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

Computer Science Technical Report No. NOTTCS-TR-2006-2

Hybrid Variable Neighbourhood Approaches to University Exam Timetabling

Edmund Burke, Adam Eckersley, Barry McCollum
Sanja Petrovic and Rong Qu

First released: May 2006

Hybrid Variable Neighbourhood Approaches to University Exam Timetabling

E. K. Burke A. J. Eckersley B. McCollum S. Petrovic
R. Qu

April 6, 2006

Abstract

In this paper we propose a variable neighbourhood search (VNS) meta-heuristic [25] for the university examination timetabling problem. In addition to a basic VNS method, we introduce a variety of variants which further improve the quality of the results produced by the technique. We also aim to demonstrate that the proposed technique is capable of producing high quality solutions across a wide range of benchmark problems. This high level of generality is one of the key features of the VNS technique. In particular, its combination with a genetic algorithm which intelligently selects which neighbourhoods to use for a given problem is shown to be very successful.

1 Introduction

A very important factor which determines the success of a local search technique when applied to a given problem is the neighbourhood employed during the search. The neighbourhood is defined by a given move operator, for example all solutions which can be reached from the current solution by moving a single element (moving an exam to a new time slot in the case of examination timetable). The most basic of local search methods is the steepest descent (or ascent for maximisation problems) technique which has no method for escaping a local minimum and simply takes the steepest route down to a nearby local minimum and terminates there since no move from within the same neighbourhood can improve on the current solution. More sophisticated methodologies such as simulated annealing [1] and tabu search [18] have a mechanism to escape from local optima. Such methodologies have been very successful when applied to a wide variety of search problems including university timetabling. Survey papers detailing many of these techniques applied to exam timetabling problems include Carter and Laporte [12], Schaerf [30], Burke and Petrovic [10], Burke et al. [7] and Petrovic and Burke [27].

The issue of how much the choice of neighbourhood affects the quality of solutions is considered by Thompson & Dowsland [32, 33]. The conclusion they arrive at is that the utilisation of a more complex neighbourhood than the standard single move neighbourhood can yield significant improvements in solution quality.

In the late 1990s, Mladenović and Hansen [25] proposed the Variable Neighbourhood (VNS) meta-heuristic for solving difficult search problems. This approach changes the neighbourhood during the search. It is both very versatile and very successful compared to other local search techniques when applied to a range of different problem domains [23].

The basic VNS meta-heuristic is a descent method, moving to a new solution if and only if it is better than the current solution. Since the neighbourhoods are varied regularly, there is no need to accept worsening solutions to escape local minima, although variations of VNS do exist in which such moves are accepted (as will be discussed in section 2.2). The VNS meta-heuristic essentially samples a large number of local minima by using a local search technique to bring the solution selected from the neighbourhood to its nearest local optima. Potentially, any local search method can be used in this part of the search and the solution arrived at will of course only be a local minimum with respect to the neighbourhood used in the local search. The aim of our research is to investigate variable neighbourhood search in the context of the university timetabling problem.

1.1 Examination Timetabling

The exam timetabling problem, as encountered by universities and other teaching institutions can be thought of as the problem of assigning a given set of exams to a number of time slots subject to a set of constraints /citeCarter96,BurkeElliman. The number and variety of these constraints has increased markedly in recent years as a result of greater numbers of students and larger modularisation of courses at many universities. This in turn causes the exam timetabling problem to be not only more difficult to solve, but also potentially very different from year to year. The key elements common to almost all exam timetabling problems include a set of exams to schedule, a given arrangement of time slots (either in a fixed or variable length timetable) and a set of student exam enrolments defining the clashes between exams.

The constraints on an exam timetabling problem can be divided into two categories, known as *hard* and *soft* constraints. *Hard constraints* are defined as those which must be satisfied in order for the solution to be considered feasible. *Soft constraints* are considered to be less essential and violation of these is acceptable, but still undesirable. The violation of these soft constraints is used in the evaluation of the quality of a given feasible solution. In some cases, it is required only that a solution is feasible, but more often a number of different solutions are evaluated with the aim of finding the *best* or most suitable. Burke et al. [4], Burke & Petrovic [10] and Merlot et al. [24] discuss a variety of side constraints.

One of the key factors in the development of meta-heuristic techniques is the connectivity of the search space. This is largely defined by the neighbourhood used and can have a major impact on the quality of solutions produced. Neighbourhoods which allow complete or near-complete connectivity of the search space give potentially far greater ability for the technique to reach a high quality solution. This is especially important if the quality of the initial solution fed to the search is low. If the search space is highly disconnected as a result of using a *bad* neighbourhood, the reliance on the initial solution becomes much greater since many good solutions will not be reachable

by any technique using such a neighbourhood alone. In the case of exam timetabling, the most commonly used is the *single move* neighbourhood in which a move consists of reallocating a single exam to a new feasible time slot. This neighbourhood, when used alone, can yield a disconnected search space since exams which clash with another exam in every other time slot often cannot be moved at all.

The aim of our research is to develop a variable neighbourhood search (VNS) technique which focuses on using a variety of different neighbourhoods in the search in order to increase the generality of the technique in the sense that it works equally well on different examination timetabling problems whilst still giving high quality competitive solutions. Section 2 introduces a generic variable neighbourhood search technique and then discusses the application of the VNS technique to exam timetabling. A number of neighbourhoods are considered and some of the many variants of the basic VNS which can be used to improve results are described. Section 3 presents a combined genetic algorithm and VNS approach, VNS-GA, to exam timetabling to improve solution quality by using intelligent selection of neighbourhoods. Results and analysis for both VNS and VNS-GA are presented in Section 4 and overall conclusions are given in Section 5.

2 Variable Neighbourhood Search

The use of more than one neighbourhood within a search provides a very effective method of escaping from a local optima. Namely, it is often the case that the current solution, which is a local optimum in one neighbourhood is no longer a local optimum in a different neighbourhood and can therefore be further improved using a simple descent approach. This is the key to the success of the VNS technique presented here, with a variety of neighbourhoods ensuring not only a far greater (potentially complete) connectivity of the search space, but also enabling a simple, yet powerful search technique to yield high quality results.

The steps of the basic VNS meta-heuristic (refer to Hansen and Mladenović [25, 22] for a more detailed description) are presented in figure 1.

Any finite number, k_{max}, of pre-defined neighbourhoods may be used within VNS. The neighbourhoods are usually, in some sense, nested with k_{max} being the most diverse neighbourhood, although this is not a strict requirement. Stopping conditions may be defined as for any local search technique. Three common examples are total number of iterations, number of iterations without improvement and CPU time. In the basic VNS, the move selected from the neighbourhood at step 2 (a) is generated at random, avoiding any issues of cycling; also the local search is performed using a single neighbourhood. There are a large number of variations of the basic VNS, with changes possible to each step of the algorithm. Many of these are discussed in [22].

Our initial implementation of the VNS for exam timetabling was based on the basic one presented in figure 1, to which a number of variations were sequentially added to further improve performance.

The one major change from the basic VNS presented is that instead of continuing the search with neighbourhood N_1 each time an improvement is found in step 2 (c), the search continues using the current neighbourhood which yielded the improvement.

- *Initialisation*: Select the set of neighbourhood structures N_k, $k = 1, \ldots, k_{max}$, to be used in the search; find an initial solution x; choose stopping criteria;
- *Repeat* until stopping criteria is satisfied:
  1. Set $k := 1$;
  2. Until $k = k_{max}$, repeat:
     (a) *Shaking*: Generate a point x' at random from the k^{th} neighbourhood of x ($x' \in N_k(x)$);
     (b) *Local search*: Apply a local search method with x' as initial solution, until a local optimum, x'', is obtained;
     (c) *Move or not*: If x'' is better than the incumbent solution x then move there ($x \leftarrow x''$), and continue the search with N_1 ($k \leftarrow 1$); otherwise, set $k = k + 1$;

Figure 1: The steps of the basic VNS meta-heuristic

The main reason for this is that it places slightly less reliance upon the ordering of the neighbourhoods and focuses the search on each neighbourhood for as long as it yields an improvement before moving to the next neighbourhood in the list. Experiments with the basic VNS, always restarting with neighbourhood N_1 after an improvement have also been performed, but results so far have not been as good as with the modified one. Hence, further experiments are all performed with this modified basic VNS. This allows further neighbourhoods to be incorporated easily without the need to consider the ordering, since with this method the order only matters for determining which neighbourhood the search begins with.

Two heuristics are employed for VNS initialisation: a greedy and a randomised largest degree graph-colouring heuristic (both of which employ limited backtracking to produce a feasible initial solution). The largest degree graph-colouring heuristic orders exams on the basis of the number of clashes that they have with the other exams. In the greedy variant, the next exam in the ordering is placed in the feasible time slot which yields the least penalty. The randomised variant chooses for the next exam a random feasible time slot. The greedy approach gives a higher quality of initial solution at the expense of diversification whilst the randomised approach provides the opposite.

The two different initialisation techniques were chosen for a number of reasons. Firstly, the greedy method produces much better initial solutions with respect to the objective function and so may result in better, or at least faster solutions produced by VNS with stopping conditions based on the number of iterations since the last improvement. However, this deterministic technique, utilising a backtracking method to reassign exams if an infeasible solution is reached, may fail to find a feasible solution. On the other hand, the random method does always find feasible solutions. Also, we were interested to investigate how much the quality of the initial solution affects the quality of solution produced by VNS. Many local search techniques, can be very dependent on their initial solution, but it was felt that VNS might be able to avoid this

dependence on the initial solution by still being capable of reaching all areas of the search space using different neighbourhoods. The search space on which our VNS operates consists of feasible solutions.

The local search part of the VNS method uses a simple steepest descent approach which is fast and yields very good results. Other more complex local search techniques could be considered. However, introducing a more complex local search technique would involve a number of new parameters which have to be carefully tuned and will notably increase the running time of the algorithm. Therefore, in order to keep the number of parameters as small as possible a simple steepest descent approach was chosen.

2.1 Neighbourhoods used within VNS

In the exam timetabling problem, neighbourhoods used in local search techniques generally consist of moving some subset of exams from their current time slot to a new time slot. The number and identity of exams to move define a neighbourhood. Our initial implementation of VNS used the following eight neighbourhoods:

1. *Single move*: a neighbourhood consists of all moves obtained by selecting a single exam and moving it to a new feasible time slot. This neighbourhood is most commonly used in single-neighbourhood local search techniques. However, it can be quite limiting as many exams in a timetable have no other feasible slots to move to.

2. *Swap*: a neighbourhood contains all feasible moves involving swapping the time slots of a pair of exams, e_i and e_j. This is a very limited neighbourhood since it requires that the two exams selected can move to each other's time slot preserving the feasibility. Despite its limitations when used within a single neighbourhood technique, the Swap neighbourhood can prove very useful within a VNS framework.

3. *Move 2 exams randomly*: This neighbourhood is formed from all pairs of Single moves. Instead of picking a single exam to move to a new time slot, two exams are chosen at random and moved to new feasible time slots, independently of each other. This allows for a slightly more diverse change to the current solution than the single move neighbourhood.

4. *Move 3 exams randomly*: As above but with three exams

5. *Move 4 exams randomly*: As above but with four exams

6. *Move 5 exams randomly*: As above but with five exams

7. *Move a whole time slot*: In an exam timetable, the n time slots are ordered from 1 to n by the value of the penalty cost that they incur. Rather than moving individual exams between time slots, this neighbourhood moves an entire time slot, selected at random, to a new position in the ordering (also randomly selected), with the other periods being shuffled along accordingly. Moving entire time

slots enables exams which would otherwise be unable to move (due to clashing with exams in all other time slots) to move around the timetable, thus changing the value of the penalty function. This can be very useful, especially if VNS is seeded with a bad initial solution, since it allows for exploration of a wide area of the search space.

8. *Swap time slots*: Similar to the previous neighbourhood, instead of moving a single time slot to a new position in the ordering, this neighbourhood only affects two randomly selected time slots, simply swapping all exams in one with all exams in the other. Again, this allows for previously immobile exams to move around the timetable.

The steepest descent local search uses the single move neighbourhood. All solutions yielded by the VNS neighbourhood are taken to the nearest local minimum using this neighbourhood. Therefore, VNS solutions produced in the single move neighbourhood invariably led to the same local minimum by applying the steepest descent search. A tabu list could be used to prevent the search from dropping straight back into the same local minimum. However it was decided to replace neighbourhood 1 with the more successful Kempe chain neighbourhood used by Thompson and Dowsland in their simulated annealing technique [32].

The Kempe chain neighbourhood involves swapping a subset of exams in two distinct time slots. In our implementation, an exam e, currently in slot $T1$ and a new time slot, $T2$, are selected at random, with the exams in the two time slots forming a bi-partite graph, as in Figure 2, since only feasible solutions are allowed. The Kempe chain is defined as the connected components of this bi-partite graph from a given starting exam (the exams forming one such chain are shown in black in Figure 2). These are the exams which clash with each other (indicated by an edge between the two exams) and therefore must be switched to the other period as their clashing exams are moved. The single move neighbourhood is a subset of the Kempe chain neighbourhood consisting of all disconnected vertices (exams) in the bi-partite graph - these are the exams which can be moved across to the new period without inducing a clash.

Figure 2 shows a simple Kempe chain move involving 2 time slots, T1 and T2 each containing 4 exams. If exam 1 is selected to be moved to slot T2, the Kempe chain represented by the black circles in figure 2 is constructed resulting in exams 1, 2 & 3 moving to T2 and exams 5 & 8 moving across to T1 to maintain feasibility of the solution and the bi-partite nature of the graph defined by time slots T1 and T2, as shown in figure 3. Exam 6 is a disconnected node which could move from slot T2 to T1 in the single move sub-neighbourhood of the Kempe chain neighbourhood. All other exams have clashes (represented by edges of the graph) with exam(s) in the other time slot and so could not be moved in the single move neighbourhood. Exams 4 and 7 can be exchanged in the Kempe chain neighbourhood, the equivalent of a swap move.

The Kempe chain neighbourhood eliminates the main failing of the single move neighbourhood by enabling any exam within the timetable to be moved to a new time slot. Every pair of time slots forms a bi-partite graph meaning that there will always be a Kempe chain move defined by any exam. The largest Kempe chain move would result in every exam being exchanged between two periods. Whilst in graph colouring

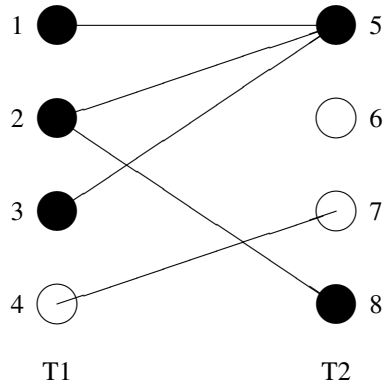

Figure 2: A Kempe chain move before execution

this would not have any effect on the solution, in exam timetabling it does since the ordering of the periods is important. However, unlike the simple move neighbourhood, the swap time slots neighbourhood is *not* wholly contained within the Kempe chain neighbourhood. For example, in Figure 2, no Kempe chain exists which would move exam 6 across to T2 together with exams 5, 7 and 8 because it has no clashes with exams in T1.

2.2 Variations of VNS for exam timetabling

One of the significant advantages of VNS is that it is a very modular technique which allows for changes in almost any of the steps given in figure 1, to produce potentially better results. A number of variations are listed below:

- *Descent-ascent*: Basic VNS is a 'descent, first improvement method with randomization' [22]. A very simple change to the algorithm would be to make it a descent-ascent method by accepting worsening moves with some probability in a similar way to that used by simulated annealing.

- *Best improvement* [22]: Instead of taking the first random move from a single neighbourhood, make a move to the best neighbourhood among all k_{max} of them.

- *Variable neighbourhood descent (VND)* [22]: VND uses many neighbourhoods during the local search phase of the VNS method as oppose to the more standard use of just a single neighbourhood. Hansen and Mladenović [22] report that this technique is crucial to obtain good results in certain problem domains.

- *Biased VNS*: In step 2 (a) of the basic VNS, it is possible to generate the solution x' by a number of different methods rather than purely at random. One such method could be to choose the best from a random selection of moves from a

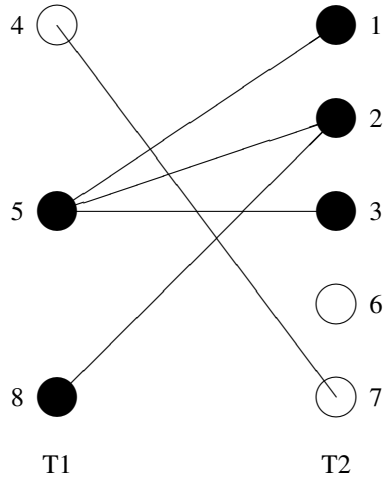

Figure 3: The result of a Kempe chain neighbourhood move

given neighbourhood. The other approach is instead to select an exam at random from the n exams adding the highest penalty to the timetable and use this exam to define a move from the Kempe chain neighbourhood.

- *More complex local search* [22]: Instead of the steepest descent local search technique, any other local search technique, such as simulated annealing, great deluge or tabu search could also be used, with an appropriate stopping condition. One of the main disadvantages of the current implementation, however, is the run time so a more complex local search technique would be likely to increase this still further.

- *Problem-specific neighbourhoods*: Since the structure of different exam timetabling problems can vary hugely, it may be the case that some neighbourhoods work far better on one problem than another, and that discarding certain neighbourhoods from the original set will not detract from the quality of solutions produced. Even more, it may in fact improve solution quality by allowing the search to spend longer in more promising neighbourhoods. In the implementation described here, the more neighbourhoods that are included, the fewer iterations are performed in each neighbourhood before the stopping condition is met. Reducing the number of neighbourhoods to the *best* subset for the specific problem may allow for further improvements to be found or for good solutions to be found faster.

- *Different initialisation strategies*: Currently, two different initialisation strategies to seed VNS are used. Section 4 presents results from VNS when initialised by a greedy and a random construction technique. One of the objectives of this comparison is to examine the importance of the initial solution to the quality of

the final solution produced by VNS. For most data sets, the greedy initialisation technique produces far better starting solutions. However, there are a number of other methods which can be used. For example, a constraint-satisfaction technique could prove to be very effective to find an initial feasible solution which VNS can then further improve, since constraint-based techniques are especially suited to finding feasible solutions without the need to consider the optimisation of an objective function.

All of the described variations are investigated in the context of exam timetabling problems. In the *biased VNS* method, two new neighbourhoods are added based on the successful Kempe chain neighbourhood. The first of these samples a random 5% of the total exams and selects the one adding the highest penalty to the current timetable. This exam is then used to form a Kempe chain move in exactly the same way as when a random exam is chosen. The second variation selects an exam at random from the 20% of exams adding the highest penalty to the timetable, and continues with the Kempe chain move as before.

The aim of adding these two neighbourhoods is to better utilise the strengths of the Kempe chain neighbourhood which has the ability to move any exam in the timetable to any other slot, unlike many simpler neighbourhoods. Selecting the first exam for the Kempe chain purely at random will quite often lead to a worse solution if the exam in question is already adding very little penalty to the timetable. It is hoped that by biasing selection toward the most troublesome exams, which are also the exams which can rarely move in the simpler neighbourhoods, better improvements can be found. Results from this variation of VNS are presented in section 4 together with those of basic VNS for comparison.

Also considered is the *descent-ascent* approach which includes an acceptance criteria combining aspects from both simulated annealing and great deluge algorithm. In order to maintain a descent method, only solutions which are less than 1% worse than the current solution are considered, with on average 10 per cent of these being accepted. This variant of the basic VNS adds in further parameters, but can yield some improvement without these parameters having been tuned to each individual problem. Further improvement may be possible with better tuning, but parameter tuning has not been introduced into the algorithm since that would take away one of the major advantages of VNS.

The *best improvement* and *variable neighbourhood descent* methods are not considered here because they involve far more significant changes to the algorithm and also increase the run time of each iteration. Neighbourhoods consisting of up to five Kempe chain moves were tested, but did not yield any successful results when used in VNS. These are considered in Section 3. Also tested were variants of VNS utilising great deluge or simulated annealing in place of the simple steepest descent local search method, but again these introduced a large number of parameters and far longer run times without yielding any improvement.

The other variation which was considered is the *problem-specific neighbourhood* selection. In the early implementation described in this section, when a new neighbourhood is identified which may add something to the performance of the algorithm it is simply added to the existing set of neighbourhoods. If the stopping condition is

extended to take account of the extra neighbourhoods, this should not detract from the ability of the VNS meta-heuristic to find high quality solutions. However, it will increase the run time of the algorithm. If the stopping condition is kept the same, adding more and more neighbourhoods may cause performance to drop due to the search spending less time in each neighbourhood. Statistics collated from many runs of VNS on 11 benchmark problems show that for some problems a certain neighbourhood very often results in an improvement whilst in other problems the same neighbourhood is rarely successful. On the contrary, a different neighbourhood is most effective.

Rather than just use these fairly crude statistics to decide which neighbourhoods to discard for each problem, the use of a genetic algorithm (GA) technique is proposed to intelligently select the best neighbourhoods to include in the search for a given problem.

3 Combining VNS with a Genetic Algorithm to improve solution quality

The main focus of VNS is on the neighbourhoods with the ability for the search to pick solutions out of a variety of neighbourhoods providing a high degree of flexibility. Section 2.1 considered initially just eight neighbourhoods with a further two added for the Biased VNS technique.

The ease with which new neighbourhoods can be introduced into the technique allows us to consider a far wider variety of neighbourhoods than the original eight defined in section 2.1. Since exam timetabling problems differ in their structure it is likely that different neighbourhoods will suit different problems. Increasing the number of neighbourhoods tested and taking note of which neighbourhoods are used most frequently when optimising different problems can give a good indication of this.

This section presents a genetic algorithm (GA) approach to intelligently selecting a subset of neighbourhoods to use within VNS for a given problem. The number of neighbourhoods is increased to 23 (Table 1) with the potential for more. A more detailed discussion of these 23 neighbourhoods is given in section 3.1. As more analysis is carried out on which neighbourhoods work best for which problems, giving a better understanding of problem structure, more problem-specific neighbourhoods can be incorporated into the algorithm. For instance, in problems were the density of clashes (graph density) is very high many exams will be unable to move to a new time slot using generic neighbourhoods. More complex neighbourhoods may be developed which allow these exams to move around more freely.

The idea of using a GA at a higher level of abstraction rather than being applied directly to the problem itself has been successfully implemented by Terashima et al. [31] to evolve the configuration of constraint satisfaction methods. Ross et al. [29] also comment that GAs may be better suited to searching for good algorithms rather than acting on the problem itself. This work is strongly connected to recent work on hyper-heuristics [5, 6] which also work at a higher level of abstraction to select from a variety of low-level techniques the best one to apply to a problem. Han et al. [15, 19, 20, 21] successfully utilise a GA within a hyper-heuristic framework to evolve an ordering of

low-level heuristics applied to the trainer scheduling problem. The key difference between the work presented here and the work of Han et al. is that in their hyper-heuristic framework, low-level heuristics are being ordered by the GA to be applied sequentially to the problem, whereas in our implementation of VNS, all neighbourhoods used within the technique are searched, but a move is only made within a given neighbourhood if it fulfils the criteria for move acceptance.

The genetic algorithm is used to evolve a subset of neighbourhoods from a large pool for use within the VNS framework. The approach is referred to hereafter as VNS-GA with further possibilities for extension also considered. A simple chromosome representation is used with fixed length equal to the total number of neighbourhoods to select from. Each neighbourhood is represented by a number, but their ordering is unimportant since the VNS method cycles through all neighbourhoods. The search moves to the next neighbourhood only when the move selected from the current neighbourhood is not accepted. If the move is accepted, the search continues in the same neighbourhood rather than returning to the first neighbourhood. Neighbourhoods can be repeated within the chromosome representation, but duplicates are removed when the chromosome is translated to the actual set of neighbourhoods to be used within VNS. The chromosome represents the set of neighbourhoods to be used within VNS. This enables creation of many possible distinct subsets of the full neighbourhood set. A chromosome in which all elements are the same would represent just that single neighbourhood supplied to VNS.

Crossover and mutation operators are both implemented in a simple manner. A percentage of the population of each generation is chosen to produce an equal number of offspring. The rest of the next generation is selected directly from the chromosomes of the current generation. In this implementation, 70 per cent of the chromosomes are selected for crossover using a roulette wheel style selection based on their fitness evaluation. Most of the 30 per cent of chromosomes to survive to the next generation are also selected using the same roulette wheel method. The probability, $P(x_i)$, of a chromosome x_i being selected from the population X_g of chromosomes in generation g is given in expression (1) with the fitness function given in expression (2)

$$P(x_i) = fitness(x_i)/(\sum_{\forall x_j \in X_g} fitness(x_j)) \tag{1}$$

$$fitness(x_i) = \max\{(\max_{\forall x_j \in X_1} \{VNS(x_j)\} \times f) - VNS(x_i), 0\} \tag{2}$$

$VNS(x_j)$ gives the solution penalty, according to the objective function, resulting from the application of the VNS algorithm with the neighbourhoods specified by chromosome x_j. If more than one VNS run per chromosome is used then this value can be either the average or best across the runs for a given chromosome. f is the fitness modifier. A lower value of f causes the better chromosomes to have a much larger chance of roulette wheel selection than the worse chromosomes, whereas a higher value gives worse chromosomes a better chance of being selected by making the difference between fitness values of the best and worse chromosomes relatively much smaller.

A standard one-point crossover technique is used to produce the two *offspring* from the two selected *parents* with the crossover point selected randomly. Figure 4 shows

an example of the crossover procedure for chromosomes of length 8 with the crossover point indicated by the dashed line. Child 1 takes the front portion of the chromosome from Parent 1 and the back portion from Parent 2 and Child 2 the opposite. Since multiple copies of a neighbourhood are permitted in this representation there is no need to perform any repair operator after the crossover. The interpretation of each chromosome as a set of neighbourhoods is given to the right of figure 4.

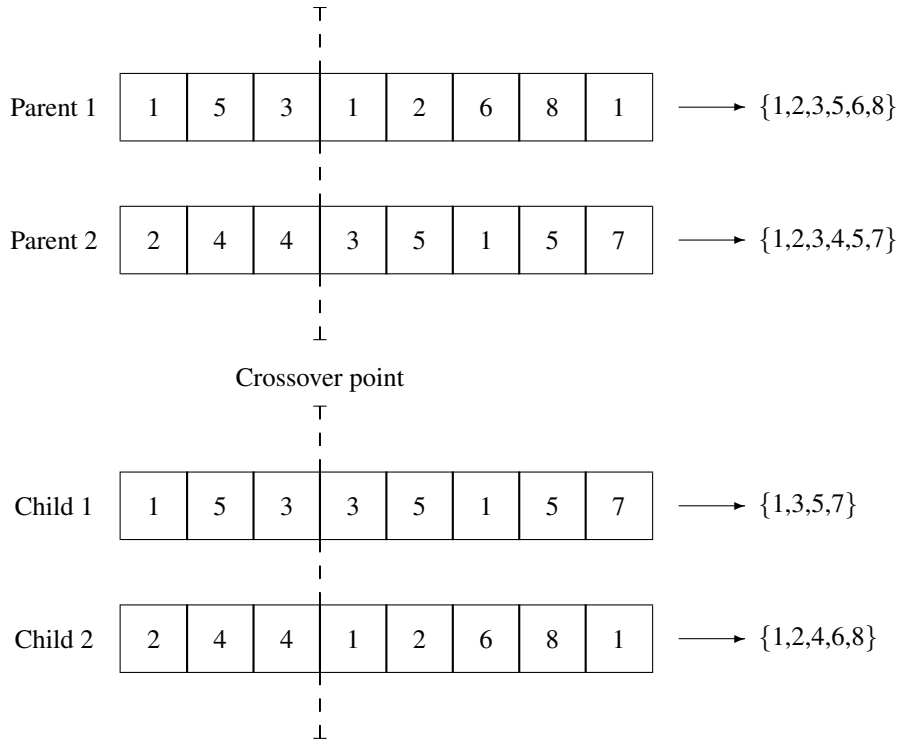

Figure 4: The one-point crossover operator for chromosomes of length 8

The mutation operator acts after selection and crossover, changing an element (gene) of a chromosome to a random neighbourhood with a given probability which can be increased or decreased to alter the random element of the evolution.

The basic steps of the VNS-GA procedure are given in Figure 5.

- *Initialisation*: Set the following parameters:
  - Total number of generations, G
  - Population size, n
  - Number of independent runs of VNS using the neighbourhoods represented by a chromosome, v
  - Number of members of the population selected as parents for crossover, s
  - Fitness modifier, f
  - Mutation rate, m

  The initial population X of chromosomes $(x_1, \ldots, x_n)$ is created, either at random or by including some selection method.

- *Repeat* for G generations:
  1. Calculate fitness of each chromosome in X using v runs of VNS
  2. Select $s/2$ pairs of chromosomes from X using roulette wheel selection and a random crossover point for each pair
  3. Perform crossover between all $s/2$ pairs of parents to produce s children, $(c_1, \ldots, c_s)$
  4. Select $n - s - 1$ chromosomes, $(c_{s+1}, \ldots, c_{n-1})$, from the current population, X, using roulette wheel selection
  5. Set c_n to be the best chromosome from population X
  6. Perform mutation on all new chromosomes, $(c_1, \ldots, c_n)$ according to the mutation rate, m
  7. Set $X = (c_1, \ldots, c_n)$

Figure 5: The steps of VNS-GA procedure

As discussed earlier, the fitness calculation in Step 1 of figure 5 can be calculated either from the average result or the best result across the runs of VNS if $v > 1$. Step 5 causes the most promising chromosome from the last population to be carried through to the new population automatically (elitist approach).

3.1 The Neighbourhoods

For experiments using the VNS-GA technique, 23 neighbourhoods were defined, from which to select a subset for each problem (Table 1). 'Type A' represents the Kempe chain move with first exam selected being the one which gives the highest penalty from a random selection of 5% of the total exams, 'Type B' represents the Kempe chain move with first exam selected at random from the 20% giving the highest penalty:

1. Single random Kempe chain move
2. Swap two exams
3. Move two exams at random
4. Make two random Kempe chain moves
5. Move three exams at random
6. Make three random Kempe chain moves
7. Move four exams at random
8. Make four random Kempe chain moves
9. Move five exams at random
10. Make five random Kempe chain moves
11. Make one Kempe chain move (Type A)
12. Make one Kempe chain move (Type B)
13. Make two Kempe chain moves (Type A)
14. Make two Kempe chain moves (Type B)
15. Make three Kempe chain moves (Type A)
16. Make three Kempe chain moves (Type B)
17. Make four Kempe chain moves (Type A)
18. Make four Kempe chain moves (Type B)
19. Make five Kempe chain moves (Type A)
20. Make five Kempe chain moves (Type B)
21. Move a whole time slot
22. Swap time slots
23. Randomly order time slots

Table 1: Neighbourhoods defined for VNS-GA

4 Results

4.1 VNS Results

A number of benchmark problems have been made publically available to test and compare exam timetabling techniques. Of these, the most widely used are those given by Carter et al. [13] for the uncapacitated exam timetabling problem. In these, the number of time slots is fixed *a priori* and the aim is to spread clashing exams around the timetable as much as possible. The objective function adds a penalty w_s for a timetable whenever a student must sit two examinations within s periods of each other, where $w_1 = 16, w_2 = 8, w_3 = 4, w_4 = 2, w_5 = 1$. The key features of these problems are given in Table 2. Graph density is calculated as the ratio of the number of exams that are in conflict with each other (i.e. have students in common) to the square of the total number of exams. Results for these problems are reported as average penalty per student. Results from the literature for a variety of methods applied to these data sets are presented in Table 3. As can be seen from these results, a wide variety of different techniques have been successful on these problems with no single technique outperforming all others across the whole range of problems. Techniques which are successful on one problem are often far less successful on other problems when compared against different techniques.

Data Set	No. of exams	No. of students	No. of enrolments	Graph Density	No. of periods
CAR-S-91	682	16925	56877	0.13	35
CAR-F-92	543	18419	55522	0.14	32
EAR-F-83	190	1125	8109	0.27	24
HEC-S-92	81	2823	10632	0.42	18
KFU-S-93	461	5349	25113	0.06	20
LSE-F-91	381	2726	10918	0.06	18
STA-F-83	139	611	5751	0.14	13
TRE-S-92	261	4360	14901	0.18	23
UTA-S-92	622	21267	58979	0.13	35
UTE-S-92	184	2750	11793	0.08	10
YOR-F-83	181	941	6034	0.29	21

Table 2: Characteristics of uncapacitated benchmark problems [13]

Table 4 compares the results produced by the Basic VNS algorithm with the best reported solution taken from Table 3. Figures in italics represent the best solution found between the two initialisation techniques, VNS-RI using the randomised initialisation heuristic, VNS-GI using the greedy version. Basic VNS uses neighbourhoods 1-8 from Table 1, all using random move selection in step 2 (a) of the algorithm in Figure 1.

Table 5 gives the results of the Biased VNS approach when initialised by the two different methods. Biased VNS-RI and Biased VNS-GI uses neighbourhoods 1-8 from Figure 1 with two additional biased Kempe Chain neighbourhoods, as described in section 2.2, one selecting the exam currently adding the largest penalty to the timetable

Data Set	Carter et al. (1996) [13]	Caramia et al. (2001) [11]	Burke & Newall (2003) [8]	Di Gaspero (2002) [16]	Casey & Thompson (2003) [14]	Merlot et al. (2003) [24]
CAR-S-91	7.1	6.6	**4.6**	5.7	5.4	5.1
CAR-F-92	6.2	6.0	**4.0**	-	4.4	4.3
EAR-F-83	36.4	**29.3**	36.1	39.4	34.8	35.1
HEC-S-92	10.8	**9.2**	11.3	10.9	10.8	10.6
KFU-S-93	14.0	13.8	13.7	-	14.1	**13.5**
LSE-F-91	10.5	**9.6**	10.6	12.6	14.7	10.5
STA-F-83	161.5	158.2	168.3	157.4	**134.9**	157.3
TRE-S-92	9.6	9.4	8.2	-	8.7	8.4
UTA-S-92	3.5	3.5	**3.2**	4.1	-	3.5
UTE-S-92	25.8	24.4	25.5	-	25.4	25.1
YOR-F-83	41.7	36.2	36.8	39.7	37.5	37.4

Table 3: Selected results from the literature on uncapacitated benchmark problems from [13] (best results given)

Data Set	Petrovic et al. (2002) [28]	Abdullah et al. (2004) [2]	Burke & Newall (2004) [9]	Di Gaspero & Schaerf (2001) [17]	Burke et al. (2004) [3]	Paquete & Stützle (2002) [26]
CAR-S-91	-	5.2	5.0	6.2	4.8	-
CAR-F-92	-	4.4	4.3	5.2	4.2	-
EAR-F-83	34.5	34.9	36.2	45.7	35.0	38.9
HEC-S-92	10.9	10.3	11.6	12.4	10.6	11.2
KFU-S-93	14.8	**13.5**	15.0	18.0	13.7	16.5
LSE-F-91	10.6	10.2	11.0	15.5	10.4	13.2
STA-F-83	159.9	150.3	161.9	160.8	159.1	158.1
TRE-S-92	**8.0**	8.1	8.4	10.0	8.3	9.3
UTA-S-92	-	3.6	3.4	4.2	3.4	-
UTE-S-92	-	**24.2**	27.4	29.0	25.7	27.8
YOR-F-83	36.7	**36.1**	40.8	41.0	36.7	38.9

Table 3: (cont.) Selected results from the literature on uncapacitated benchmark problems from [13] (best results given)

Data Set	Best reported solution	Basic VNS-RI		Basic VNS-GI	
		Best	Average	Best	Average
CAR-S-91	4.6	5.10	5.29	*5.07*	5.24
CAR-F-92	4.0	4.20	4.39	*4.17*	4.30
EAR-F-83	29.3	*33.56*	36.33	33.70	36.35
HEC-S-92	9.2	*10.41*	11.08	-	-
KFU-S-93	13.5	*13.72*	14.40	13.85	14.54
LSE-F-91	9.6	*11.13*	11.70	11.18	11.74
STA-F-83	134.9	*156.86*	157.12	*156.86*	157.15
TRE-S-92	8.0	*8.48*	8.88	8.49	8.84
UTA-S-92	3.2	3.49	3.59	*3.40*	3.49
UTE-S-92	24.2	*25.10*	25.94	25.18	26.01
YOR-F-83	36.1	36.80	38.70	*36.77*	38.47

Table 4: Results from the basic VNS meta-heuristic with random and greedy initialisations

from a random 5% sample, the other selects a random exam from the 20% of exams adding the largest penalty. Again, the best result reported from Table 3 is included for comparison.

Average and best results from 100 runs on a 750MHz Athlon are presented for both Basic and Biased variants. The stopping condition is 2,500 iterations without improvement after a minimum of 10,000 and the random move selection from neighbourhoods causes run times to vary considerably between problems as well as across the 100 runs on a single problem. On the smaller datasets, the algorithm generally terminates in the order of 1-2 minutes, whereas run times for larger data sets range from 30 minutes to 90 minutes. Run times for results reported from the literature in Table 3 vary between around 0.5 seconds for smaller datasets up to 15 minutes on the larger datasets.

From Table 4, it can be seen that the basic implementation of VNS fails to match the best reported solutions from the literature on the 11 benchmark problems, but does still provide high quality results across all problems. When compared with other techniques from Table 3, the results from Basic VNS are highly competitive and beat the results of each other technique on at least one problem, indicating that whilst it is unable to produce any best known solutions, it is highly consistent across the range of problems. Methods from Table 3 which produce the best solution on one problem are often outperformed by a few techniques on other data sets, showing that they are much more suited to some problems than others.

Comparing the results of Tables 4 & 5 shows the clear improvement in solution quality of introducing the two biased neighbourhoods. Apart from the anomalous STA-F-83 problem, Biased VNS outperforms Basic VNS for all problems and significantly also manages to produce a best known solution to the KFU-S-93 data set of 13.38 compared to the previous best of 13.5.

When comparing the two initialisation strategies, results are mixed with neither initialisation approach outperforming the other across all problems. For the Biased VNS method, the random initialisation outperformed greedy initialisation on five problems,

Data Set	Best reported solution	Biased VNS-RI		Biased VNS-GI	
		Best	Average	Best	Average
CAR-S-91	4.6	*5.02*	5.28	5.07	5.12
CAR-F-92	4.0	4.17	4.34	*4.12*	4.23
EAR-F-83	29.3	*33.10*	36.00	33.46	35.78
HEC-S-92	9.2	*10.26*	11.02	-	-
KFU-S-93	13.5	***13.38***	13.87	***13.38***	14.03
LSE-F-91	9.6	*10.66*	11.33	10.93	11.58
STA-F-83	134.9	*156.86*	157.04	*156.86*	157.06
TRE-S-92	8.0	*8.35*	8.76	8.39	8.77
UTA-S-92	3.2	3.47	3.55	*3.39*	3.50
UTE-S-92	24.2	*24.86*	25.41	*24.86*	25.43
YOR-F-83	36.1	36.48	38.33	*36.43*	38.03

Table 5: Results from the VNS with biased neighbourhoods meta-heuristic with random and greedy initialisations

whilst leading to inferior results on three problems and equal performance on the other three. This in itself is quite significant however, implying that VNS has the capability to overcome a seemingly bad initialisation to still produce equally high quality results. Unsurprisingly, the average performance of the VNS initialised with the greedy technique was higher in a majority of cases, but not by a significant amount, and not on all problems. Again, this shows that the VNS technique can take a range of highly diverse initial solutions (for certain problems the random technique yields some initial solutions which have twice the cost of others) and lead them to high quality solutions.

Further research is required to determine exactly how much the initial solution is changed to produce the final solution using the two initialisation techniques. It may be the case that the increased diversity of the random initialisation approach allows VNS to more easily find high quality solutions. The less diverse greedy approach may start the search of in a relatively bad region by placing certain exams in unfavourable time slots (whilst still producing a solution with an overall lower penalty) which then have to be corrected by the VNS algorithm.

Also worthy of note is that Biased VNS invariably leads to a higher level of consistency in the quality of solutions produced with the best average performance (for 10 of the 11 problems). The average result from Biased VNS on some data sets also outperforms the best result of some techniques from Table 3, showing that the method is capable of producing good results consistently across a number of runs as well as individual good results from a selection.

Table 6 presents the best results obtained by the descent-ascent variation of the Biased VNS approach [1] compared to those of selected other techniques in the literature. On the majority of datasets, this variation outperforms the pure descent Biased VNS variant with only the EAR-F-83 data set proving to be an exception. However, further analysis of the individual runs shows that the 33.10 result obtained by the Biased VNS-RI (Table 5) technique was an anomalous result from the 100 runs with the second best

[1] These results are the best across both initialisation techniques

Data Set	Abdullah et al. [2]	Caramia et al. [11]	Burke & Newall [8]	Casey & Thompson [14]	Merlot et al. [24]	Descent-ascent Biased VNS
CAR-S-91	5.2	6.6	**4.6**	5.4	5.1	4.9
CAR-F-92	4.4	6.0	**4.0**	4.4	4.3	4.1
EAR-F-83	34.9	**29.3**	36.1	34.8	35.1	33.2
HEC-S-92	10.3	**9.2**	11.3	10.8	10.6	10.3
KFU-S-93	13.5	13.8	13.7	14.1	13.5	**13.2**
LSE-F-91	10.2	**9.6**	10.6	14.7	10.5	10.4
STA-F-83	150.3	158.2	168.3	**134.9**	157.3	156.9
TRE-S-92	8.1	9.4	**8.2**	8.7	8.4	8.3
UTA-S-92	3.6	3.5	**3.2**	-	3.5	3.3
UTE-S-92	**24.2**	24.4	25.5	25.4	25.1	24.9
YOR-F-83	**36.1**	36.2	36.8	37.5	37.4	36.3
Total Penalty	**300.8**	306.2	322.3	-	310.8	305.8

Table 6: Ascent-descent Biased VNS compared to results from the literature (best results given)

being only 33.88. In a method such as VNS, which involves a large random element, there is always the possibility of any variant producing a very high quality solution on a single run, but not consistently across many runs.

It can be seen from Table 6 that the descent-ascent Biased VNS method performs very favourably compared to many current state of the art techniques, improving the best known solution for the KFU-S-93 data set to 13.2. Perhaps more significantly is the consistency of performance across the range of all 11 problems. The approach is ranked 2nd or 3rd out of the 6 presented on the other 10 data sets, giving a total penalty across all data sets second best among four approaches whose techniques were applied to all data sets. This can be misleading as it may be biased heavily by one problem, especially since the penalty for the STA-F-83 dataset accounts for as much of the total as the other 10 datasets combined. This can be overcome by normalising the penalties for each problem. Indeed, our technique compares favourably to that of Abdullah et al. when measured across only those 10 datasets. From this there are clear indications that VNS is very capable of producing high quality results across all data sets tested upon.

4.2 VNS-GA Results

VNS-GA was tested on the same 11 benchmark problems with certain parameters varied between data sets. These parameters were those which affect the time taken by the technique since our aim was to run VNS-GA on all problems for a similar amount of time rather than for the exact same number of runs of VNS. The total number of neighbourhood combinations evaluated [2] by running VNS at step 1 of figure 5 is given by $G \times n \times v$ with v determining how many times the same neighbourhood set is tested for a given chromosome.

[2]including multiple evaluations of the same neighbourhood set

In order to run all problems for roughly the same amount of time, an approximation of the time taken for a single VNS run to give a value for $G \times n \times v$ (for the given amount of time) was calculated. This was approximate as run times can vary greatly for a single problem, but it was not important that all problems had exactly the same time since their results are not being compared against each other. For the experiments reported in this paper, v was set to 1 with the exception of the HEC-S-92 and UTE-S-92 data sets whose fast run times allowed us to obtain results using $v = 3$. G and n were either balanced equally or in a 2:1 or 1:2 ratio. Mutation rates of 0.002 (0.2%) and 0.01 (1%) were both tested with fitness multiplier, f, set at either 1.01 or 1.05.

The main aim of this work is to discover the potential performance of the VNS algorithm under favourable conditions (selected neighbourhoods) rather than to evaluate the GA itself. For this reason, it was deliberately decided not to carefully tune the many GA parameters for our experiments. Instead, a few sets of values which looked to give a balanced setup to run the experiments were selected. If the method is to be used by exam timetablers who may not be experts in genetic algorithms, it is important to examine the performance with a set of default parameters rather than with carefully tuning ones. The experiments reported here are based on a variety of different parameters as described above, but these were not *tuned*, just selected in advance with no knowledge of how any would perform. The version of VNS used is the descent-ascent Biased VNS with the largest degree heuristic with randomisation initialisation (Biased VNS-RI) as this proved to be the best combination overall from Section 4. The greedy initialisation heuristic was not used here as it is less reliable at finding feasible initial solutions and gives a lower diversity of solutions.

The focus is on the performance of the VNS technique with different neighbourhoods in order to show that results can be improved by selecting subsets of the large set of neighbourhoods which may be more suitable to a problem so that more time can be spent searching these neighbourhoods rather than those which are not contributing. From the results obtained in these experiments, it should be possible to determine how effective this technique is so that the GA itself can then be more carefully considered to improve performance even further.

Table 7 gives the best results found by the VNS-GA algorithm on each data set and gives a comparison with the previous best result from the VNS variations presented in section 4 and the best known solutions reported in Table 3. The neighbourhoods which were used to produce the best solutions are also given. It can easily be seen that the VNS-GA method improves on the performance of the descent-ascent Biased VNS with the 10 neighbourhoods used previously. Experiments were also carried out using the full set of 23 neighbourhoods with VNS for all problems. In all cases these results were at least as good as the previous best results with the 10 neighbourhoods, but were not as good as the results from the VNS-GA. This indicates that some (or all) of the additional 13 neighbourhoods are adding something useful to the ability of VNS, but that selecting a subset of these 23 to focus on gives still better results.

In order to test whether the neighbourhoods which provided the best solutions given in Table 7 are consistently better than using all 23 neighbourhoods or whether they just happened to be the neighbourhoods which were being used in VNS while the random element of the algorithm output that best result, a further series of tests were undertaken. In the VNS-GA, a set of neighbourhoods was only tested for use within VNS

Data Set	Best Reported Solution	Best VNS Solution	Best VNS-GA Solution	Neighbourhood Subset For Best Solution
CAR-S-91	**4.6**	4.9	**4.6**	{1,4-8,11-13,16,17,19-23}
CAR-F-92	4.0	4.1	**3.9**	{1,3-6,8-11,13-17,19-23}
EAR-F-83	**29.3**	33.1	32.8	{1,3,4,7,11,13-15,17,21-23}
HEC-S-92	**9.2**	10.3	10.0	{1-4,6,8,10-12,14,16,17,19-22}
KFU-S-93	13.5	13.2	**13.0**	{2,4,6,8-10,12-15,17-20,22}
LSE-F-91	**9.6**	10.4	10.0	{2,3,5-8,10,13,15-17,19,20,22,23}
STA-F-83	**134.9**	156.9	156.9	Many
TRE-S-92	8.0	8.3	**7.9**	{2,4,7-12,15,19,21-23}
UTA-S-92	**3.2**	3.3	**3.2**	{1-9,13,18-22}
UTE-S-92	**24.2**	24.9	24.8	{1-3,5-10,13-17,19,20,22,23}
YOR-F-83	36.1	36.3	**34.9**	{1,5,6,9,10,12-14,16,17,19,21,22}

Table 7: Best results obtained from the VNS-GA algorithm with neighbourhoods given

Data Set	VNS with all neighbourhoods			VNS with selected neighbourhoods		
	Best	Average	Time (s)	Best	Average	Time (s)
CAR-S-91	4.7	**4.9**	2751	4.6	**4.9**	3084
CAR-F-92	4.0	4.2	1605	3.9	**4.1**	1686
EAR-F-83	32.9	34.2	175	32.8	**34.1**	162
HEC-S-92	10.2	**10.6**	28	10.0	**10.6**	28
KFU-S-93	13.2	13.6	633	13.0	**13.4**	673
LSE-F-91	10.1	**10.6**	359	10.0	10.8	345
TRE-S-92	8.3	8.4	244	7.9	**8.2**	218
UTA-S-92	3.3	**3.4**	2358	3.2	**3.4**	2040
UTE-S-92	24.9	25.1	67	24.8	**25.0**	73
YOR-F-83	35.2	**36.4**	124	34.9	36.6	126

Table 8: Results from VNS comparing all neighbourhoods with the 'best' subset of neighbourhoods

a single time ($v = 1$), except for the two problems for which $v = 3$ was also tested. Table 8 shows the results of running VNS 100 times (v = 100) with the neighbourhood sets suggested in Table 7. Results are compared to those produced by 100 runs of VNS with the full set of 23 neighbourhoods. Average run times for the technique are also given in both cases with experiments carried out on a P4 1.8 GHz Athlon PC.

Results from table 8 are fairly inconclusive with regard to the merits of using selected neighbourhoods rather than just using all 23. Using selected neighbourhoods, the average result is better in five of the problems whilst using all 23 neighbourhoods provides a better average for two of the problems. In all cases the difference between the two sets of results is small. The STA-F-83 data set was excluded from further experiments because the best solution found by VNS is always 156.86 irrespective of the technique or neighbourhood selection and this solution is found regularly across 100 runs. Further analysis would be needed to investigate the reasons for this strange behaviour, but for the purposes of this research, VNS produces a competitive result on

this data set with any selection of neighbourhoods.

4.2.1 Notes on Results

There are a number of points to note from the results presented in tables 7 & 8 together with the raw data produced by the experiments. It is clear that the VNS-GA is capable of producing the best results of all the VNS variants tested on all problems, although on many problems the difference between this method and supplying the VNS algorithm with all 23 neighbourhoods is relatively small. Also the use of selected neighbourhoods with the same stopping conditions, based on the number of iterations without improvement, gives no advantage in terms of the time taken by the algorithm to find solutions. However, it is the case that the selected neighbourhoods presented in table 7 enabled VNS to obtain the best results whereas no other combination of neighbourhoods was able to produce results of that quality.

A major aim of this work was to show that VNS can provide highly competitive results and the results shown demonstrate that this is the case, irrespective of whether the GA was successful at selecting neighbourhoods or not. Having shown that VNS is capable of such high quality results, the next step is to give VNS the best chance of repeating that quality of result on a consistent basis. Comparing the average results from Table 8 with the best results reported in the literature (Table 3) confirms that even the average performance over 100 runs of VNS can outperform the best results of a number of specially designed meta-heuristic techniques.

The disadvantage of VNS is that the run time to obtain these results is notably longer than that of other meta-heuristic techniques. However, analysis of the individual runs of the algorithm shows that the vast majority of improvement takes place very quickly, the remainder of the time is then spent making relatively small improvements. Therefore, in common with many local search techniques, there is a trade-off between run time and solution quality. High quality solutions can be obtained in a quarter of the time taken for the results given, but for best performance the longer run time is required.

As regards the effectiveness of the GA for selecting neighbourhoods, clearly more research needs to be carried out to determine whether the method is successful at evolving the 'best' subset(s) of neighbourhoods for a given problem, but results from the relatively untuned GA presented here give a lot of promise. One obvious drawback of the method compared to a more conventional GA is that the fitness function is calculated using VNS which involves a very high random element meaning that the same chromosome will be evaluated with a different fitness every time, unlike a standard GA fitness function which returns the same fitness for identical chromosomes. This can be significantly improved by increasing the value of v to 10 or more and using the average VNS result across the v runs to calculate chromosome fitness from. This would also allow the GA to *evolve* the neighbourhoods more obviously than it does currently by improving the consistency.

In the current GA implementation, with $v = 1$, the total fitness of each successive generation is very varied for most problems with only the larger data sets tested showing an obvious initial evolution. This is largely because of this variable fitness calculation causing a chromosome which may have been given a high fitness in one generation

to have a relatively low fitness in the next generation. The higher the value of v, the more similar the fitness evaluation will be for two identical chromosomes meaning that when the fitter chromosomes are selected they represent sets of neighbourhoods which consistently give high quality results rather than just being able to produce a single good result. Further tuning of the other parameters of the GA, especially the population size, fitness multiplier and the mutation rate could also significantly improve the consistency of its performance. Combined with increasing the number of VNS runs per chromosome, v, tuning the fitness multiplier will give a much more consistent selection of the best chromosomes resulting in a convergence of the GA to a set of neighbourhoods which should be capable of giving highly consistent performance both in terms of best and average results.

The implementation described in this paper contains just 23 neighbourhoods, many of which are very similar in terms of what they do. It is possible however to implement a huge variety of far more complex neighbourhoods which can easily be added into VNS to give a much larger pool of neighbourhoods. Some of these could be specifically designed with particular problems in mind. The more that is known about the structure of a given problem, the easier it may be to develop a neighbourhood specifically for that problem which takes into account its main features and how they affect the search process. For a technique to be used by people with expertise in exam timetabling rather than in optimisation techniques, this can be a significant advantage. Whilst many methods require a high level of knowledge of the actual optimisation technique in order to make improvements, new neighbourhoods for VNS can be developed using domain knowledge instead.

5 Conclusions

The Variable Neighbourhood Search (VNS) approach presented in this paper has been shown to produce solutions of a high quality across a range of benchmark data sets, producing a best reported result on one medium sized data set and performing consistently on most other data sets using the Biased VNS method with 10 neighbourhoods.

A Biased VNS variation was developed and tested against the Basic VNS. Results from this Biased VNS have shown significant improvement over the Basic VNS on many of the data sets tested and are very competitive with current state of the art techniques. Further improvements were yielded by the addition of an ascent mechanism similar to that of simulated annealing. The best results from this approach compare very favourably with techniques reported in the literature and it proves robustly competitive across all data sets.

One significant advantage of the Basic VNS approach is that it involves very few parameters apart from the selection of the neighbourhoods. All of these are easily implemented since only a random move is required rather than an exhaustive search of the neighbourhood. With a high degree of modularisation, neighbourhoods can be added and taken away easily, any local search technique can be used and the method of move acceptance altered. Basic VNS also has a large number of potential improvements, which, whilst adding more parameters can be used to improve performance. The one notable disadvantage to the VNS implementation described here (so far) is the time

taken on large problems which can be as much as 90 minutes for a single run. Whilst run time is less crucial for exam timetabling than many other combinatorial optimisation problems, since exams are generally only taken once or twice a year with a fair degree of planning time, it is still an area which needs improvement.

Another variant of VNS was proposed, utilising a genetic algorithm to intelligently select the neighbourhoods for use within VNS from an increased selection. A set of 23 neighbourhoods was considered, but this can be increased by adding many more diverse neighbourhoods which may be much more suited to certain problems than others. The neighbourhoods so far considered are fairly general rather than being targeted at specific problem features. As a result of this and the GA not having been carefully tuned (a deliberate effect), the results of the VNS-GA compared to VNS applied with all 23 neighbourhoods are not as impressive as they perhaps could be, but still yield very high quality results. The ability of the technique to easily incorporate a wide variety of differing neighbourhoods allows it to successfully fulfil our aim of increasing the generality of a technique to be successfully applied to exam timetabling problems.

Best known solutions compared to those published at the time of writing have been found to four benchmark data sets with solutions equal to the best known found for two more data sets. Results on a further five problems are also highly competitive with state of the art techniques, indicating that the proposed VNS-GA method is capable of producing high quality solutions, even better than those achieved with the Biased VNS, under the right conditions. Those conditions involve the evolution of a set of neighbourhoods which can provide better solutions than simply including all implemented neighbourhoods within VNS.

This high level of generality is one of the key aspects of our research. Many different meta-heuristics have been applied to the benchmark data sets used within this paper, but most struggle to produce high quality solutions across the whole range of problems, when compared against each other. The variants of VNS proposed in this paper have succeeded in producing high quality results across a wider range of problems than many of these previous techniques. With a large number of further variants and improvements it is quite likely that these results could be improved still further. The inclusion of more complex and problem-structure specific neighbourhoods in the VNS-GA technique should also help to further increase the generality of the method. For those problems on which these neighbourhoods provide no improvement, the GA will remove them from the subset of neighbourhoods used within the VNS.

A number of methods for improving the performance of the GA with particular focus on its consistency have been considered. Most notable amongst these is to run VNS 10 or more times for each chromosome and to take an average result for the fitness function rather than a single run. This should result in a much better evolution of neighbourhoods, but at the cost of significantly increased running time. Combining the VNS-GA with a case based reasoning system to store the best set of neighbourhoods for previously solved problems which can be used with VNS for new similar problems could help to alleviate the problem of running time. This would be done by allowing the GA to evolve the best sets of neighbourhoods for problems within a case base. When a new problem is required to be solved, the most similar problem from the case base will be matched and the set of neighbourhoods used for that problem retrieved to be used for the new problem.

6 Future Work

Due to the modular neighbourhood structure of VNS, the technique can be adapted more easily than many single neighbourhood techniques and be applied to exam timetabling problems with more complex side constraints. New neighbourhoods can be added and tested easily, which are specifically designed for the side constraints encountered. For instance, if time window constraints are included, new neighbourhoods can be implemented which specifically target exams which are scheduled outside of their time window in order to satisfy that constraint. Further work is required to fully investigate the application of VNS to problems with a wider range of side constraints, but results presented in this paper indicate that the use of a number of different neighbourhoods in the search can yield very high quality results across a broad range of problems.

With the increasing complexity of exam timetabling problems in many universities, we believe that the proposed VNS techniques can be adapted easily to take into account any new side constraints without taking away from the generality of the method.

References

[1] E. Aarts, J. Korst and W. Michiels. Simulated Annealing. *In*: E. K. Burke and G. Kendall (eds). *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Methodologies*, Chapter 7. Springer 2005.

[2] S. Abdullah, S. Ahmadi, E. K. Burke and M. Dror. Investigating Ahuja-Orlin's Large Neighbourhood Search Approach for Examination Timetabling. *University of Nottingham Technical Report* NOTTCS-TR-2004-8.

[3] E.K. Burke, Y. Bykov, J. P. Newall and S. Petrovic. A Time-predefined local search approach to exam timetabling problems. *IIE transactions on operations engineering*, 2004, 36(6), pp 509-528.

[4] E. K. Burke, D. G. Elliman, P. H. Ford and R. F. Weare. Examination timetabling in British universities: a survey. *In*: E. K. Burke and P. Ross (eds). *Practice and theory of automated timetabling: Selected papers from the first international conference.* Volume 1153 of Lecture notes in computer science. Springer-Verlag, Berlin, Heidelberg, 1996, pp 76-90.

[5] E. K. Burke, G. Kendall and E. Soubeiga. A Tabu Search Hyper-Heuristic for Timetabling and Rostering. *Journal of Heuristics*, 2003, 9(6), pp 451-470.

[6] E. K. Burke, E. Hart, G. Kendall, J. Newall, P. Ross and S. Schulenburg. Hyper-heuristics: An Emerging Direction in Modern Search Technology. *In*: F. Glover and G. Kochenberger (eds). *Handbook of Meta-heuristics*, Chapter 16, Kluwer 2003.

[7] E. K. Burke, K. Jackson, J. H. Kingston and R. Weare. Automated university timetabling: The state of the art. *The computer journal*, 1997, 40(9), pp 565-571.

[8] E. K. Burke and J. P. Newall. Enhancing timetable solutions with local search methods. *In*: E. K. Burke and P. De Causmaecker (eds). *Practice and theory of automated timetabling: Selected papers from the fourth international conference.* Volume 2740 of Lecture notes in computer science. Springer-Verlag, Berlin, Heidelberg, 2003, pp 195-206.

[9] E. K. Burke and J. P. Newall. Solving examination timetabling problems through adaptation of heuristic orderings. *Annals of operations research*, 2004, 129, pp 107-134.

[10] E. K. Burke and S. Petrovic. Recent research directions in automated timetabling. *European journal of operational research (EJOR)*, 2002, 140(2), pp 266-280.

[11] M. Caramia, P. Dell'Olmo and G. F. Italiano. New algorithms for examination timetabling. *In*: S. Näher and D. Wagner (eds): *Algorithm Engineering 4th International Workshop, Proceedings WAE 2000 (Saarbrücken, Germany)*. Volume 1982 of Lecture notes in computer science, Springer-Verlag, Berlin, Heidelberg, New York, 2001, pp 230-241.

[12] M. W. Carter and G. Laporte. Recent developments in practical examination timetabling. *In*: E. K. Burke and P. Ross (eds). *Practice and theory of automated timetabling: Selected papers from the first international conference.* Volume 1153 of Lecture notes in computer science. Springer-Verlag, Berlin, Heidelberg, 1996, pp 373-383.

[13] M. W. Carter, G. Laporte and S. Y. Lee. Examination timetabling: Algorithmic strategies and applications. *Journal of the Operational Research Society*, 1996, 47(3), pp 373-383.

[14] S. Casey and J. Thompson. GRASPing the examination scheduling problem. *In*: E. K. Burke and P. De Causmaecker (eds). *Practice and theory of automated timetabling: Selected papers from the fourth international conference.* Volume 2740 of Lecture notes in computer science. Springer-Verlag, Berlin, Heidelberg, 2003, pp 232-244.

[15] P. I. Cowling, G. Kendall and L. Han. An investigation of a hyperheuristic genetic algorithm applied to a trainer scheduling problem. *In: Proceedings of congress on evolutionary computation (CEC2002)*, 2002, pp 1185-1190.

[16] L. Di Gaspero. Recolour, shake and kick: A recipe for the examination timetabling problem. *In*: E. K. Burke and P. De Causmaecker (eds): *Proceedings of the fourth international conference on the practice and theory of automated timetabling*, Gent, Belgium, August 2002, pp 404-407.

[17] L. Di Gaspero and A. Schaerf. Tabu search techniques for examination timetabling. *In*: E. K. Burke and W. Erben (eds). *Practice and theory of automated timetabling: Selected papers from the third international conference.* Volume 2079 of Lecture notes in computer science. Springer-Verlag, Berlin, Heidelberg, 2001, pp 104-117.

[18] M. Gendreau and J-Y Potvin. Tabu Search. *In*: E. K. Burke and G. Kendall (eds). *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Methodologies*, Chapter 6. Springer 2005.

[19] L. Han, G. Kendall and P. Cowling. An adaptive length chromosome hyperheuristic genetic algorithm for a trainer scheduling problem. *In: Proceedings of the 4th Asia-Pacific conference on simulated evolution and learning (SEAL'02)*, Singapore, November 18-22, 2002, pp 267-271.

[20] L. Han and G. Kendall. Guided operators for a hyper-heuristic genetic algorithm. *In: Proceedings of the 16th Australian conference on artificial intelligence*, Perth, Australia, 2003, pp 807-820.

[21] L. Han and G. Kendall. Investigation of a tabu assisted hyper-heuristic genetic algorithm. *In: Proceedings of congress on evolutionary computation*, 2003, vol 3, pp 2230-2237.

[22] P. Hansen and N. Mladenović. Variable neighbourhood search: Principles and applications. *European journal of operational research*, 2001, 130, pp 449-467.

[23] P. Hansen and N. Mladenović. Variable Neighbourhood Search. *In*: E. K. Burke and G. Kendall (eds). *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Methodologies*, Chapter 8. Springer 2005.

[24] L. T. G. Merlot, N. Boland, B. D. Hughes and P. J. Stuckey. A hybrid algorithm for the examination timetabling problem. *In*: E. K. Burke and P. De Causmaecker (eds). *Practice and theory of automated timetabling: Selected papers from the fourth international conference.* Volume 2740 of Lecture notes in computer science. Springer-Verlag, Berlin, Heidelberg, 2003, pp 207-231.

[25] N. Mladenović and P. Hansen. Variable neighbourhood search. *Computers Ops. Res.*, 1997, 24(11), pp 1097-1100.

[26] L. Pacquete and T. Stützle. Empirical analysis of tabu search for the lexicographic optimization of the examination timetabling problem. *In*: E. K. Burke and P. De Causmaecker (eds): *Proceedings of the fourth international conference on the practice and theory of automated timetabling*, Gent, Belgium, August 2002, pp 413-420.

[27] S. Petrovic and E. K. Burke. Chapter 45: University timetabling. *In*: J. Leung (ed): *Handbook of scheduling: Algorithms, models, and performance analysis*, CRC Press, April 2004.

[28] S. Petrovic, G. Kendall and Y. Yang. A tabu search approach for graph-structured case retrieval. *In: Proceedings of the Starting artificial intelligence researchers symposium*, IOS Press, 2002, pp 55-64.

[29] P. Ross, E. Hart and D. Corne. Some observations about GA-based exam timetabling. *In*: E. K. Burke and M. W. Carter (eds). *Practice and theory of automated timetabling: Selected papers from the second international conference.*

Volume 1408 of Lecture notes in computer science. Springer-Verlag, Berlin, Heidelberg, 1998, pp 115-129.

[30] A. Schaerf. A survey of automated timetabling. *Artificial Intelligence Review*, 1999, 13(2), pp 87-127.

[31] H. Terashima-Marín, P. Ross and M. Valenzuela-Rendón. Evolution of constraint satisfaction strategies in examination timetabling. *In: Proceedings of the genetic and evolutionary conference*, Orlando, Florida, July 13-17 1999, pp 635-642. 998, 25, pp 637-648.

[32] J. Thompson and K. Dowsland. Variants of simulated annealing for the examination timetabling problem. *Annals of Operational Research*, 1996, 63, pp 105-128.

[33] J. Thompson and K. Dowsland. A robust simulated annealing based examination timetabling system. *Comput. Oper. Res.*, 1998, 25, pp 637-648.