MISTA 2013

An Investigation of Multistage Approaches to Examination Timetabling

Syariza Abdul Rahman · Andrzej Bargiela · Edmund Burke · Barry McCollum · Ender Özcan

Abstract Many successful approaches to examination timetabling consist of multiple stages, in which a constructive approach is used for finding a good initial solution, and then one or more improvement approaches are employed successively to further enhance the quality of the solution obtained during the previous stage. Moreover, there is a growing number of studies describing the success of approaches which make use of multiple neighbourhood structures. In this study, we investigate the methods of ordering neighbourhood structures within a Variable Neighbourhood Search approach using a great deluge move acceptance method. We also analyse how this approach performs as an improvement algorithm when combined with different initialisation strategies while performing multiple runs for examination timetabling. The empirical results over a well known examination timetabling benchmark show that the performance of Variable Neighbourhood Search great deluge performs competitively, ranking second among previously proposed approaches, with the right choice of initialisation and neighbourhood ordering methods.

Syariza Abdul Rahman
Universiti Utara Malaysia, School of Quantitative Sciences, College of Art and Science, 06010 Sintok, Malaysia
E-mail: syariza@uum.edu.my

Andrzej Bargiela and Ender Özcan
University of Nottingham, School of Computer Science, Jubilee Campus,Nottingham NG8 1BB, UK
E-mail: {abb,exo}@cs.nott.ac.uk

Edmund Burke
University of Stirling, Cottrell Building, Stirling FK9 4LA, UK
E-mail: e.k.burke@stir.ac.uk

Barry McCollum
University of Queen Belfast, School of Electronics, Electrical Engineering and Computer Science, University Road, Belfast, BT7 1NN, Northern Ireland, UK
E-mail: b.mccollum@qub.ac.uk

1 Introduction

Examination timetabling is a well known computationally difficult real-world combinatorial optimisation problem ([32], [19]). Constructing a solution requires the assignment of a set of examinations to a limited number of time-slots subject to certain constraints which are frequently classified as hard and soft. The hard constraints must be satisfied for a solution to be considered *feasible*. A standard example of a hard constraint is the requirement that students cannot sit two examinations at the same time. Soft constraints, such as allowing students free time between examinations represent what are commonly referred to as preferences. An examination timetabling solution method aims to reduce the number of violations of such constraints while producing feasible high quality solutions.

As an indication of the inherent difficulty of examination timetabling, one of the largest problem instances solved to optimality as a real world problem Yeditepe University is reported in [34]. It has thirty-eight scheduled examinations. Due to the size of real world instances, it is therefore not surprising that the goal of examination timetabling is frequently defined as finding a high quality schedule for a given set of examinations subject to various institutional and individual preferences. The approaches to examination timetabling are mostly based on two search paradigms: construction and improvement [13]. A constructive approach starts with an empty solution and incrementally builds towards a complete solution. For example, graph colouring heuristics are commonly used as construction heuristics in examination timetabling. On the other hand, an improvement approach uses a complete or feasible solution and is concerned with further refining the quality of that initial solution. In the work presented here, we use an iterative framework which employs a set of mutational neighbourhood structures making random modifications and using local search.

Solution methods for examination timetabling problems have been widely investigated over the last decade or so. These methods include exact ([38]), constraint-based ([29]), heuristic ([5]), meta-heuristic ([14]), hyper-heuristic ([12], [22]) and multi-objective ([33]) approaches. Meta-heuristic techniques can be thought of as usually comprising two major classes of methodology i.e. stochastic local search-based techniques that deal with a single candidate solution at each iteration, and population-based techniques that maintain a set of candidate solutions during the search process. The stochastic local search-based techniques include tabu search ([23]), simulated annealing ([9]), iterated local search ([17]), large neighbourhood search ([1]) and variable neighbourhood search ([11]), whilst the population-based techniques include genetic algorithms ([18]), memetic algorithms ([22])and ant algorithm ([21]). Some other recent approaches applied to timetabling are hyper-heuristics ([15], [36]), case-based reasoning ([41]), fuzzy-based methods([7]), granular modeling ([4]), the developmental approach ([35]) and a harmony search algorithm ([6]). A review of the major approaches in examination timetabling can be found in [19] and [37].

Many successful approaches to examination timetabling, for example [17] and [31], consist of multiple stages, in which a constructive approach is used for finding a good quality initial solution, and then one or more improvement approaches are employed successively to further improve the quality of solution obtained during the previous stage. In most of the cases in which single point based search is performed, there are two main stages. An approach is used to build an initial solution which is fed into an improvement algorithm. For example, the winning approach in the 2007 International Timetabling Competition (ITC2007), uses constraint programming to build a solution

which is then passed to a stochastic local search algorithm for improvement ([31]). If a deterministic approach is used to construct an initial solution and the improvement algorithm is a stochastic algorithm, then for each trial of the improvement algorithm for a given instance, the same initial solution has to be used. If the initialisation approach employs randomness within, then the improvement approach can use a different initial solution produced by that approach at each trial. The main objective of this study is to investigate the influence of various initialisation strategies on a stochastic constructive approach which is used in the first stage and the influence on neighbourhood ordering methods on the performance of variable neighbourhood search (VNS) algorithm which is used in the succeeding stage. Within the later stage, VNS is combined with a great deluge acceptance method, allowing the acceptance of some worsening solutions for solving the examination timetabling problem. A range of methods managing different neighbourhood structures were tested as a part of this approach. The influence of initialization strategies within the construction phase and neighbourhood orderings within the VNS during the second phase are investigated in terms of the quality of resultant solutions across the Toronto benchmark instances ([20]). As part of further work, it is intended to further apply the technique to the examination datasets associated with the ITC2007 competition ([27])

The components of the VNS approach for examination timetabling are described in Section 2, with a focus on the algorithmic components, neighbourhood structures, neighbourhood ordering, initialisations and acceptance criteria. Section 3 provides the experimental results and remarks. Finally, the conclusion is presented in Section 4.

2 VNS for Examination Timetabling

It is a common problem that the the search process guided solely by a local search method could get trapped at a local optimum. The use of a different neighbourhood structure could potentially lead to a different local optimum ([30], [24]). Therefore, the use of multiple neighbourhood structures were introduced in [30] with the aim of ensuring that the search process is able to continue to explore the local solution space without becoming trapped and therefore stagnating. Variable neighbourhood search (VNS) performs an iterative search over candidate solutions from one neighbourhood structure to another whenever necessary. This algorithm can be considered as a single point-based selection hyper-heuristic [13] which manages multiple neighbourhood structures.

2.1 Neighbourhood Structures

It is known that the choice of neighbourhood structure influences the search process in VNS ([30], [24]). The purpose of employing more than one neighbourhood structure is to be able to combine the strengths of multiple operators during the search process dynamically. If one neighbourhood structure fails to improve the current solution, then the other neighbourhood structures might still have a chance for a better exploitation. Similarly, if the search gets stuck at a local optimum, the step size of the mutation operator in use could be small to overcome the trap, but another operator which supports a larger step size could provide the diversification needed to explore the space better.

Recently, Kempe-chain moves have been successfully employed in solving timetabling problems ([11], [3]). A standard neighbourhood structure as discussed in [39] is formed by a single move neighbourhood which consists of choosing an examination randomly and rescheduling it into a new feasible time-slot. In this study, along with this simple neighbourhood structure, Kempe-chain moves are introduced.

In a general implementation, the Kempe-chain move operates over two subsets of examinations by swapping between two feasible time-slots. Each subset of examinations is connected by edges to represent conflict between the examinations. In this study, the neighbourhood structures allow for infeasible moves. In the case of the Kempe-chain neighbourhood, the infeasible move may be due to more examinations from different time-slots being connected to the chosen examination. One of the aims of using multiple neighbourhood structures in VNS is to shake the current solution in various ways. In this case, the neighbourhood structures shake the current solution by allowing an infeasible move. In order to treat this infeasibility, a repair mechanism is used by the VNS algorithm.

Fifteen neighbourhood structures for examination timetabling are considered in this study. Traditionally, neighborhood structures are ordered in increasing step size used to perturb a candidate solution. The biased neighbourhood structures (referred to as number 8, 9, 11 and 12 below) were also used as described in [11] and [2] which choose the examination generating the highest penalty value for rescheduling from a number of randomly selected examinations. The aim is to repair an examination that contributes a large value to the cost function. Experiments using different sets of neighbourhood structures would also be useful. The implemented neighbourhood structures are ordered as follows by choosing:

1. One examination at random and move to a new random feasible time-slot.
2. Two examinations at random and move each examination to a new random feasible time-slot.
3. Two examinations at random and swap the time-slots between these two examinations. The feasibility of the two examinations is maintained.
4. Three examinations at random and move each examination to a new random feasible time-slot.
5. Four examinations at random and move each examination to a new random feasible time-slot.
6. Five examinations at random and move each examination to a new random feasible time-slot.
7. One move of Kempe-chain with one random examination.
8. One move of Kempe-chain with one examination selected randomly from 10% selection of examinations that give highest penalty.
9. One move of Kempe-chain with one examination selected randomly from 20% selection of examinations that give highest penalty.
10. Two moves of Kempe-chain with one random examination.
11. Two moves of Kempe-chain with one examination selected randomly from 10% selection of examinations that give highest penalty.
12. Two moves of Kempe-chain with one examination selected randomly from 20% selection of examinations that give highest penalty.
13. Two time-slots at random and swap between them.
14. One time-slot at random and move to a new feasible time-slot.
15. Shuffle all time-slots at random.

2.2 Ordering of Neighbourhood Structures

The studies by [24], [10], [8] and [2] have shown that the ordering of neighbourhood structures within the VNS framework could considerably influence the solution quality obtained at the end. The neighbourhood structures are frequently ordered on the basis of a pre-defined sequence with increasing step size, to achieve a better solution quality at the end of a run. [10], Burke et al. introduced a parameter $success_k$ to penalise a non-improving neighbourhood structure. In the case that an improvement in the solution quality is detected, search always starts using the first neighbourhood with the smallest size. Any neighbourhood that has $success_k$ value which is less than 1, cannot have priority in the next iteration, since this indicates that the neighbourhood cannot improve the solution quality. In another study, [8] observed that a combination of three different neighbourhood structures yielded an improved performance. Nevertheless, a recent study by [11] showed that fixing the ordering of neighbourhood structures was not essential. The study demonstrated that VNS and a hybridisation with a genetic algorithm could produce a good quality solution. Since the solution quality was dependent on the selection of the neighbourhood, the genetic algorithm worked intelligently by selecting a neighbourhood structure from the VNS framework.

Motivated from these studies, an investigation has been undertaken into the ordering of neighbourhood structures, in which five variants of neighbourhood ordering strategies are explored. The first and second variants are the ordering of neighbourhood structures based on an increasing step size, as proposed in [10] and [2], respectively. The first variant is a basic VNS where the neighbourhood structure starts with $k = 1$. The solution search continues with the next neighbourhood k if the improvement to the solution quality can not be found. Once an improvement to the solution quality is found, the search starts back with $k = 1$; the second variant is related with the neighbourhood structure used in the previous iteration. In this variant, a neighbourhood k that has been used to improve the solution quality in the previous iteration is used in the next iteration for finding good solution quality. These variants of neighbourhood orderings are represented as 'basic VNS' and 'start-k', respectively.

The third variant of neighbourhood ordering is based on a strategy adapted from squeaky wheel optimisation [25], by assigning and increasing the penalty under a parameter (called 'priority') of a neigbourhood structure yielding a worsening move. If there is no improvement in the solution quality while using the current neighbourhood, then the neighbourhood with lowest priority is chosen next. This ordering strategy is denoted as 'adaptive I'. A variant of 'adaptive I' considers the effect of neighbourhood structures using a small step size which is a faster operation than the third ordering strategy, but this gives relatively greater priority to neighbourhood structures using a small step size, which is referred to as 'adaptive II'. In the case of ties, where more than one neighbourhood has the same priority value, then the neighbourhood structure with the smallest step size is chosen in the next iteration.

The neighbourhood structures are first classified with respect to the step size used in a move, as small and large. For example, a neighbourhood operator with a small step size consists of a single move of an examination to a new feasible time-slot, whilst a neighbourhood operator with a large step size consists of a number of examinations to be moved and might incur large differences in the solution quality when the chosen move takes place, for instance, when allocations in two time-slots are swapped at random. This neighbourhood structure involves all examinations in one time-slot to be swapped with all examinations in the other time-slot.

As the fifth variant of neighbourhood ordering, a reinforcement learning mechanism is also investigated. Reinforcement learning ('RL') is a mechanism that interacts with the behaviour of an environment by assigning punishments and rewards based on its performance. In this case, a neighbourhood operator is given a reward if there is improvement in the solution quality after its application, while a punishment is given if the move fails to improve the current solution.

2.3 Initialisation Strategies

This study is concerned with improving the initial solution obtained from the constructive approaches presented in a previous study. The main objective is to understand the influence of different kinds of initialisation strategies on the solution quality when an improvement approach is employed in searching the landscape starting from that initial solution which is locally optimal. Several studies showed that good initial solutions can ultimately translate to a further improvement in the solution quality within a short time ([16]). Starting from a different random point (solution) in the search space at each trial frequently helps, particularly if a stochastic local search algorithm is used for solving a given problem. In many cases, the initial solutions are extremely poor. Recent studies indicate the success of multi-stage hybrid algorithms in which an already improved solution is constructed and then it is further improved by another algorithm. While testing a two-stage stochastic local search algorithm, certain choices can be made. The algorithm dealing with the second stage will not receive a totally random initial solution and will be improving upon the solution generated in the first stage which will potentially be a locally optimal solution. Three different variants of initial solutions are considered to be used in the second stage of our algorithm, i.e. *poor*, *good* and *multiple*. The good initial solution is the best one obtained using the previously proposed constructive approaches, while a poor initial solution has a quality level which is at least 20% worse than the solution quality of that good initial solution. During the multiple trials of our stochastic local search algorithm for solving a given instance, good and poor initial solutions are used separately as an initial solution during all runs, while multiple initial solutions are used individually at each trial. It is possible that some of the multiple solutions are not feasible and can be dealt with during the improvement phase with a repair mechanism that is incorporated into the VNS-GD algorithm. The initial solutions are obtained from the constructive approach of Adaptive Heuristic Orderings described in [5].

2.4 Acceptance Criteria of VNS

Algorithm 1 illustrates the pseudo-code of the VNS algorithm using the great deluge algorithm for acceptance criteria. The great deluge algorithm is a local search approach that accepts a worsening solution based on an acceptance level of quality, denoted as B which is decreased with a certain amount called the decay rate, denoted as β. This approach has demonstrated a good performance for timetabling ([28], [40]). Variants of the decay rate have been investigated by [40], [26] and [28].

The decay rate, β for this study is set as 0.001. This value is chosen as a result of some initial tests which are not reported in this paper due to space restrictions. B is initialised using the initial solution quality. In Algorithm 1, let s be an initial solution

and is set as the best solution, s_{best}, obtained so far. The quality of solution s, $f(s)$ is set as $f(s_{best})$. While the algorithm starts the search, a solution s' is generated randomly by visiting the k^{th} neighbourhood sequentially until a local optimum s'' is found. The solution s'' is accepted whenever the solution quality of s'', $f(s'')$ is better than $f(s_{best})$. Otherwise, if $f(s'')$ is better than B, then the solution s'' is accepted. Afterwards, B is reduced proportional to β. Every time the solution quality, $f(s)$ is accepted, the search will continue with the next k that is identified based on an ordering strategy of neighbourhood structure described in Section 2.2.

Algorithm 1 Variable Neighbourhood Search - Great Deluge

Select the set of neighbourhood structures $N_k, k = 1, ..., k_{max}$, to be used in the search; choose stopping condition;
set initial solution s; $s_{best} \leftarrow s$; $f(s_{best} \leftarrow f(s))$;
Estimate the acceptance level of quality to be accepted, $B = f(s)$; set the decay rate, β;
Repeat until stopping criteria is satisfied:
1. Set $k := 1$;
2. Until $k = k_{max}$, repeat:
(a) Shaking: Generate a point s' at random from the k^{th} neighbourhood of $s(s' \in N_k(s))$;
Repair mechanism:
(b) Local Search: Apply a local search method with s' as initial solution until local optimum s'' is obtained.
(c) Move or not:
Calculate $f(s'')$
Great deluge acceptance criteria:
if $f(s'')$ is better than $f(s_{best})$ **then**
  $s \leftarrow s''$
  $s_{best} \leftarrow s''$
else
  if $f(s'')$ is better than B **then**
    $s \leftarrow s''$
  end if
end if
$B = B - \beta$
Continue the search with identified k.

The proposed approach is incorporated with a repair mechanism since infeasible moves can happen. The pseudo-code of the repair mechanism for the Toronto benchmark datasets is shown in Algorithm 2. Delta evaluation is always applied for fast execution of the proposed algorithm. An examination is checked against all possible values of delta_cost obtained by an assignment to each possible time-slot. Once the lowest delta_cost is found, the examination, i_{best} is moved to that best time-slot, j_{best}, and the process continues until there is no improvement in the solution quality.

3 Experiments and Results

The experiments in this study are performed using the examination timetabling datasets from various universities, introduced by [20][1]. The version I of the datasets as introduced in [37] is employed as a test bed for the proposed approaches. The objective for the Toronto benchmark problem is to create a feasible timetable so that no student is

[1] It can be accessed at ftp://ftp.mie.utoronto.ca/pub/carter/testprob/

Algorithm 2 Repair mechanism for the Toronto benchmark datasets

```
Repair mechanism:
while the delta_cost > 0 do
  for i = 1 to number of examinations do
    for j = 1 to number of slots do
      Find the lowest delta_cost;
      i_best ← i;
      j_best ← j;
    end for
  end for
  Move i_best to j_best
end while
```

required to sit two examinations at any one time. To achieve a high quality timetable, the soft constraints need to be satisfied as much as possible. Thus, during the timetable construction, it is required that student's examinations are assigned as far apart as possible. The proximity cost function introduced in [20] in conjunction with the introduced datasets was used in order to measure the quality of the obtained timetable and to describe the average penalty of students distributed in the examination schedule.

Pentium IV 1.86 GHz. Windows machines having 1.97 Gb memory were used during the experiments. The stopping conditions for the experiments were set as 50000 iterations. However, the running time for the initial solutions is not included during the improvement phase because of the challenge of comparing the performance of the algorithm using different initial solution values. 100 runs were obtained for each dataset tested with three different initial solutions and with different types of neighbourhood orderings. The results are provided in the tables below, each table representing the results for each initial solution tested with different neighbourhood orderings. The best solution for each dataset is represented in bold font.

Tables 1, 2 and 3 illustrate the results of poor, good and multiple initial solutions respectively, tested with different neighbourhood orderings for the Toronto benchmark datasets. The initial values of the poor and good solutions for each of the datasets are also provided in the tables. It is intended that these values will be improved using the VNS-GD algorithm, repeating each value for a hundred runs. However, since the multiple initial solution is generated at each run before proceeding to the improvement phase, only the average value of all generated initial solutions is provided.

The overall results with poor initial solution are presented in Table 1. They show that the basic VNS that always starts with $k = 1$ whenever there is improvement to the solution quality, generates most of the best solutions with four best results and two ties for kfu93 and sta83 I. With other neighbourhood orderings, start-k obtained four best results with one tie, adaptive I obtained one best result and one tie with other types of neighbourhood orderings, adaptive II obtained one best result but tied with basic VNS. RL obtained one best result and a tie for sta83 I. The standard deviation (of less than one) for different neighbourhood orderings is relatively small.

The results generated using the good initial solutions indicate the same pattern as for the poor initial solutions, where the basic VNS obtained the best results for six problems and a tie for sta83 I. On the other hand, the other neighbourhood ordering types obtained three best results for start-k and one best result for the adaptive II and RL. The rest of the neighbourhood orderings are tied for sta83 I. Considering the standard deviation value, the good initial solution provides less than the poor initial solution. The poor initial solution contributes a higher standard deviation as expected

Table 1 The results of poor initial solutions tested with different neighbourhood orderings for the Toronto benchmark datasets (RL = reinforcement learning, stdev. = standard deviation, Av. t(m) = average running time in minutes)

		One poor initial solution						
Problem	Initial value	Start-k	Basic VNS	Adaptive I	Adaptive II	RL	Av. t(m)	Best
car91	6.35	4.95	**4.89**	4.97	4.97	4.97	518.95	4.89
stdev.		0.06	0.06	0.06	0.07	0.07		
car92	5.43	4.19	4.18	**4.14**	4.15	4.18	274.76	4.14
stdev.		0.07	0.06	0.06	0.05	0.05		
ear83 I	47.85	**33.45**	33.55	33.52	33.63	33.60	18.56	33.45
stdev.		0.52	0.59	0.51	0.51	0.55		
hec92 I	13.91	10.29	**10.21**	10.25	10.38	10.33	1.33	10.21
stdev.		0.15	0.14	0.14	0.14	0.14		
kfu93	18.03	13.48	**13.46**	13.56	13.46	13.52	68.87	13.46
stdev.		0.15	0.15	0.15	0.15	0.14		
lse91	14.29	**10.53**	10.55	10.60	10.57	10.55	57.30	10.53
stdev.		0.19	0.19	0.15	0.17	0.19		
rye93	11.71	**8.37**	8.45	8.57	8.50	8.61	120.96	8.37
stdev.		0.10	0.10	0.10	0.11	0.11		
sta83 I	196.68	**157.06**	**157.06**	**157.06**	157.07	**157.06**	1.06	157.06
stdev.		0.06	0.05	0.06	0.06	0.05		
uta92 I	4.40	3.41	**3.40**	3.44	3.43	3.42	485.08	3.40
stdev.		0.05	0.04	0.03	0.04	0.04		
ute92	32.80	25.04	**24.96**	25.08	25.06	24.96	4.06	24.96
stdev.		0.10	0.10	0.11	0.11	0.10		
tre92	10.91	**8.18**	8.26	8.32	8.28	8.29	38.08	8.18
stdev.		0.11	0.13	0.09	0.10	0.10		
yor83 I	50.48	36.22	36.77	36.23	36.71	**36.05**	10.32	36.05
stdev.		0.40	0.40	0.39	0.36	0.44		

and this is due to the poor starting-point which tends to generate a variation in the final solution quality.

The results from the multiple initial solutions also appear to work well with this approach, indicating the same pattern when the solutions are generated. The basic VNS shows great success with the best solutions for six problems, while start-k, adaptive I and RL, each obtained one best result respectively while adaptive II obtained two best results. The rest of the neighbourhood orderings are tied for the sta83 I problem. The standard deviation among the neighbourhood orderings is clearly about the same, but there are some significant differences when different types of initialisation strategies are compared.

Among the problems, sta83 I shows almost no differences when employed with different types of neighbourhood ordering. Even when sta83 I is addressed with different initial solutions, the move is tended to get stuck at local optimum and no further improvement could be obtained. This may be due to the incorporation of the repair mechanism that checked and moved each examination and time-slot which could reduce the least penalty cost to the lowest level. Although there are variations in the standard deviation values for different initial solutions used for the sta83 I problem, the start with good initial solution shows that the standard deviations (less than 0.004) are very small.

The overall results demonstrate that the basic VNS-GD performed very well where it obtained most best results over the thirteen problems of the Toronto benchmark

Table 2 The results of good initial solutions tested with different neighbourhood orderings for the Toronto benchmark datasets (RL = reinforcement learning, stdev. = standard deviation, Av. t(m) = average running time in minutes)

		One good initial solution						
Problem	Initial value	Start-k	Basic VNS	Adaptive I	Adaptive II	RL	Av. t(m)	Best
car91	5.08	**4.86**	4.87	4.89	4.94	4.88	542.65	4.86
stdev.		0.03	0.03	0.03	0.02	0.03		
car92	4.34	4.24	4.22	4.22	4.21	**4.20**	286.64	4.20
stdev.		0.02	0.01	0.01	0.01	0.02		
ear83 I	36.91	34.36	33.81	34.08	**33.76**	34.17	17.62	33.76
stdev.		0.54	1.01	0.69	0.69	0.63		
hec92 I	11.13	**10.11**	10.19	10.28	10.23	10.30	1.38	10.11
stdev.		0.15	0.15	0.14	0.16	0.14		
kfu93	14.42	13.83	**13.62**	13.89	13.80	13.84	60.38	13.62
stdev.		0.06	0.09	0.05	0.07	0.06		
lse91	11.41	10.65	**10.58**	10.65	10.66	10.59	54.89	10.58
stdev.		0.12	0.13	0.14	0.12	0.11		
rye93	9.37	8.51	**8.45**	8.51	8.50	8.53	133.74	8.45
stdev.		0.08	0.08	0.08	0.09	0.08		
sta83 I	157.34	**157.08**	**157.08**	**157.08**	**157.08**	**157.08**	1.10	157.08
stdev.		0.00	0.00	0.00	0.00	0.00		
uta92 I	3.52	3.49	**3.46**	3.48	3.48	3.47	496.32	3.46
stdev.		0.00	0.01	0.01	0.01	0.01		
ute92	26.24	**24.95**	24.99	24.97	25.04	24.97	4.52	24.95
stdev.		0.12	0.13	0.11	0.11	0.13		
tre92	8.73	8.36	**8.28**	8.31	8.37	8.38	36.08	8.28
stdev.		0.06	0.08	0.06	0.06	0.06		
yor83 I	39.67	37.53	**36.51**	37.14	37.19	37.31	10.39	36.51
stdev.		0.35	0.52	0.42	0.38	0.39		

datasets when tested with various initialisations. This suggests that the neighbourhood ordering with respect to the size of the neighbourhood can affect the search for good solutions. On each occasion when there is improvement to the solution quality, the search always starts with a small neighbourhood structure. This would allow the search to explore more regions that cannot be achieved by other larger neighbourhood structures, while at the same time it could reduce the processing time because the search always begins with a small size neighbourhood structure.

The running time for this approach is quite long because of the incorporation of a repair mechanism. The repair mechanism for the Toronto benchmark datasets considers each examination to be repaired or improved, taking into consideration a move to time-slot that can reduce the current penalty cost to the lowest level. In this study, the repair mechanism not only works for the infeasible moves but it also tries to repair each examination assignment by reducing the assignment cost.

Any VNS-based approach initialised with the multiple solutions which are all obtained by the Adaptive Heuristic Orderings approach ([5]) performs the best based on the best-of-runs results across almost all instances. Moreover, initialising VNS-based approaches using a single solution during all trials regardless whether that solution is good or poor does not perform as well.

Tables 4 (a) and (b) illustrate the comparison of the results from different improvement approaches in the literature with our VNS-GD approach tested with multiple initial solution and basic VNS ordering strategies. In order to see the best approach,

Table 3 The results of multiple initial solutions tested with different neighbourhood orderings for the Toronto benchmark datasets (RL = reinforcement learning, stdev. = standard deviation, Av. t(m) = average running time in minutes)

	Av.	Multiple initial solution						
Problem	initial value	Start-k	Basic VNS	Adaptive I	Adaptive II	RL	Av. t(m)	Best
car91	5.65	4.87	4.88	4.89	4.87	**4.83**	519.42	4.83
stdev.		0.08	0.07	0.08	0.07	0.08		
car92	4.93	4.10	**4.06**	4.07	4.13	4.12	299.9	4.06
stdev.		0.06	0.05	0.06	0.18	0.06		
ear83 I	41.94	33.43	**33.22**	33.38	33.53	33.41	19.49	33.22
stdev.		0.52	0.70	0.47	0.47	0.63		
hec92 I	12.76	10.25	10.27	10.28	**10.23**	10.35	1.35	10.23
stdev.		0.16	0.17	0.16	0.14	0.15		
kfu93	16.23	13.39	**13.30**	13.57	13.46	13.41	66.36	13.30
stdev.		0.18	0.17	0.13	0.15	0.16		
lse91	12.82	10.45	10.45	**10.36**	10.38	10.46	56.81	10.36
stdev.		0.25	0.27	0.26	0.25	0.28		
rye93	11.08	8.56	**8.42**	8.53	8.48	8.53	140.12	8.42
stdev.		0.09	0.11	0.13	0.10	0.12		
sta83 I	160.16	**157.04**	**157.04**	**157.04**	**157.04**	**157.04**	1.37	157.04
stdev.		0.08	0.08	0.07	0.07	0.07		
uta92 I	3.88	**3.36**	3.38	3.37	**3.36**	3.40	504.24	3.36
stdev.		0.04	0.05	0.05	0.05	0.05		
ute92	28.55	24.97	**24.92**	24.99	24.93	24.93	4.38	24.92
stdev.		0.12	0.13	0.09	0.09	0.09		
tre92	9.58	8.25	**8.12**	8.22	8.24	8.22	39.38	8.12
stdev.		0.11	0.13	0.11	0.11	0.14		
yor83 I	45.18	36.48	**35.88**	36.27	36.37	35.96	10.69	35.88
stdev.		0.35	0.50	0.42	0.40	0.38		

the results of each problem are ranked and the best approach is identified based on the least average rank. The rank value of each approach is provided in brackets next to the solution quality in each table. Based on the average ranking of approaches across all instances considering their best performances, the VNS-GD approach is placed as the second best approach. However, it did not obtain the best result for any of the benchmark problems. The best approach is the study provided by [11] which employs the VNS approach hybridized with a genetic algorithm. The results of pur93 I were not included in the previous tables since it required a long running time and it was almost impossible to obtain the results for a hundred runs due to the size of the problem. The run for pur93 I was performed with only a good initial solution starting with solution quality (5.74), and was repeated only three times. The best result for pur93 I is presented in Table 4.

4 Conclusion

This study investigated different initialisation strategies in order to perform multiple runs using a variable neighbourhood search algorithm hybridised with a great deluge move acceptance method (VNS-GD). The overall framework represents a two-stage stochastic search algorithm which uses the VNS-GD in the second stage. We have simulated the behavior of a deterministic algorithm in the first stage by using a single

Table 4 Comparison of different improvement approaches with VNS-GD. The rank of each approach as compared to the rest for each instance is indicated in parenthesis.

Problem	[23]	[33]	[16]	[29]	[41]	[18]
car91	6.2 (10)	-	4.65 (3)	5.1 (6)	**4.5** (1)	5.4 (9)
car92	5.2 (10)	-	4.1 (4.5)	4.3 (7.5)	3.93 (2)	4.2 (6)
ears83 I	45.7 (12)	38.9 (11)	37.05 (10)	35.1 (8)	33.71 (4)	34.2 (5)
hec92 I	12.4 (12)	11.2 (10)	11.54 (11)	10.6 (6)	10.83 (8)	10.4 (5)
kfu93	18 (12)	16.5 (11)	13.9 (8)	13.5 (4.5)	13.82 (7)	14.3 (9)
lse91	15.5 (12)	13.2 (11)	10.82 (8)	10.5 (7)	10.35 (5)	11.3 (9.5)
pur93 I	-	-	-	-	-	-
rye92	-	-	-	8.4 (2)	8.53 (4)	8.8 (6)
sta83 I	160.8 (11)	158.1 (6)	168.73 (12)	157.3 (4.5)	158.35 (9)	157 (2)
tre92	10 (12)	9.3 (10)	8.35 (5)	8.4 (6.5)	7.92 (3)	8.6 (8.5)
uta92 I	4.2 (11)	-	3.2 (3.5)	3.5 (8)	**3.14** (1)	3.2 (3.5)
ute92	29 (12)	27.8 (11)	25.83 (7)	25.1 (4)	25.39 (6)	25.3 (5)
yor83 I	41 (12)	38.9 (10)	37.28 (8)	37.4 (9)	36.53 (7)	36.4 (6)
Av. Rank	11.15	10.38	7.62	6.35	5.04	6.38
Rank	12	11	9	6	5	7

(a)

Problem	[1]	[21]	[17]	[11]	[40]	VNS-GD
car91	5.2 (7)	5.2 (7)	6.6 (11)	4.6 (2)	4.8 (4)	4.88 (5)
car92	4.4 (9)	4.3 (7.5)	6 (11)	**3.9** (1)	4.1 (4.5)	4.06 (3)
ears83 I	34.9 (6)	36.8 (9)	**29.3** (1)	32.8 (2)	34.92 (7)	33.22 (3)
hec92 I	10.3 (4)	11.1 (9)	**9.2** (1)	10 (2)	10.73 (7)	10.27 (3)
kfu93	13.5 (4.5)	14.5 (10)	13.8 (6)	**13.0** (1.5)	**13.0** (1.5)	13.3 (3)
lse91	10.2 (4)	11.3 (9.5)	**9.6** (1)	10 (2)	10.01 (3)	10.45 (6)
pur93 I	-	4.6 (2)	**3.7** (1)	-	4.73 (3)	5.71 (4)
rye92	8.7 (5)	9.8 (8)	**6.8** (1)	-	9.65 (7)	8.42 (3)
sta83 I	159.2 (10)	157.3 (4.5)	158.2 (7)	**156.9** (1)	158.26 (8)	157.04 (3)
tre92	8.4 (6.5)	8.6 (8.5)	9.4 (11)	7.9 (2)	**7.88** (1)	8.12 (4)
uta92 I	3.6 (10)	3.5 (8)	3.5 (8)	3.2 (3.5)	3.2 (3.5)	3.38 (6)
ute92	26 (8)	26.4 (10)	**24.4** (1)	24.8 (2)	26.11 (9)	24.92 (3)
yor83 I	36.2 (3.5)	39.3 (11)	36.2 (3.5)	**34.9** (1)	36.22 (5)	35.88 (2)
Av. Rank	6.62	8.00	4.88	3.00	4.88	3.69
Rank	8	10	3.5	1	3.5	2

(b)

poor solution and a single good solution obtained from previously proposed approaches for each instance. The same poor and good solutions are used separately at each run of the proposed algorithm to see their influence on the performance of the overall algorithm. Also, we used a previously proposed stochastic constructive algorithm to build multiple solutions for each run of VNS-GD. Moreover, the performance of the overall framework combining different neighbourhood ordering methods with different initialisation strategies is tested. Various initialisations and neighbourhood ordering influenced the resultant solution quality as expected. The use of different initial solutions at each run, even though these solutions are already locally optimum, still demonstrates good performance for examination timetabling. Considering the neighbourhood orderings, the overall framework performed the best with a strategy which grows the step size whenever there is no improvement on the Toronto benchmark. Additionally, our approach ranks the second considering all instances based on the best of runs for each

instance when compared to previously proposed approaches. The intention is to fully analyse and understand the results presented here before applying the method to, in the first instance, the ITC2007 examination benchmark datasets. Although the results so far are extremely encouraging, it is important to carry out further evaluation and testing.

References

1. Abdullah, S., Ahmadi, S., Burke, E. K., & Dror, M., Investigating Ahuja-Orlin's large neighbourhood search approach for examination timetabling, OR Spectrum, 29(2), 351-372 (2007)
2. Abdullah, S., Burke, E. K. & McCollum, B., An investigation of variable neighbourhood search for university course timetabling, Proceedings of the 2nd Multi-disciplinary International Conference on Scheduling: Theory and Applications (MISTA), New York, USA, 18-21 July, 413-427. (2005)
3. Abdullah, S., Shaker, K., McCollum, B. & McMullan, P., Incorporating great deluge with Kempe chain neighbourhood structure for the enrolment-based course timetabling problem, Yu, J., Greco, S., Lingras, P., Wang, G. & Skowron, A. (Eds.), Rough Set and Knowledge Technology, Springer Berlin / Heidelberg, 6401, 70-77. (2010)
4. Abdul Rahim, S. K., Bargiela, A., & Qu, R., Granular modelling of exam to slot allocation, In roceedings of the 23rd european conference on modelling and simulation (ECMS), Madrid, Spain, 861-866. (2009)
5. Abdul Rahman, S., Bargiela, A., Burke, E. K., McCollum, B., & Özcan, E., Construction of examination timetables based on ordering heuristics, In proceedings of the 24th international symposium on computer and information sciences, 727-732. (2009)
6. Al-Betar, M. A., Khader, A. T. & Thomas, J. J., A combination of metaheuristic components based on harmony search for the uncapacitated examination timetabling, Proceedings of the 8th International Conference on the Practice and Theory of Automated Timetabling, (PATAT 2010), 10-13 August 2010, Belfast, Northern Ireland, (2010)
7. Asmuni, H., Burke, E. K., Garibaldi, J. M., McCollum, B., & Parkes, A. J., An investigation of fuzzy multiple heuristic orderings in the construction of university examination timetables, Computers and Operations Research, 36(4), 981-1001 (2009)
8. Bolaji, A. L., Khader, A. T., Al-Betar, M. A., Awadallah, M. A. & Thomas, J. J., The effect of neighborhood structures on examination timetabling with artificial bee colony, Proceedings of the 9th International Conference on the Practice and Theory of Automated Timetabling, (PATAT 2012), 29-31 August 2012, Son, Norway, (2012)
9. Burke, E. K., Bykov, Y., Newall, J. P., & Petrovic, S., A time-predefined local search approach to exam timetabling problem, IIE Transactions, 36(6), 509-528 (2004)
10. Burke, E. K., De Causmaecker, P., Petrovic, S. & Vanden Berghe, G., Metaheuristics: Computer decision-making, Chapter 7: Variable neighbourhood search for nurse rostering problems, Resende, M. G. C. & de Sousa, J. P. (Eds.) Kluwer, 153-172. (2003)
11. Burke, E. K., Eckersley, A. J., McCollum, B., Petrovic, S., & Qu, R., Hybrid variable neighbourhood approaches to university exam timetabling, European Journal of Operational Research, 206(1), 46-53 (2010)
12. Burke, E. K., Hart, E., Kendall, G., Newall, J., Ross, P. & Schulenburg, S., Handbook of meta-heuristics Hyper-heuristics: An emerging direction in modern search technology, Glover, F. & Kochenberger, G. (Eds.) Kluwer, 457-474. (2003)
13. Burke, E. K., Hyde, M., Kendall, G., Ochoa, G., & Özcan, E., A classification of hyper-heuristic approaches, In M. Gendreau and J-Y Potvin (Eds.), Handbook of metaheuristics, 449-468. Springer (2010)
14. Burke, E. K. & Kendall, G. (Eds.) Search methodologies: Introductory tutorials in optimization and decision support techniques, Springer, (2005)
15. Burke, E. K., Mccollum, B., Meisels, A., Petrovic, S., & Qu, R., A graph-based hyper-heuristic for educational timetabling problems, European Journal of Operational Research, 176, 177-192 (2007)
16. Burke, E. K. & Newall, J. P., Enhancing timetable solutions with local search methods, Lecture notes in computer science: Practice and theory of automated timetabling IV: selected papers from the 4th international conference, Springer-Verlag, 2740, 195-206. (2003)

17. Caramia, M., Dell'Olmo, P, & Italiano, G. F., Novel local search based approaches to university examination timetabling, INFORMS Journal of Computing, 20, 86-99 (2008)
18. Côté, P., Wong, T. & Sabourin, R., A hybrid multi-objective evolutionary algorithm to the uncapacitated exam proximity problem, Lecture Notes in Computer Science: Selected papers from the 5th International Conference on Practice and Theory of Automated Timetabling (PATAT 2004), 3616, 151-168. (2005)
19. Carter, M. W. & Laporte, G., Recent developments in practical examination timetabling, Selected papers from the first international conference on practice and theory of automated timetabling, 3-21. Springer-Verlag (1996)
20. Carter, M. W., Laporte, G. & Lee, S., Examination timetabling: Algorithmic strategies and applications, Journal of the Operational Research Society, 47, 373-383 (1996)
21. Eley, M., Ant algorithms for the exam timetabling problem, Burke, E. K. & Rudova, H. (Eds.), Lecture notes in computer science: Practice and theory of automated timetabling VI: selected papers from the 6th international conference, Berlin: Springer, 3867, 364-382. (2007)
22. Ersoy, E., Özcan, E., & Sima Uyar, A., Memetic algorithms and hyperhill-climbers, In P. Baptiste, G. Kendall, A. M. Kordon, & F. Sourd (Eds.), Lecture notes in computer science: Multidisciplinary international conference on scheduling: theory and applications: selected papers from the 3rd international conference, 159-166. (2007)
23. Di Gaspero, L. & Schaerf, A., Tabu search techniques for examination timetabling, Lecture notes in computer science: Practice and theory of automated timetabling III: selected papers from the 3rd international conference, Berlin: Springer, 104-117. (2001)
24. Hansen, P. & Mladenović, N., Variable neighborhood search: Principles and applications, European Journal of Operational Research, 130, 449-467 (2001)
25. Joslin, D. E., & Clements, D. P., "Squeaky wheel" optimization, Journal of Artificial Intelligence Research, 10, 353-37 (1999)
26. Landa-Silva, D. & Obit, J. H., Great deluge with nonlinear decay rate for solving course timetabling problems, Proceedings of the 2008 IEEE Conference on Intelligent Systems (IS 2008), IEEE press, 11-18. (2008)
27. McCollum, B., McMullan, P., Burke, E. K., Parkes, A. J., & Qu, R. (2008). A new model for automated examination timetabling. Annals of Operations Research, 2, 2-3.
28. McCollum, B., McMullan, P., Parkes, A. J., Burke, E. K., & Abdullah, S., An extended great deluge approach to the examination timetabling problem, In proceedings of the 4th multidisciplinary international conference on scheduling: theory and applications (MISTA09), Dublin 424-434. (2009)
29. Merlot, L. T. G., Boland, N., Hughes, B. D. & Stuckey, P. J., A hybrid algorithm for the examination timetabling problem, Burke, E. K. & Erben, W. (Eds.), Lecture notes in computer science: Practice and theory of automated timetabling V: selected papers from the 4th international conference, Berlin: Springer, 2740, 207-231. (2003)
30. Mladenović, N. & Hansen, P., Variable neighborhood search, Computers & Operations Research, 24, 1097-1100 (1997)
31. Müller, T., ITC2007 solver description: a hybrid approach Annals OR, 172, 429-446 (2009)
32. Schindl, D., Some new hereditary classes where graph coloring remains NP-hard, Discrete Mathematics, 295 (1-3), 197-202 (2005)
33. Paquete, L. F. & Fortseca, C. M., A study of examination timetabling with multiobjective evolutionary algorithms, Proceedings of the 4th Metaheuristics International Conference (MIC 2001), 149-154. (2001)
34. Parkes, A. & Özcan, E., Properties of Yeditepe examination timetabling benchmark instances, Proceedings of the 8th International Conference on the Practice and Theory of Automated Timetabling, 531-534. (2010)
35. Pillay, N., The revised developmental approach to the uncapacitated examination timetabling problem, Proceedings of the 2009 Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists, 187-192. (2009)
36. Pillay, N., Hyper-heuristics for educational timetabling, Proceedings of the 9th International Conference on the Practice and Theory of Automated Timetabling, 316-340. (2012)
37. Qu, R., Burke, E. K., Mccollum, B., Merlot, L. T., & Lee, S. Y., A survey of search methodologies and automated system development for examination timetabling, Journal of Scheduling, 12(1), 55-89 (2009)
38. Qu, R., He, F. & Burke, E. K., Hybridizing integer programming models with an adaptive decomposition approach for exam timetabling problems, Proceedings of the 4th Multidisciplinary International Scheduling: Theory and Applications 2009, 435-446. (2009)

39. Thompson, J. & Dowsland, K., Variants of simulated annealing for the examination timetabling problem, Annals of Operational Research, 63, 105-128 (1996)
40. Turabieh, H., & Abdullah, S., An integrated hybrid approach to the examination timetabling problem, Omega, 39, 598-607 (2011)
41. Yang, Y. & Petrovic, S., A novel similarity measure for heuristic selection in examination timetabling, Proccedings of the 5th International Conference on the Practice and Theory of Automated Timetabling (PATAT 2004), (2004)