An Order Based Evolutionary Approach to Dual Objective Examination Timetabling

Christine L. Mumford

Abstract— **This paper explores a simple bi-objective evolutionary approach to the examination timetabling problem. The new algorithm handles two hard constraints: 1) avoiding examination clashes and 2) respecting the given maximum seating capacity; while simultaneously minimizing two objective functions: 1) the overall length of the examination period and 2) the total proximity cost. An order based representation with a greedy decoder ensures that neither of the hard constraints is violated, and produces only feasible timetables. At the same time the dual objectives are attacked and the multi-objective evolutionary algorithm (MOEA) attempts to pack all the examinations into as short a period as possible while, at the same time, favoring a good spread of examinations for individual students. Most other published timetabling algorithms require the number time slots to be fixed in advance of any optimization for soft constraints, such as proximity costs. Smart genetic and heuristic operators used in the present study ensure that a good set of non-dominated results is produced by the new MOEA, covering a range of timetable lengths.**

I. Introduction

Examination timetabling belongs to a large group of NP-hard problems collectively known as *set partitioning problems*. The examination timetabling problem involves scheduling a set of examinations into a number of time slots in such a way that the resulting timetable complies with any hard constraints and also gives due consideration to other features considered desirable in a "good" timetable. Many different variants exist for this important real-world problem, and the choice of practical solution method will depend on the types of constraints involved and also on the objectives that need to be optimized (see [4] and [17] for more details). In its most basic version, the examination timetabling problem is identical to the graph coloring problem, with the colors representing time slots and vertices representing examinations. In this model, an undirected edge between vertices indicates that at least one student is taking both exams. The goal is to schedule all the examinations in the minimum number of time slots, so that there are no clashes.

In practice, available resources are finite and additional hard constraints may be imposed, over and above the requirement to schedule examinations with no clashes. A university has an upper limit on the number of candidates it can seat in a time slot, for example. Interestingly, the seat capacity limitation is identical to the weight capacity constraint for the bin packing problem: the items of various sizes (bin packing) being replaced with examinations having various numbers of candidates (timetabling). Thus, a feasible solution to the timetabling problem that avoids all clashes and seats all students requires the simultaneous solution to the underlying graph coloring and bin packing problems. Other common constraints include requirements to schedule some exams before others and restrictions limiting certain exams to specific rooms, if special resources are required.

In addition to the various hard constraints imposed by different institutions, universities have differing views as to what constitutes a "good" timetable, as opposed to simply a feasible one. Most commonly these desirable but not essential properties (often referred to as *soft constraints*) include some measure of a "fair spread" of examinations for the students taking them. A schedule that requires students to take two examinations in consecutive time slots is usually avoided if possible, for example. Indeed, some institutions will go much further than this to ensure that as many students as possible have good revision gaps between their examinations. Universities may also consider issues that affect the staff involved in marking the scripts. For example, examinations with large numbers of candidates may be scheduled early to give more time for marking. In the present study the following hard and soft constraints apply:

Hard Constraints:

1) Avoiding clashes
2) Keeping within the total seating capacity

Soft Constraints (dual objectives):

1) Minimizing the number of time slots
2) Minimizing near clashes - ensuring a good spread of examinations for individual students

Two main encoding methods can be identified for timetabling, and other set partitioning problems: direct encoding, and order based encoding. With direct encoding arbitrary time slots are first assigned and then heuristics are used to move some examinations in an attempt to improve the solution. In contrast, order based approaches organize the exams into permutation lists, and rely on a greedy decoder to assign the time slots in a methodical way. Direct approaches work to minimize and eventually eliminate conflicts but do not guarantee legal solutions. With heavily and/or multiply constrained problems, it can become increasingly difficult to escape infeasibility when using directly encoded methods. On the other hand, if an order based approach is used, feasibility is usually guaranteed.

In this paper an order based approach is used which builds on previous work on the graph coloring problem

Christine Mumford is with the School of Computer Science, Cardiff University, 5 The Parade, Cardiff, CF24 3AA, United Kingdom (phone: +44 (0)29 20875305; fax: +44 (0)29 20874598; email: C.L.Mumford@cs.cardiff.ac.uk).

by the same author, [16]. The earlier study introduced two new crossover operators that are specially adapted for set partitioning problems. A major contribution to the successful operation of the new crossover operators is the incorporation of some grouping and reordering heuristics, originally devised by Culberson and Luo [8]. These heuristics proved very effective in preprocessing the permutation lists prior to crossover, by grouping and reordering complete color classes. The new crossover operators are specially honed so that offspring inherit complete color classes from their parents, wherever this is possible. Yet, at the same time, the order based greedy assignment process ensures all solutions remain feasible (i.e., that no hard constraint is violated). This is in contrast to the crossovers used in other state-of-the-art methods for set partitioning. The crossover used in the grouping genetic algorithm (GGA) [11], [12], for example, uses repair and/or backtracking to reassign elements and correct conflicts. Similarly, the very successful hybrid evolutionary algorithm of Galinier and Hao, [13], relies on extensive tabu search eliminate the conflicts.

In the present work a simple multi-objective evolutionary algorithm (MOEA) is used to simultaneously minimize the number of time slots and the total proximity cost for the examination timetabling problem. The idea is to pack all the examinations into as short a period as possible, while favoring a good spread of examinations for individual students. Most other published timetabling algorithms require the number time slots to be fixed in advance of any optimization for soft constraints, such as proximity costs. In contrast, a MOEA approach produces results that cover a range of timetable lengths. Although an earlier paper by Pascal Côté, Tony Wong and Robert Sabourin [7] has also tackled the bi-objective optimization of timetable length versus proximity costs, obtaining excellent results, the present approach differs from this ground-breaking work in a number of important ways. The present work:

- adds seating capacity constraints,
- uses an order based representation,
- does not produce infeasible solutions, and thus needs no repair heuristics
- uses a recombination (crossover) operator.

One clear advantage of the present approach is the ease with which it can deal with multiple hard constraints by incorporating appropriate tests for them within the greedy decoder.

II. Key Heuristics, Genetic Operators and Performance Measures used in the Study

A. Culberson and Luo's grouping and reordering heuristics

The grouping and reordering heuristics of Culberson and Luo (CL) [8] play a very important role in preprocessing the chromosomes to make the crossover operator more effective (see [16] for evidence of this). The CL heuristics, originally devised to solve the graph coloring problem, belong to a family of methods that use simple rules to produce orderings of vertices. Once created, the orderings are presented to a greedy decoder for transformation into legal colorings. Successful heuristics of this type can be distinguished by the production of high quality solutions. The simplest and fastest ordering heuristics generate a solution in one go. For the graph coloring problem probably the best known one-shot techniques generate the orderings by placing the most heavily constrained vertices (e.g., those with many edges connecting them to other vertices) before those that are less constrained. While most of these techniques can be described as static, because the orderings remain unchanged during the greedy color assignment process [14], [19], a somewhat more sophisticated technique, known as *DSatur*, [10] operates dynamically. by giving priority to vertices with the most neighbors already colored. Similar one-shot ordering criteria have been successfully applied to timetabling. *Largest degree* (LD) and *largest enrolment* (LE) are popular static methods for ordering examinations, while *saturation degree* (SD) operates in a similar fashion to Desatur. Of particular note is a recent paper by H. Asmuni and E.K. Burke and J.M Garibaldi, [1], in which these three ordering criteria (LD, LE and SD) are brought together in a fuzzy system, producing very good results.

Despite their attractiveness in terms of speed and simplicity, however, one shot ordering heuristics are not always very effective in practice. In particular, with the possible exception of fuzzy systems, they are not easy to adapt to problems with multiple constraints or objectives. Nevertheless, such algorithms are extremely useful in providing upper bounds and starting points for more sophisticated methods.

The present author believes that the CL heuristics have been rather overlooked in the past. These techniques operate very differently from the one-shot heuristics, focussing their procedures on whole groups of vertices (i.e., color classes), rather than on sorting individual items. Furthermore, these techniques are iterative, and can prove very effective if used over a period of time. Of particular significance is a rare property of the CL heuristics which ensures that it is impossible to get a worse result by applying any of their reordering techniques to the graph coloring problem, and it is possible that a better result (using fewer colors) may be produced (see [8] for details). Numerical measures associated with each group or color class, such as its cardinality or its total degree sum, are used as criteria for rearranging the classes. Following each rearrangement, the greedy decoder reassigns the colors. It is at this stage that better solutions can arise, requiring less colors. Although the CL heuristics were developed for the graph coloring problem, they can be applied equally effectively to the bin packing problem, and also to basic versions of the examination timetabling problem. Culberson and Luo suggest a random mix of various reordering heuristics and call the composite algorithm *iterated greedy, IG*.

Two main stages of IG can be identified:

1) *grouping*, and
2) *reordering*.

Figure 1 illustrates some key operations from IG applied

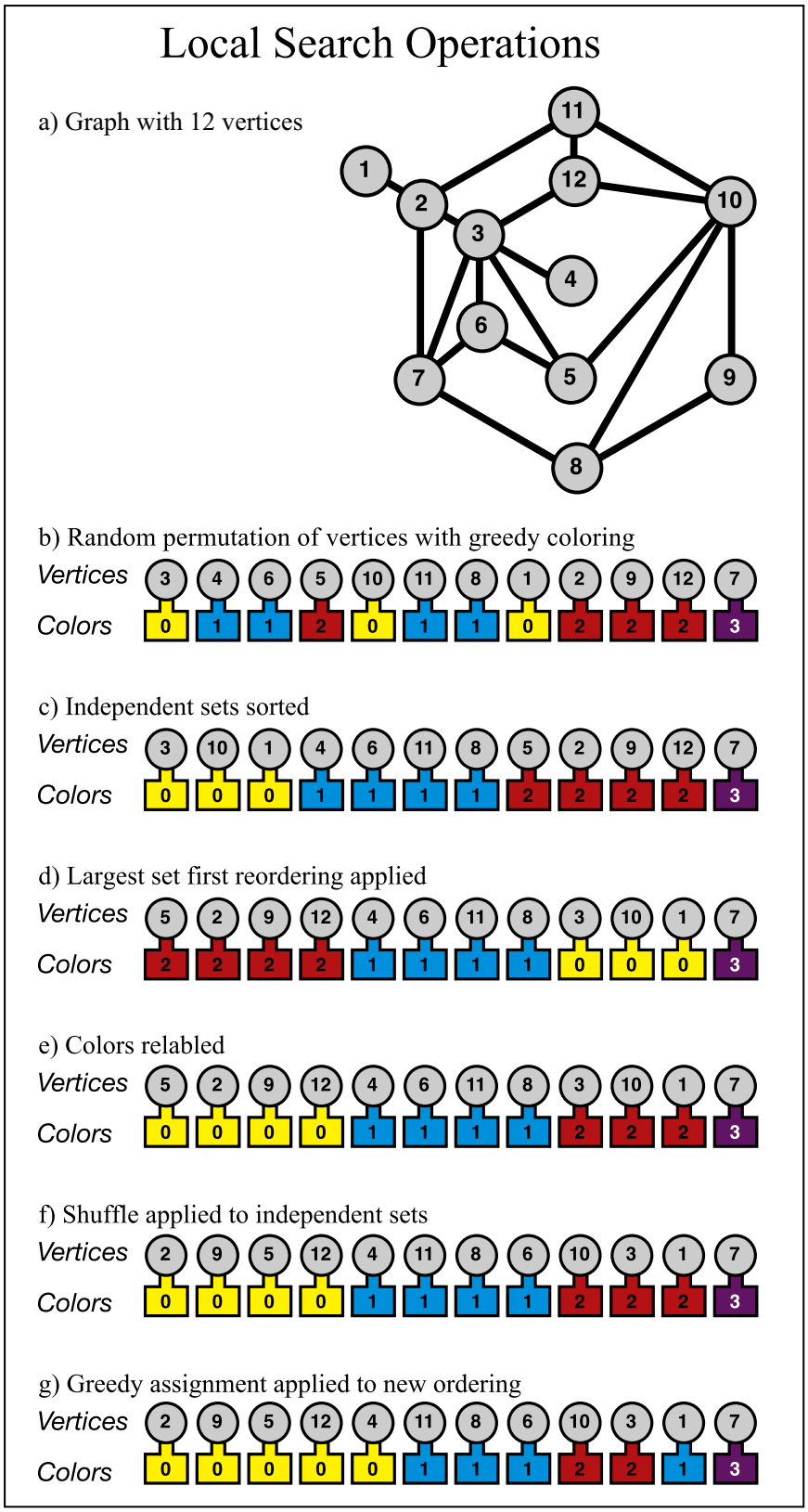

Fig. 1. Various operations by Culberson and Luo, [8], used in the local search procedure

to a small graph with 12 vertices and 14 edges. Figure 1 (b) gives a typical random permutation of the vertices from Figure 1 (a) and also the resulting greedy coloring. Figure 1 (c) shows the grouping operation used to sort the list in non-descending sequence of color label, and 1 (d) gives the arrangement following the application of one of the CL reordering heuristics called *largest first*. The largest first heuristic rearranges the color classes in non-ascending sequence of their size. Note that the positions of color classes 1 and 2 have been reversed in Figure 1 (d). This follows advice in [8] to interchange positions of equal sized color classes. In Figure 1 (f) vertices are randomly "shuffled" within (but not between) the color classes. (Note: shuffle, although mentioned, was not extensively used by Culberson and Luo in the IG algorithm. However it is included here because of its value in the present study). Finally, the greedy algorithm is applied to the new arrangement, (f), and the result is shown in Figure 1 (g). Interestingly, vertices 4 and 1 are reassigned lower color labels, leading to a reduction in the size of color class 2. Thus, given an initial permutation of vertices, the IG algorithm can be defined by the following repeating sequence:

1) greedy assignment
2) grouping of color classes
3) reordering of complete color classes
4) shuffle within each color class (optional)

Various numerical properties of the color classes were tried as criteria for reordering:

1) **Reverse**: Reverse the order of the color classes
2) **Random**: Place the color classes in random order
3) **Largest first**: Place the classes in order of decreasing size (Figure 1 d))
4) **Smallest first**: Place the smallest classes first
5) **Increasing total degree**: Place the classes in increasing order by the total degree of the group
6) **Decreasing total degree**: Place the classes in decreasing order by the total degree of the group

The favored combination of Culberson and Luo was: largest first, reverse and random used in the following ratio 50:50:30. We will use a slightly different regime, described later.

B. The Genetic Operators

The crossover used in the study is the *permutation order based crossover (POP)* introduced in [16]. POP was found to work better than the other crossover (MIS) from the earlier paper during some preliminary experiments. The operation of POP is illustrated in Figure 2.

POP was inspired by the simple one point crossover commonly applied to the "standard" bit string GA, which simply selects two parents and a cut point. The first portion of parent 1 up to the cut point becomes the first portion of offspring 2. However, the remainder of offspring 2 is obtained by copying the vertices absent from the first portion of the offspring in the same sequence as they occur in parent 2.

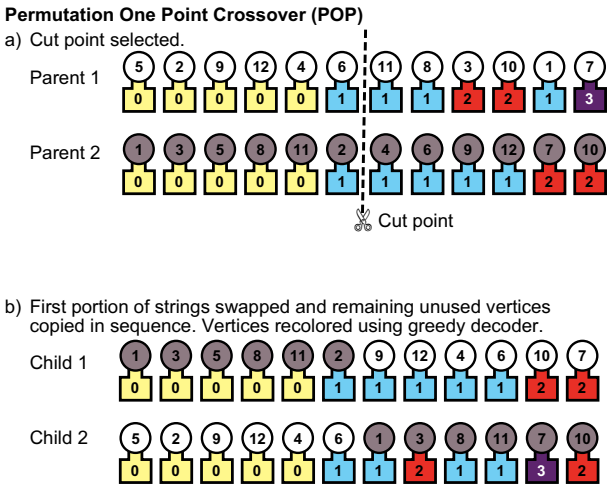

Fig. 2. POP Crossover

The mutation operator used is the *position based mutation* by Davis, [9]. This operator, also known as *insertion mutation*, simply involves selecting two values at random from a permutation list, and placing the second before the first.

C. Performance Measures/Fitness Values used by the MOEA Algorithm

The role of the MOEA is to simultaneously minimize the timetable length and the proximity costs. We shall now examine the performance measures used to measure the progress of these two objectives during the execution of the algorithm.

Minimizing the Time Slots: For many set partitioning problems the objective function (i.e., the value we are trying to optimize) is not always the best measure of progress for an optimization algorithm to use. For example, if we wish to minimize the number of colors or time slots used, respectively, for graph coloring or timetabling, enormous solution redundancy can make it difficult for an algorithm to make any progress. It is important somehow to distinguish between "good assignments" and "bad assignments", for a given number of colors or time slots. The progress measure below, P_1, was devised by Erben [11]. The goal is to maximize P_1.

$$P_1 = \frac{1}{c}\sum_{1}^{c} D_j^2 \qquad (1)$$

In Equation 1, $D_j = \sum_{i \in S_j} d_i$ represents the *total degree* for group j with d_i denoting the vertex degree of the i_{th} node, and c the total number of classes (i.e., colors or time slots). P_1 favors solutions with large numbers of highly constrained vertices concentrated in the same classes. Under this regime it appears that the members of small classes, consisting of weakly constrained vertices, tend to be gradually reassigned to the larger classes, eventually driving down the total number of classes. P_1 has the added advantage that it is insensitive to color (or time slot) labelling, unlike

TABLE I
CHARACTERISTICS OF TIMETABLING PROBLEMS

Instance	# exams	# students	# edges	# seats	GCP slots	BPP slots	UB_{av} Prox
car-f-92	543	18419	20305	2000	28	28	44
car-s-91	682	16926	29814	1550	28	37	54
kfu-s-93	461	5349	5893	1955	19	13	90
pur-s-93	2419	30032	86261	5000	30	25	71
tre-s-92	261	4362	6131	655	20	23	55
uta-s-92	622	21266	24249	2800	30	22	39

the measure devised by Culberson and Luo [8] and used in [16].

Minimizing Near Clashes: The second performance measure is based on the proximity costs, w_s, described in [6]. Cost, w_s, is imposed whenever a student has to sit two examinations scheduled s periods apart. The weights imposed are as follows: $w_1 = 16, w_2 = 8, w_3 = 4, w_4 = 2$ and $w_5 = 1$. Using these weights, cost values are evaluated for each student and all of these are then added together to give a total cost accumulated for all students. We will call this accumulated cost our *proximity cost*. For convenience, though, we will convert proximity cost, into a *proximity profit*, P_2, so that our MOEA will simultaneously maximize the two objectives, P_1 and P_2. The conversion utilizes a simple upper bound for proximity, UB, evaluated by generating the worst possible examination schedule that is possible for each individual student, and then adding together the corresponding proximity costs. A pathological schedule for an individual student would involve all exams occurring in consecutive time slots, with no gaps. P_2 is defined as:

$$P_2 = \frac{(UB - proximity)}{UB} \tag{2}$$

III. CHARACTERISTICS OF THE DATA SETS

Six data sets selected from Carter's benchmarks [6] are used in this paper. The chosen instances are the only ones to be assigned seating capacity limitations in the original data. Their characteristics are summarized in Table I. The first five columns of the table are self explanatory. Column 6 lists the best known solutions to the underlying graph coloring instances. The uta-s-92 best is taken from [5] and the purs-s-92 best from [3]. All the other graph coloring solutions can be found in [6]. Column 7 presents solutions to the underlying bin packing instances, as calculated by the present author using a simple first fit decreasing weight algorithm (FFD). This involved sorting the examinations in sequence according to the number of students taking each one, with the most popular exam listed first. FFD then placed the examinations in the earliest available time slot, complying with the seating capacity constraint, but ignoring any clashes. Interestingly, all the solutions obtained using FFD matched the so-called "ideal solutions". Ideal solutions can be evaluated by counting the total number of student_examination events, then filling up all the seats in consecutive time slots, ignoring any clashes, until all the events are used up. Thus, all the solutions in column 7 are optimal for the underlying bin packing problem. Assuming the listed graph coloring solutions are also optimal, we can say that for each instance the **larger solution** of GCP and BPP gives a lower bound for the corresponding timetabling problem.

Column 8 specifies an upper bound for the proximity cost for each instance (as explained in Section II-C). This time, however, UB_{av} is quoted as an average for each student. As explained previously, the UB measure assumes that each student has all of his/her exams in one continuous sequence. Instances with high values in column 8 (i.e., kfu-s-93 and pur-s-93) correspond to universities where, on average, students have a lot of examinations to sit. We shall use the results from columns 6, 7 and 8 to help assess solution quality for our MOEA.

IV. THE MULTI-OBJECTIVE EVOLUTIONARY FRAMEWORK

Multi-objective optimization problems are common in the real world and involve the simultaneous optimization of several (often competing) objectives. Problems such as these are characterized by optimum sets of alternative solutions, known as *Pareto sets*, rather than by a single optimum. Pareto-optimal solutions are *non-dominated solutions* in the sense that it is not possible to improve the value of any one of the objectives, in such a solution, without simultaneously degrading the quality of one or more of the other objectives in the vector. The multi-objective algorithm used here is based on ideas from the SEAMO algorithm (simple evolutionary algorithm for multi-objective optimization) algorithm) [15], [18].

In the present paper we are concerned with the simultaneous minimization of two objectives: the number of time slots in the timetable, and the overall severity of near clashes. Although these two objectives are well studied by other researchers, they are normally optimized separately, with the number of time slots being fixed prior to near clash minimization. An exception is the work by Côté *et al*, as discussed in Section I. As has already been mentioned, the order based approach used in the present study makes it easy to deal with more than one hard constraint, e.g., avoiding clashes whilst respecting the seating capacity. In contrast, Côté *et al* impose only clash avoidance using their direct representation approach. On the other hand, Burke, Bykov and Petrovic [2] express the number of students that cannot be seated as one of their nine criteria to be optimized.

The multi-objective framework, outlined in Figure 1, illustrates a simple steady-state approach, which sequentially

Algorithm 1 Simple MOEA

```
Generate N random strings {N is the population size}
Evaluate the objective vector for each string and store it
Set cr = 1 {crossover rate}
for (generation = 1; generation < totalGenerations; generations + +) do
    for all strings in the population do
        Select each string in turn, it becomes parent1
        Choose either crossover (probability = cr) or mutation (probability = 1 − cr)
        if Crossover is selected then
            Select parent2 at random
            Apply crossover to create a single offspring
            Apply mutation followed by one iteration of iterated greedy
        else if Mutation is selected then
            Apply either mutation or iterated greedy in 50:50 ratio
        Evaluate the objective vector for the offspring
        if The offspring's objective vector improves on any bestSoFar then
            It replaces a parents in the population
        else if Offspring is a duplicate then
            It dies
        else if Offspring dominates a parent then
            It replaces it in the population
        else if Offspring neither dominates nor is dominated by a parent then
            it replaces another individual that it dominates at random
        Otherwise it dies
    cr = 1 − generation/totalGenerations
```

selects every individual in the population for breeding. Once and individual is selected, crossover is applied at a rate, cr, which begins at 100 % at the start of a run, but decreases with each generation at a linear rate, finishing at 0 %. Pilot studies indicated an important role for crossover at the start of the run, but an increasing reliance on mutation later on.

Before crossover is applied, a second parent is selected at random from the population. A single offspring results from each reproduction. Following crossover, a single insertion mutation is applied to the offspring and this is followed by one iteration of the iterated greedy algorithm.

In situations where crossover is not applied, either insertion mutation or a single application of iterated greedy is applied in a 50:50 ratio.

Following the application of the genetic operators, the new individual will replace a parent, replace another individual or die, following the precise conditions stated in Algorithm 1. However, before any replacement takes place, the new individual will be preprocessed and the time slots grouped along the permutation list, as shown in Figure 1 (c), ready for the next application of the POP crossover.

V. Experimental Setup

As previously mentioned, the version of the timetabling problem addressed here combines the bin packing problem with the graph coloring problem, and adds proximity costs. The maximum number of seats per time slot corresponds to the bin packing constraint, and the avoidance of clashes to the graph coloring constraint. Given a set of students to be examined for different courses, we wish to schedule the examinations so that all clashes are avoided and the seating capacity is not exceeded in any time period. At the same time, we wish to spread the examinations so that individual students have sufficient revision time.

One potential problem with the approach used in this paper for timetabling is that, while we can guarantee that application of the CL reordering heuristics can never produce a worse solution for the underlying graph coloring or bin packing problems, in terms of the number of time slots required, it is unfortunate that a similar guarantee does not hold for the proximity costs. The time slot sequence does not matter at all, if all that is required is to avoid clashes and respect the seating capacity. On the other hand, it is precisely the sequence of examinations that determines the proximity costs. Fortunately, the grouping heuristic respects the time slot spacing perfectly, and "reverse" has a relatively small effect following any reallocation of exams to time slots, imposed by the greedy decoder. We shall explore these issues in more detail below.

The greedy decoder: To cope with the clashes and seating capacity simultaneously, the greedy decoder fits examinations sequentially into the earliest possible time slot, respecting seating capacities as well as avoiding clashes.

Adaptations to the reordering heuristics: Recall that Culberson and Luo applied the following heuristics to reorder their color classes "largest first", "reverse" and "random" in the ratio 50:50:30. For the present study the "largest first" heuristic is replaced with a heuristic that orders on *decreasing*

total degree (DTD). DTD ties in well with our objective function, P_1, which favors solutions that have classes with high values for their total degree. The "random" ordering heuristic, was also abandoned in favor of an alternative which we will call "deletion and insertion". This heuristic simply selects a time slot (color), and deletes it from one part of the chromosome, reinserting it elsewhere at random, respecting class boundaries. Deletion and insertion was found to be less disruptive to the proximity costs than randomly reordering all of the time slots.

Parameter settings: The MOEA ran for 3,000 generations on populations of 250 for each of the six instances. Five replicate runs were carried out in each case. Applications of the reordering heuristics were applied in the ratio 50:50:50 for "reverse", "deletion and insertion" and DTD. One iteration of the iterated greedy algorithm specified in Algorithm 1 corresponds to a random choice between the three reordering heuristics, with just one selection being made.

VI. RESULTS

The results for the six timetabling instances are presented in Table II. The values presented in columns 1 and 2 of the table are derived from the approximate Pareto sets produced by the 5 replicate runs carried out on each of the problem instances. Instead of quoting values for the objective functions, P_1 and P_2 used by the MOEA, however, we refer to more meaningful measures: the number of time slots (column 1) and the average proximity cost per student (column 2). Associated with each value stating the number of time slots in column 1, is a range of average proximity costs in column 2. Where a single value is listed in column 2 instead of a range, this indicates that only one of the five replicate runs produced a result for a particular number of time slots. Unfortunately, direct comparisons between the proximity costs in Table II and other published results (including [7]) is not possible, because previous researchers have not included a seating capacity constraint. Thus, to give some basis for comparison, the MOEA was re-run with the same parameters of population size and number of evaluations, but with the second objective set to a constant value. In this way random values for proximity were collected for various numbers of time slots, and these are presented in column 3 of the tables. Note: to save space results involving large numbers of time slots are omitted from the tables. However, the gradual improvement in proximity costs that can be observed in the tables, as one progresses down the rows, continues at this slow rate for longer timetables.

The results demonstrate the ability of the MOEA to produce short timetables and obtain proximity costs that are consistently better than random. Indeed, the smallest number of time slots obtained for car-s-91 (37) and tre-s-92 (23) exactly match the solutions to the underlying bin packing problems and are thus provably optimal for time slots. On the other hand, kfu-s-93 (19) matches the underlying graph coloring solution and is thus most likely optimal for time slots. Nevertheless, there is probably room for improvement with regards to proximity costs. Although results quoted by other researchers are not directly comparable to the present study because no seating capacities were imposed, their proximity values are notably better. Of particular significance is the multi-objective work of Côté *et al*, [7].

TABLE II
MOEA RESULTS

# Slots	MOEA Costs	"Random" Costs
car-f-92		
31	6.59 - 6.63	7.65 - 10.51
32	6.28 - 6.58	7.47 - 10.60
33	6.09 - 6.33	7.28 - 10.75
34	5.96 - 6.08	7.28 - 10.18
35	5.82 - 6.02	7.07 - 10.10
36	5.74 - 5.90	6.91 - 9.94
car-s-91		
37	7.23	8.26 - 9.75
38	6.77 - 7.04	8.22 - 10.58
39	6.57 - 6.83	7.83 - 10.62
40	6.47 - 6.70	7.64 - 10.19
41	6.33	7.41 - 9.95
42	6.28	7.47 - 10.00
kfu-s-93		
19	23.26 - 23.84	27.12 - 48.38
20	21.23 - 22.24	26.36 - 49.13
21	21.21 - 21.50	25.44 - 46.90
22	20.53 - 20.73	24.85 - 47.22
23	20.18 - 20.84	24.62 - 47.42
pur-s-93		
34	12.92	-
35	12.46 - 12.70	15.19 - 15.84
36	12-06 - 12.57	13.93 - 17.29
37	11.78 - 12.22	13.59 - 16.94
38	11.53 - 12.01	13.64 - 16.75
39	11.35 - 11.80	13.39 - 16.66
40	11.21 - 11.68	13.38 - 16.43
41	11.03 - 11.51	13.20 - 16.58
42	10.91 - 11.29	12.94 - 16.44
43	10.87 - 11.29	12.78 - 15.93
44	11.07 - 11.11	12.59 - 16.00
45	10.79 - 10.96	12.55 - 16.21
46	10.70 - 10.95	12.46 - 16.03
tre-s-92		
23	10.88 - 11.25	-
24	10.01 - 10.34	11.98 - 15.78
25	9.51 - 9.84	11.07 - 15.92
26	9.26 - 9.59	11.02 - 15.93
27	9.08 - 9.39	10.77 - 15.36
28	8.81 - 8.90	10.76 - 14.99
29	8.38 - 8.98	10.64 - 14.61
uta-s-92		
31	6.00	
32	5.76 - 5.90	6.79 - 8.33
33	5.52 - 5.65	6.35 - 9.58
34	5.35 - 5.45	6.13 - 9.33
35	5.17 - 5.22	6.12 - 9.21
36	4.95 - 5.09	6.03 - 9.26
37	4.81 - 5.01	5.85 - 9.32
38	4.71 - 4.99	5.76 - 9.10
39	4.64 - 4.85	5.67 - 8.85

It is interesting to observe that even the "random costs" in column 3 of Table II are well below the upper bounds for proximity costs given in the last column of Table I for all six timetabling instances.

VII. CONCLUSION AND DISCUSSION

This paper documents a preliminary study that uses an order based bi-objective evolutionary algorithm to solve the

examination timetabling problem. The order based representation used with a greedy decoder makes it easy to impose hard constraints, of which two are applied here: 1) avoiding examination clashes and 2) respecting the given maximum seating capacity. The bi-objective evolutionary algorithm uses a simple steady-state approach to simultaneously minimize the overall length of the examination period and the total proximity cost, the proximity cost being a measure of how well the examinations are spread out for individual students. Genetic operators specially devised to be sensitive to time slot boundaries, ensure that timetables of short duration are easily produced. While time slots can be sequenced in any order without altering the length of a timetable, the (random) reordering of time slots can unfortunately have a devastating effect on the values of the proximity costs. Although some effort has been made to address the issue of proximity costs in the present study, for example some modifications to the iterated greedy heuristics have been made, more work is needed. The genetic operators and greedy decoder are perhaps rather biased towards breeding short timetables, rather than good ones. Work is currently in progress to address this issue and take more account of proximity costs. Nevertheless, results presented herein clearly demonstrate that an order based MOEA approach shows promise. While single objective algorithms require the number time slots to be fixed in advance of any optimization for soft constraints, such as proximity costs, MOEAs have the advantage that a good set of non-dominated results can be produced, covering a range of timetable lengths. Furthermore, the order based approach used in the present paper ensures that only feasible timetables are produced, eliminating the need for time consuming repair heuristics which undoubtedly become an increasing burden for methods that use direct representations if problems are multiply constrained.

VIII. Acknowledgment

The author would like to thank the three anonymous referees for their helpful suggestions, and also Mark Mumford for producing the excellent diagrams in Figures 1 and 2.

References

[1] H. Asmuni, E. Burke, and J. Garibaldi. Fuzzy multiple ordering criteria for examination timetabling. In E. Burke and M. Trick, editors, *PATAT 2004*, volume 3616 of *Lecture Notes in Computer Science*, page 334354. Springer-Verlag Berlin Heidelberg, 2005.

[2] E. Burke, Y. Bykov, and S. Petrovic. A multicriteria approach to examination timetabling. In *Practice and Theory of Automated Timetabling III: Third International Conference, PATAT 2000*, volume 2079 of *Lecture Notes in Computer Science*, pages 118–131. Springer, 2001.

[3] E. Burke and J. Newell. A multi-stage evolutionary algorithm for the timetabling problem. *IEEE Transactions on Evolutionary Computation*, 3(1):63–74, 1999.

[4] E. Burke and S. Petrovic. Recent research directions in automated timetabling. *European Journal of Operational Research*, 140(2):266–280, 2002.

[5] M. Caramia, P. Dell'Olmo, and G. Italiano. New algorithms for examination timetabling. In *Algorithm Engineering 4th International Workshop*, volume 1982 of *Lecture Notes in Computer Science*, pages 230–240. Springer, 2000.

[6] M. W. Carter, G. Laporte, and S. Y. Lee. Examination timetabling: algorithms, strategies and applications. *European Journal of Operational Research*, 47:373–383, 1996.

[7] P. Côté, A. Wong, and R. Sabourin. A hybrid multi-objective evolutionary algorithm for the uncapacitated exam proximity problem. In E. Burke and M. Trick, editors, *PATAT 2004*, volume 3616 of *Lecture Notes in Computer Science*, pages 294–312. Springer-Verlag Berlin Heidelberg, 2005.

[8] J. Culberson and F. Luo. Exploring the *k*-colorable landscape with iterated greedy. In D. S. Johnson and M. A. Trick, editors, *Cliques, Coloring and Satisfiability: Second DIMACS Implementation Challenge*, volume 26 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 499–520. American Mathematical Society, 1996.

[9] L. Davis. Order-based genetic algorithms and the graph coloring problem. In *Handbook of genetic algorithms*, chapter 6, pages 72–90. Van Nostrand Reinhold, New York, 1991.

[10] D.Brélaz. New methods to color the vertices of graphs. *Communications of the ACM*, 24(4):251–256, 1979.

[11] W. Erben. A grouping genetic algorithm for graph colouring and exam timetabling. In *PATAT 2000*, volume 2079 of *Lecture Notes in Computer Science*, pages 132–156. Springer, 2001.

[12] E. Falkerauer. A hybrid grouping genetic algorithm for bin packing. *Journal of Heuristics*, 2:5–30, 1996.

[13] P. Galinier and J. K. Hao. Hybrid evolutionary algorithms for graph coloring. *Journal of Combinatorial Optimization*, 3(4):379–397, 1999.

[14] D. Matula, G. Marble, and J. Isaacson. Graph coloring algorithms. In *Graph theory and computing*, pages 104–122. Academic Press, 1972.

[15] C. L. Mumford. Simple population replacement strategies for a steady-state multi-objective evolutionary algorithm. In *Proceedings of the 2004 Genetic an Evolutionary Computation Conference (GECCO)*, pages 1389–1400, Seattle, Washington, USA, 2004.

[16] C. L. Mumford. New order-based crossovers for the graph coloring problem. In *Proc. of the 9th Int. Conference on Parallel Problem Solving from Nature, PPSN IX*, volume 4193 of *Lecture Notes in Computer Science*, pages 880–889, Reykjavik, 2006. Springer.

[17] A. Schaerf. A survey of automated timetabling. *Artificial Intelligence Review*, 13:87–127, 1999.

[18] C. L. Valenzuela. A simple evolutionary algorithm for multi-objective optimization (seamo). In *Proceedings of the 2002 IEEE Congress on Evolutionary Computation (CEC2002)*, pages 717–722, Honolulu, Hawaii, 2002. (C.L. Valenzuela is now known as C.L. Mumford).

[19] D. Welsh and M. Powell. An upper bound for the chromatic number of a graph and its application to timetabling problems. *The Computer Journal*, 10:85–86, 1967.