problems, and compared empirically with standard operators on a range of realistic problems.

Our timetabling experiments employ the commonly used (but often *not* used) implementation technique we call *delta evaluation*, whereby the evaluation of a new individual is speeded up by making use of previously evaluated similar individuals. We discuss this for two main reasons: (a) to emphasise and encourage its use as a way of significantly speeding up EA/timetabling applications (especially those involving the direct representation discussed later), and (b) given the use of delta evaluation as standard in this endeavour, it makes sense to record 'machine-independent' indicators of speed in terms other than simply 'number of evaluations', since the speed of an evaluation will vary greatly throughout the process. We hence describe the idea of 'evaluation equivalents' as a useful comparative measure.

We begin in Sect. 2 with a brief description of a common kind of timetabling problem. Section 3 then reviews the basic penalty-function approach, and introduces several new candidate genetic operators, and discusses the use of delta evaluation. Section 4 describes a family of benchmark timetabling problems we are developing, attending mainly to those we use later in experiments. Section 5 describes and records a collection of experiments comparing the performance of the operators described earlier on some of these benchmarks. Finally, Sect. 6 provides some discussion and summary of the paper.

2 Basic Timetabling problems

Timetabling problems involve a set of events $E = \{e_1, e_2, \ldots e_v\}$, and a set of times $T = \{t_1, t_2, \ldots, t_s\}$. Timetablers often also need to take into account a set of places $P = \{p_1, p_2, \ldots, p_m\}$, and/or a set of agents $A = \{a_1, a_2, \ldots, a_n\}$. An *assignment* is a 4-tuple (e, t, p, a) such that $e \in E$, $t \in T$, $p \in P$, and $a \in A$, with the simple interpretation: "event e occurs at time t in place p involving agent a". In a real case, this may for example mean: "The AI lecture starts at 9:00 in room LT-5, given by Minsky".

A timetable is simply a collection of assignments, one per event. The problem is to find a timetable that satisfies, or minimally violates, a (usually large) collection of constraints. The most common such constraint is simply that there should be no clashes; that is, a person should not have to be in two places at once. Considering the relationship with graph colouring problems, we call this an 'edge constraint'. A related and important constraint is what we call an 'event-spread' constraint, which expresses that a person should normally have at least a certain amount of time free between certain of the events in which he or she is involved. There are several other kinds of constraints. The most prominent among them are illustrated in Table 1. This table defines common kinds of constraints in terms of inequalities over assignments (or sets of them). In the table, $t(e)$ refers to the time assignment of event e; an ordering is assumed over the set of times T; e and f are events where, without loss of generality, we assume that $t(e) < t(f)$; and $l(e)$ stands for the duration of event e. Note that from now on in this paper, and for space reasons in Table 1, we will look at problems involving only time assignments, ignoring for now any constraints involving places or agents.

This is a convenience for present purposes, rather than any gross simplification. Place and agent constraints are often easy to handle in exam timetabling problems, for example, and such assignments can be made manually after the harder job of producing the events/times timetable. Also, constraints involving rooms and agents can often be dealt with implicitly via the constraints detailed in Table 1 alone. In many cases, of course, room and agent constraints do make the problem significantly harder; we begin to address this type of problem in [6], for example.

Table 1. Kinds of Timetabling Constraint

Constraint Type	Name	Assignment version
No clash between e and f	Edge	$t(e) + l(e) \leq t(f)$
Spare time s between e and f	Event-Spread	$t(e) + l(e) + s \leq t(f)$
Excluded times S for e	Exclusion	$NOT(t(e) \in S)$
Specified time u for e	Specification	$t(e) = u$
e must be before/after/same-time f	Juxtaposition	$t(e)(<\|>\|=)t(f)$

3 Timetabling with EAs

The most commonly employed EA approach is the use of a direct representation coupled with a penalty-allocating fitness function. A timetable is represented by a chromosome in which the allele of the ith gene directly represents the time assigned to the ith event; extensions dealing with room and/or agent assignments may be readily imagined. Fitness is simply a weighted sum of constraint violations; ie: high penalties accrue from violations of important constraints, while low penalties are given for soft or unimportant constraint violations. The EA then attempts to mimimise this sum.

It is wise to use *delta evaluation*, in which the computation of a chromosome's fitness is simplified as follows. Consider two timetables, g and h, which differ only in the assignments made to some subset D of the events E. Let $P(t) = \sum_{c \in C} w_c v(c,t)$ be the total penalty accruing to timetable t, where w_c is the weight associated with constraint c and $v(c,t)$ is 1 (0) if constraint c is (not) violated in t. C is the set of constraints. Now, if C_D is the subset of C containing only those constraints which involve one or more events from the subset D of E, then we can easily deduce:

$$P(h) = P(g) - \sum_{c \in C_D} w_c v(c,g) + \sum_{c \in C_D} w_c v(c,h) \tag{1}$$

which expresses P(h) solely in terms of constraints involving the events in D. If the size of C_D is small in relation to C, then this promises to save much time. In general, if h differs from g in k places, then evaluating g needs $2k$ 'constraint-checks'.

We can use of the concept of a 'constraint-check' to form a convenient measure of the speed of an EA/timetabling run. Number of evaluations is no longer a useful measure of this, because the time of an evaluation will vary quite significantly. By summing the total number of constraint-checks, we can thus form more useful cross-comparisons. Further, by noting the constant number of constraint checks that would be needed in a *full* evaluation, we can convert a constraint-checks sum into *evaluation equivalents* (EEs). This gives both a pertinent measure for comparing speed of different configurations, and also enables us to easily see the speedup effect of using delta evaluation.

Applying delta evaluation of timetable h in an EA needs a decision as to which already-evaluated timetable to take as g. A natural choice, which we use, choose g arbitrarily from the parents of h (or just use the single parent, if h was produced via a single parent operator).

3.1 Violation-Directed Operators

Consider the process involved in working out the penalty score for a timetable t; the major part of the process involves checking each constraint in turn. While doing this, extra steps may easily be incorporated which help keep track of the events whose assignments seem to be causing the most difficulty. This might be done as follows. Keep a separate 'violation score' v_i for each event e_i; initially (before evaluation) set $v_i = 0$ for each i. Then, for each constraint c, if $v(c, t) = 1$ (and hence the constraint is violated), add w_c to the violation scores of each event involved in c. At the end of this process, the set of violation scores can be used to infer which events are most problematic in t, and hence inform as to where in the chromosome it would probably be best to direct mutation. We can put this information to use in a variety of ways. the way we examine in this paper is what we call *violation directed mutation* (VDM).

There are two key aspects to VDM; the choice of a gene to mutate, and the choice of allele to mutate it to. Three possibilities for the former are: (a) simply choose the 'best' candidate for mutation, ie: randomly choose one of a set of genes with maximal violation score, or (b) stochastically select one, biased towards genes with higher violation scores, or (c) simply choose one at random (in which case the 'directedness' of the mutation will arise through the later choice of allele).

For allele choice, it is simple to conceive of a directly similar set of possibilities; this needs, though, some analogue of 'violation score' for alleles. This can be done as follows: if the gene chosen for mutation is g, then a violation score for the candidate new allele a is calculated by simply amassing the penalty-weighted sum of violations of constraints involving g which would occur if it was given allele a.

In both cases, the 'stochastic selection' aspect can be done in various ways; in fact we can use almost any standard EA selection scheme. We choose to use straightforward tournament selection for this. A good reason for this choice follows from the added computational complexity associated with the allele-choice operation; rank-based or fitness-based selection, for example, would require us to calculate, every time, a violation score for each possible new allele, which means checking s times through the list of all constraints involved with g (where s is the number of alleles). Using tournament selection with a tournament size of k, however, this only need be done k times at each application of the VDM operator. Notice that we can absorb the extra time complexity of VDM, which only arises significantly when the allele choice component is other than `rand`, into our 'evaluation equivalents' notion by simply recording the number of constraint checks made during the allele choice operation.

We shall refer to variants of the VDM operator as ordered pairs (g,a), where g stands for the gene choice operation, and a stands for the allele choice operation. Instances we look at later will involve rand (random choice), tnK (tournament selection with a given tournament size of K), and best (choose a gene (allele) with a maximal (minimal) violation score).

Finally, we note that very similar operators are described by Eiben *et al*, among others, in [4] for use on graph-colouring and other benchmark constraint satisfaction problems. The differences are that Eiben also considers variants of the operator in which either one, two, or a random number of genes are mutated each time, whereas we only look at single gene mutations here, and Eiben *et al* do not look at variations of bias in the stochastic choice component, and do not look at a best component.

4 Benchmark Timetabling problems

A fully general timetabling problem involves, as we have suggested time, place, and agent assignments along with many and varied kinds of constraints involving, for example, room capacities and teaching loads, as well as those already mentioned. Benchmark problems of this sort are in preparation, but for convenience it makes sense to also investigate a simpler variant of the general problem which is, nevertheless, both realistic and common. Here we will describe one template for such a problem, based on the recurring EDAI[1] multi-departmental modular MSc examination timetabling problem (edai-ett).

The edai-ett involves v events, all of the same duration, and s timeslots spread out evenly over d days. There are four timeslots per day. The problem involves edge constraints, event-spread constraints, and exclusions. The overall event-spread objective is that, whenever a pair of events is edge constrained, these events should also (as well as being in different slots) either be on different days, or respecively occupy the first and third, or second and fourth slot of a day. Our edai-ett benchmark set contains the four years' versions of the real problem, coupled with an arbitrarily large collection of randomly generated solvable problems of the same type produced via a problem generator.

Randomly generated problems of the edai-ett type are based on the construction of a random complete timetable $\mathcal{T}$ of 50 events within a four-slots-per-day, nine-day timetable structure. A set of edge constraints is then generated which render $\mathcal{T}$'s particular partition of events into days, coupled with certain details of its partition of a day's events into slots, as unique in satisfying these edge constraints. A set of exclusion constraints is also generated, which by themselves make $\mathcal{T}$ the only solution. A problem itself is then constructed by filtering these edges and exclusions. For example edai-ett-rand(3,40,10) is a problem involving version 3 of $\mathcal{T}$, and containing 40% of the generated edges and 10% of the exclusions.

The benchmark set we are developing is much along the lines discussed; for each real timetabling problem it contains, there is also a generator for random

[1] University of Edinburgh Department of Artificial Intelligence

variants of problems of the same type. Here we will experiment with a small but useful sample of these problems.

5 Experiments

Two general questions guided the following set of experiments; a) What is the most successful variant of violation-directed mutation on timetabling problems?, and b) How can we generally expect performance to vary as we alter the difficulty of the problem? These are both extremely difficult questions to investigate, owing to the multiplicity of different variations on timetabling problems that can be imagined, to the many distinct (and interacting) ways in which we could alter the so-called 'difficulty' of such a problem, to the many variations possible on violation directed mutation, and also to inevitable restrictions on computational resources. The approach we take is to simplify the questions, and hence the investigation, without, we believe, sacrificing the usefulness of our results. Our investigation thus concentrates on these two issues: (a) How do a reasonable set of variants on violation-directed mutation compare on a typical exam timetabling problem?, and (b) How does performance of the best such variant behave as we vary the number of constraints on a similarly structured problem? To address (a), we investigated several variants of violation directed mutation on `edai-ett-93`, and to address (b) we looked at `edai-ett-rand(1,N,15)` for each of N from 20 to 100 in steps of 20.

5.1 EA Configuration

All experiments shared the following common EA configuration: A reproduction cycle consisted of a breeding step (in which one new chromosome was produced) followed by an insertion step, in which this new chromosome replaced the currently least fit (if strictly fitter itself). With probability 0.2, a breeding step involved the selection of one parent and the simple gene-wise mutation of it with a probability of 0.02 of randomly reassigning the allele of each gene in turn. With probability 0.8, a breeding step involved the selection of one parent (or two, when the operator was uniform crossover)and the application of the given operator. Tournament selection was used with a tournament size of 6, and the population size was always 1,000.

5.2 Variations on Directed Mutation

Several variations on violation directed mutation were explored using the real `edai-ett-93` problem as a benchmark. For each variant of the VDM operator. A trial consists of running the EA for 20,000 reproduction cycles, or until a perfect timetable was found. If convergence occurred to a non-perfect timetable during a trial (the entire population represented the same non-perfect timetable), then a *complete* reinitialisation of the population occurred at the next reproduction step — this counted as 1 reproduction cycle, but 1,000 evaluations (translated further into the appropriate number of evaluation equivalents). For each variation of the VDM operator, 100 trials were performed with a different random seed each time. Eight results are given for each set of trials: the number of trials in which an optimum was found (which, as a percentage, can be taken as a measure of the reliability of the EA variant on this problem), the mean number of constraints violated by the best timetable over all trials, the lowest, mean, and highest number of evaluations recorded to find an optimum over those trials in which an optimum was found, and the lowest, mean, and highest number of EEs recorded to find an optimum over those trials in which an optimum was found.

Table 2. Performance of VDM Variants on edai-ett-93

	No. Perfect	Mean	Evaluations			Eval. Equiv's		
VDM Variant	Trials	Violations	Lowest	Mean	Highest	Lowest	Mean	Highest
uniform	0	3.8	n/a					
(rand, rand)	3	4.2	18704	19733	19950	1610	1701	1940
(rand, tn3)	80	0.4	10613	13706	17417	3168	4109	5247
(rand, tn6)	92	0.1	7591	10598	15458	3881	5427	7906
(rand, tn9)	98	0.02	6146	8398	11259	4450	6086	8187
(rand, best)	0	29.0	n/a					
(tn3, rand)	5	2.1	17063	17660	18257	1532	1573	1613
(tn3, tn3)	65	1.4	11713	16344	19655	4717	10194	12248
(tn3, tn6)	83	0.3	6752	9160	12090	3603	4796	6190
(tn3, tn9)	93	0.9	5326	8530	19119	4029	6403	15190
(tn3, best)	0	22.4	n/a					
(tn6, rand)	8	2.1	17874	18811	19746	1646	1725	1779
(tn6, tn3)	88	0.20	10463	13930	18644	3175	4335	5687
(tn6, tn6)	82	0.03	6741	9458	13242	3537	4969	7061
(tn6, tn9)	97	0.03	5831	9572	20583	4304	7266	16309
(tn6, best)	0	22.4	n/a					
(tn9, rand)	3	3.1	12120	15756	16044	1237	1419	1554
(tn9, tn3)	77	0.3	10228	14880	18909	3253	4667	6242
(tn9, tn6)	92	0.08	7484	10656	13802	4141	5623	7532
(tn9, tn9)	85	0.19	5359	10684	20771	3875	8186	17121
(tn9, best)	0	23.9	n/a					
(best, rand)	0	20.7	n/a					
(best, tn3)	0	21.5	n/a					
(best, tn6)	0	19.1	n/a					
(best, tn9)	0	13.9	n/a					
(best, best)	0	30.1	n/a					

Clearly enough, it seems that stochastic variations of VDM, in which both the choice of gene to mutate and choice of new allele are biased, stochastic choices, are superior to basic uniform crossover on this problem. Points to note are how reliability varies with the selection pressure for these choices, and also how evaluation equivalents, rather than evaluations, are clearly a more useful indicator of speed than number of evaluations. To see the latter, note how mean EEs rises as we increase selection pressure (because we are performing more computations in, especially, the 'choice of allele' side of the operator), although mean evaluations falls. For example, `(rand, tn6)` and `(tn3, tn9)` are closely comparable in reliability on this problem, and a glance at the mean number of evaluations would seem to indicate that `(tn3, tn9)` is the better of the two owing to its speed. This would be true if we *had to* use full evaluation; using delta evaluation though, it is clear from the mean EEs figures that `(rand, tn6)` uses significantly fewer constraint checks to achieve similar reliability, and is hence overall the better of the two on this problem.

One clear point is the different degrees to which gene choice and allele choice matter. Looking solely at variations in performance with allele choice fixed, we find there is little difference between, say, `(rand, tn3)` and `(tn9, tn3)`. Allele choice is rather more significant. For example, `(tn3, tn9)` is much more reliable than `(tn3, rand)`. Whenever `best` appears for either allele choice or gene choice however, performance plummets. Another interesting point is that although biased gene choice is better than `rand` when the allele choice component is `rand`, it generally leads to slight degradation in reliability when the allele choice component is stochastic.

5.3 Increasingly Hard `edai-ett-rand` Problems

Some VDM variants were applied to each of 5 randomly generated solvable `edai-ett`-like problems of increasing constrainedness. The problems addressed were `edai-ett-rand(1,N,15)` for N from 20 to 100 in steps of 20. Using the same EA configuration as above (except of course for the operator choice), each entry in Table 3 records the number of optima found over 100 trials, and the mean number of EEs over those trials which found an optimum, for the configuration using the operator defined by the row, and on the problem defined by the column. Taking heed of the results discussed above, we look only at variation in bias in biased, stochastic versions of the allele choice component, keeping gene choice fixed at `rand`.

Table 3. Performance of VDM on `edai-ett-rand(1,N,15)` for N from 20 to 100

	Reliability/Mean EEs				
Operator	N=20	N=40	N=60	N=80	N=100
(rand, tn3)	100/466	100/1715	100/5257	33/10323	0/—
(rand, tn6)	100/669	100/2244	100/6133	100/13470	58/21966
(rand, tn9)	100/906	100/2839	100/7359	100/15752	97/26440
(rand, tn12)	100/1202	100/3519	100/8679	100/17837	99/29852
(rand tn15)	100/1439	100/4160	100/10059	100/19913	100/33105

Table 3 reveals some clear trends. For the simpler problems, with N = 20, 40, and 60, all configurations are 100% reliable at finding an optimum, however the lower bias versions are better simply because they are faster. Again, this is directly attributable to taking advantage of delta evaluation. Number of evaluations actually reduces with increased bias, but the extra constraint checks done during allele choice more than counteract this. Hence, low-biased VDM seems best on simpler problems. As the problem gets harder, the same remains true in the sense that lower-bias versions are generally faster when they find an optimum, although they are rather less reliable. Overall, on the basis that the differences in speed are rather less significant than differences in reliability at finding an optimum, general indications are that higher pressure in the allele choice component is the preferred option for timetabling problems of this type; although not too high, since `best` rapidly decreases in reliability as N increases[2]. The critical region, at which higher bias starts to lose reliability on these problems, has yet to be found.

6 Concluding Discussion

We have shown that realistic timetabling problems can be effectively addressed by an EA whose main operator is VDM, particularly when incorporating a biased, stochastic method for the allele choice component, and either a random or biased stochastic method for the gene choice component. Such an EA is certainly far more effective than a straightforward EA using uniform crossover, achieving, for example 97% reliability on the `edai-ett-93` problem in certain configurations, as opposed to the 0% reliability of uniform crossover. In particular, we have found EA configurations to be very useful and effective on a range of other real timetabling problems in addition to `edai-ett-93` [5, 6].

Looking at the relative effect of different variants of VDM, it seems that the gene choice component may as well be random, while the real power and effect of the operator lies in the allele choice component. This is quite good news for implementation purposes; the combination of delta evaluation with a non-random gene-choice component at least doubles the memory requirement, since the chromosome must carry its gene violation scores around, and also makes the mechanics of delta evaluation more tricky. Results indicate, however, that it is not really necessary to have a non-random gene choice component in VDM, which alleviates these problems.

Many difficult and interesting questions remain in the EA/timetabling research arena. Here we have concentrated on one style of approach, and subsequent enhancements to its performance. General statements on the performance of EAs on timetabling is however complicated by several important factors: first, alternative styles of EA approach are also worth investigating; for example, those in which the chromosome is *indirect*, representing a timetable via a list of instructions, or a particular ordering of events, for later interpretation by a timetable

[2] Space restrictions prevent a fuller presentation of results with other configurations etc ..., but such is available from the authors.

building procedure. EA/timetabling work along these lines is pursued in [8]. Second, there are many more choices of operator possible, including further variants on mutation (single parent operators) and several alternative recombination operators worth investigating. Some of these are looked at in [5]. Third, there is great variation in the space of possible general timetabling problems. We are currently looking for useful landmarks in this space, but for the time being we take the satisficing route of examining EA performance on real timetabling problems we have encountered, and constructing further test problems based on these.

In the interests of further promoting comparative research on this important kind of problem, we encourage researchers to use the benchmarks we have looked at here, and others, which can all be obtained (and explained) via the authors, or directly from the FTP site ftp.dai.ed.ac.uk.

References

1. Abramson D., Abela, J. : A Parallel Genetic Algorithm for Solving the School Timetabling Problem. IJCAI workshop on Parallel Processing in AI, Sydney, August 1991
2. Colorni, A., Dorigo, M., Maniezzo, V.: Genetic Algorithms and Highly Constrained Problems: The Time-Table Case. Parallel Problem Solving from Nature I, Goos and Hartmanis (eds.) Springer-Verlag, 1990, pages 55–59
3. Corne, D., Fang H-L., Mellish, C.: Solving the Module Exam Scheduling Problem with Genetic Algorithms. Proceedings of the Sixth International Conference in Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, Chung, Lovegrove and Ali (eds.), 1993, pages 370–373.
4. Eiben, A.E., Raue, P.E., Ruttkay, Z Heuristic Genetic Algorithms for Constrained Problems. Working papers of the Dutch AI Conference, 1993, Twente, pages 341–353.
5. Corne, D., Ross, P., and Fang, H-L.: Fast Practical Evolutionary Timetabling. Proceedings of the AISB Workshop on Evolutionary Computation, Springer Verlag, 1994, to appear.
6. Ross, P., Corne, D., and Fang, H-L.: Successful Lecture Timetabling with Evolutionary Algorithms. Proceedings of the ECAI 94 Workshop on Applications of Evolutionary Algorithms, 1994, to appear.
7. Ling, S-E.: Intergating Genetic Algorithms with a Prolog Assignment Problem as a Hybrid Solution for a Polytechnic Timetable Problem. Parallel Problem Solving from Nature 2, Elsevier Science Publisher B.V., Manner and Manderick (eds.), 1992, pages 321–329.
8. Paechter, B. Optimising a Presentation Timetable Using Evolutionary Algorithms. Proceedings of the AISB Workshop on Evolutionary Computation, Springer Verlag, 1994, to appear.