Hybrid Variable Neighbourhood Approaches to University Exam Timetabling

E. K. Burke[1], A. J. Eckersley[2], B. McCollum[3], S. Petrovic[1], and R. Qu[1⋆]

[1] Automated Scheduling, Optimisation and Planning (ASAP) Group
School of CSiT, University of Nottingham, Nottingham, NG8 1BB, U.K.
ekb@cs.nott.ac.uk, sxp@cs.nott.ac.uk, rxq@cs.nott.ac.uk
[2] Foster Findlay Associates Ltd, Newcastle Technopole, Kings Manor
Newcastle Upon Tyne, NE1 6PA, U.K.
adam@ffa.co.uk
[3] School of Computer Science, Queens University, Belfast
University Road, N. Ireland, BT7 1NN
b.mccollum@qub.ac.uk

Abstract. In this paper, we investigate variable neighbourhood search (VNS) approaches for the university examination timetabling problem. In addition to a basic VNS method, we introduce variants of the technique with different initialisation methods including a biased VNS and its hybridisation with a Genetic Algorithm. A number of different neighbourhood structures are analysed. It is demonstrated that the proposed technique is able to produce high quality solutions across a wide range of benchmark problem instances. In particular, we demonstrate that the Genetic Algorithm, which intelligently selects approporiate neighbourhoods to use within the biased VNS produces the best known results in the literature, in terms of solution quality, on some of the the benchmark instances, although it requires relatively large amount of computational time. Possible extensions to this overall approach are also discussed.

1 Introduction

A very important factor which affects 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, i.e. all solutions that can be reached from the current solution by moving a single element. The most basic steepest descent technique simply takes the steepest route down to a nearby local minimum and terminates there. More sophisticated methodologies such as Simulated Annealing [1] and Tabu search [30] 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 exam timetabling [50].

The issue of how much the choice of neighbourhood affects the quality of solutions is considered by Thompson & Dowsland [56, 57]. The conclusion they arrive at

⋆ Corresponding author. Tel: ++44 115 8466503, Fax: ++44 115 8467877, E-mail: rxq@cs.nott.ac.uk

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 [42] proposed the Variable Neighbourhood (VNS) meta-heuristic for solving difficult optimisation problems. This approach changes the neighbourhood during the search. Since the neighbourhoods are varied regularly, there is no need to accept worsening solutions to escape local minima. It is both versatile and successful when applied to a range of different problem domains [34]. Mladenović and Hansen elaborated upon and systemised practices that had been originally introduced by Glover [31] to determine a powerful and effective search method. This work presented in this paper is evidence of the strength of the technique.

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 obtained. 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 where a single exam is reallocated to a new feasible timeslot. This neighbourhood, when used alone, can yield a disconnected search space when exams which clash with another exam in every other timeslot often cannot be moved at all. The aim of our research is to investigate the neighbourhoods within VNS methodology in the context of the university exam timetabling problem.

1.1 Examination Timetabling Problems

The formulation of the exam timetabling problem considered as part of this work [4] consists of

- a set of exams $E = \{e_1, e_2, ..., e_n\}$;
- a limited number of ordered timeslots (time periods) $T = \{t_1, t_2, ..., t_k\}$;
- a limited number of rooms of certain capacity in each timeslot $C = \{c_1, c_2, ..., c_k\}$;

where n and k are non-negative integers. The problem is to assign all the exams into the timeslots subject to a set of constraints [11, 23, 50], which can be divided into two categories, known as *hard* and *soft* constraints.

Hard constraints are defined as those which must be satisfied in order to have a feasible for use. A common hard constraint is that *conflicting* exams which have common students cannot be scheduled into the same timeslot. That is:

$$\forall\, e_s, e_p \in E, (s \neq p \text{ and } D_{e_s e_p} > 0) \Rightarrow x_{e_s} \neq x_{e_p}$$

where $D_{e_s e_p}$ is the number of students in both exams e_s and e_p; x_{e_s} is the timeslot to which exam e_s is assigned.

Soft constraints are less essential and violation of them is acceptable, although still undesirable. The violation of these soft constraints is used to evaluate the quality of a given feasible solution.

[4] Although a new formulation of the problem has been introduced as part of the 2nd International Timetabling Competition (ITC2007), the competition was not finished at the time of submission of this work. Please refer to the section on future research directions.

As a result of more students and a significantly increased level of modularisation of courses at many universities [11, 37], the number and variety of the constraints has increased markedly in recent years. This, in turn, makes the exam timetabling problem not only more difficult to solve, but also potentially very different from year to year.

1.2 Examination Timetabling Techniques

Exam timetabling has been a well studied topic across both Operational Research and Artificial Intelligence since the 1960's [**?**,23, 50]. Meta-heuristic [31] approaches to exam timetabling have been particularly successful, including Tabu Search (e.g. [28, 58]), Simulated Annealing (e.g. [41, 56, 57]), Great Deluge [8, 18], GRASP [25] and Evolutionary Algorithms (e.g. [17, 20]). Other variety of work has investigated case based reasoning [15, 59], fuzzy techniques [5], an ant algorithm [26], multi-objective method [9], artificial immune systems [36] and constraint programming techniques [41]. Hybridisations between meta-heuristics and other techniques such as graph heuristics (e.g. [10, 16, 20, 47, 49]), integer programming[51] and constraint based techniques (e.g. [55]) have also been shown to be very effective. A selection of overview and survey papers detailing many of these techniques applied to exam timetabling problems include [13, 21, **?**,23, 37, 46, 50, 53].

Another state-of-the-art development is the investigation of more general techniques which can automatically adapt to a wide range of problems. Hyper-heuristics [12] represent one of the research directions that are motivated by this goal. They search over (low level) heuristics rather than over potential solutions. Examples of papers which have explored hyper-heuristics for exam timetabling include [6, 10, 15, 16, 29, 32, 35, 45, 47, 49, 52, 55].

Another trend of research is concerned with exploring neighbourhood structures with the aim of designing more flexible approaches (e.g. VNS [42]) and very large neighborhood search [4, 44]) on exam timetabling [2, 3, 47]. This paper investigates VNS for exam timetabling problems.

VNS has started attracting attention in exam timetabling research. In [47], a steepest descent VNS has been employed within a hyper-heuristic framework to search for sequences of graph heuristics, which are then applied to construct exam timetables. It was observed that, compared to Tabu Search and Steepest Descent, VNS was much more flexible in searching the search space of graph heuristics, and thus could generate better solutions on exam timetabling problems.

The aim of our research in this paper is to develop VNS techniques which use a variety of different neighbourhoods in order to increase the generality of the method. Section 2 introduces a generic VNS technique and its application to exam timetabling. Section 3 presents a combined Genetic Algorithm and VNS approach (VNS-GA) to exam timetabling to improve solution quality by using the intelligent selection of neighbourhoods. Our results and analysis for both VNS and VNS-GA are presented in Section 4 and overall conclusions are given in Section 5.

2 The Variable Neighbourhood Search Methodology

The use of more than one neighbourhood within a search provides a very effective method of escaping from a local optimum. Indeed, 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 connectivity of the search space, but also enabling a simple, yet powerful search technique to yield high quality results.

Figure 1 presents the basic modified VNS framework we are studying in this paper (steps with '*' indicate our modifications). A number of variants of the approach are investigated (see later sections) to improve its performance. Any finite number, k_{max}, of pre-defined neighbourhoods may be used. Stopping conditions may be defined as for any local search technique. The move selected at step 2 (a) is randomly generated, avoiding issues of cycling, and the local search is performed using a single neighbourhood. A large number of variations of the basic VNS are discussed in [33].

Basic VNS Algorithm

- *Initialisation*: Select the set of neighbourhood structures N_k, $k = 1, \ldots, k_{max}$; 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 $x' \in N_k(x)$ at random from the k^{th} neighbourhood of x;
     (b) **Local search*: Apply a local search (a simple steepest descent in this paper) to x', until a local optimum, x'', is obtained;
     (c) **Move or not*: If x'' is better than the incumbent solution x then $x \leftarrow x''$, and continue the search with N_k; otherwise, set $k = k + 1$;

Fig. 1. The steps of the modified basic VNS framework.

2.1 Neighbourhoods used in Basic VNS

In order to keep the number of parameters in our VNS as small as possible, the local search within VNS uses a simple steepest descent approach (step 2 (b) * in Figure 1), which is fast and yields very good results. Instead of continuing the search with neighbourhood N_1 each time an improvement is found (as in standard basic VNS), the search continues using the current neighbourhood which yielded the improvement (continue with N_k in step 2 (c) * in Figure 1). This places slightly less reliance upon the ordering of the neighbourhoods and focuses the search on each neighbourhood.

Experiments with the basic VNS in Figure 1 so far have not been as effective as with the modified one. Hence, further experiments are all performed with this modified basic VNS.

In exam timetabling, the neighbourhoods used in local search techniques generally consist of moving some subset of exams from their current timeslot to a new timeslot. Our implementation of VNS used the following nine neighbourhoods:

- *Single move* (N_1) selects a single exam and moves it to a new feasible time slot. This neighbourhood, although widely used in timetabling, can be quite limiting as many exams in a timetable have no other feasible slots to move to.
- *Swap* (N_2) swaps the timeslots of a pair of exams, e_s and e_p whilst preserving the feasibility.
- *Move 2 exams randomly* (N_3): Two exams are chosen at random and moved to new feasible timeslots, independently of each other.
- *Move 3, 4 or 5 exams randomly* (N_4, N_5, N_6): As above but with three, four, or five exams.
- *Move a whole timeslot* (N_7) moves an entire timeslot, selected at random, with all the exams in it, to a new position p (also randomly selected) in a timetable. The original timeslots with all their exams after p are moved forward to the next timeslot in the timetable. This enables exams which would otherwise be unable to move around the timetable, thus allows for exploration of a wide area of the search space.
- *Swap timeslots* (N_8) simply swap all exams in one timeslot with the other, both randomly selected.
- *Kempe chain neighbourhood* (N_9) swaps a subset of exams which clash in two distinct timeslots. Given a starting exam, a chain of clashed exams in two timeslots can be formed and swapped. For example, if a starting exam e_s in timeslot t_i clashes with a subset of exams e_p in t_j, which clash with another subset of exams e_q in e_i, then a chain of exams $e_s \rightarrow e_p \rightarrow e_q$ is formed. N_9 swaps e_s e_q in t_i and e_p in t_j, thus enables any exam within the timetable to be moved more flexibly to a new timeslot. It was shown to be very successful within a Simulated Annealing technique [56] for exam timetabling.

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 steps in Figure 1. A number of variations are listed below:

- *Descent-ascent*: Basic VNS is a 'descent, first improvement method with randomization' [33]. A very simple change to it is to accept worsening moves with some probability (similar to that used in Simulated Annealing). We consider, in this work, an acceptance criterion that only solutions which are less than 1% worse than the current solution are considered in step 2 (c). This variant adds in further parameters, but can yield some improvement without these parameters having been tuned to each individual problem.

– *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. In this work, we investigate selecting an exam at random from the n (n = 5% and 20%) exams causing the highest penalty and use these exams to start a move by the Kempe chain neighbourhood. This is to bias the moves on the most troublesome exams, which are usually difficult to move by simple neighbourhoods. Results from this variation of VNS are presented in Section 4 together with those of basic VNS for comparison.
– *Problem-specific neighbourhoods*: The structure of different exam timetabling problems can vary significantly, thus quite often some neighbourhoods work far better on one problem than another. We investigate in this work the effect of reducing the number of neighbourhoods to the *best* subset for the specific problem (see Section 4).
– *Different initialisation strategies*: We use two different initialisation strategies to seed VNS: a greedy and a random construction technique. The largest degree graph colouring heuristic assignes exams, ordered by the number of clashes they have with the other exams, to the feasible timeslot which yields the least penalty. In the randomised variant, a random feasible timeslot is chosen for the next exam.

In addition to the above variants, we also investigated the best improvement, the variable neighbourhood descent and complex local search (great deluge or Simulated Annealing) methods (see more details in [33]). It was observed that these variants do not bring any benefit in terms of solution quality, and also increase the run time, thus they are not considered here. Also neighbourhoods consisting of up to five Kempe chain moves did not yield any successful results when used in VNS.

Another interesting observation concerns the *problem-specific neighbourhood* selection in VNS. Statistics collated from many runs of VNS on the 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. Rather than just use these fairly crude statistics we investigate a Genetic Algorithm (GA) to intelligently select the best neighbourhoods within VNS.

3 The Hybrid VNS with a Genetic Algorithm

3.1 The Hybrid VNS Approach

The idea of using a GA at a higher level of abstraction rather than being applied directly to the problem itself is strongly connected to recent work on hyper-heuristics [6, 12, 16, 29, 32, 35, 45, 47, 52, 55] and case based timetabling heuristic selection [10, 15, 59]. These algorithms select from a variety of low-level techniques the best one to apply to a problem. The key difference is that in the hyper-heuristic framework, low-level heuristics are being ordered and applied sequentially to the problem, whereas in our VNS, all neighbourhoods are searched, but a move is only made within a given neighbourhood if it fulfils the criteria for move acceptance. Our work in using a GA to select a subset of neighborhoods in VNS for specific problems is also linked to current

research on fine tuning parameters within algorithms [43]. Our work concerns the further improvement upon the VNS by using a GA for neighbourhood configuration. As far as we are aware, this is an entirely new approach in both VNS and GA research on timetabling problems.

The Genetic Algorithm is used to evolve a subset of neighbourhoods from a large pool for use within the VNS framework (referred to hereafter as VNS-GA). Figure 2 presents the pseudo-code of the hybrid GA algorithm. The chromosome represents the set of neighbourhoods to be used within VNS, where the ordering of neighbourhoods is unimportant since the VNS method cycles through all neighbourhoods (moves to the next neighbourhood when the current neighbourhood is not accepted, see the modified basic VNS in Figure 1). Duplicates of neighbourhoods in the chromosomes are removed when they are translated to the actual set of neighbourhoods to be used within VNS. A chromosome in which all elements are the same would represent just that single neighbourhood supplied to the VNS.

Hybrid Genetic Algorithm

1. *Initialisation*: randomly generate an initial population X_1 of chromosomes $\{c_1, c_2, ..., c_m\}$, $c_i = (n_1, n_2, ..., n_k)$, $n_j \in \{N_1, ..., N_k\}$ (see Table 1), k as the total number of neighborhoods defined;
2. Calculate the fitness of all chromosomes c_i in the population (see Figure 3)
3. *Repeat* for the number of generations:
   - For 70 per cent of the current population
     (a) Randomly select two parents, at a probability of $P(c_i)$ (see formula 1), using a roulette wheel style selection;
     (b) Replace the chosen two parents by the two offspring generated using one point corssover;
     (c) Mutation to change randomly one element in a chromosome;
   - Calculate the fitness of all chromosomes c_i in the population (see Figure 3)

Fig. 2. The hybrid genetic algorithm selecting subset of neighborhoods within VNS (VNS-GA).

Fitness Evaluation of Chromosomes in a Population

- For all chromosomes c_i in the population
  1. Remove duplicate neighborhoods in chromosome c_i;
  2. Produce the solution by using the modified basic VNS (see Figure 1) with the neighborhoods in c_i;
  3. Calculate the fitness of c_i using the fitness evaluation (see formula 2)

Fig. 3. The fitness evaluation for chromosomes in a GA population.

The probability, $P(c_i)$, of a chromosome c_i being selected from the population X_g in generation g is given in formula (1) with the fitness function given in formula (2).

$$P(c_i) = fitness(c_i)/(\sum_{\forall c_j \in X_g} fitness(c_j)) \quad (1)$$

$$fitness(c_i) = \max\{(\max_{\forall c_j \in X_1} \{VNS(c_j)\} \times f) - VNS(c_i), 0\} \quad (2)$$

$VNS(c_j)$ gives the solution penalty by applying the VNS with the neighbourhoods specified in chromosome c_j. f is a fitness modifier. A lower value of f gives better chromosomes a higher 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 smaller.

3.2 The Neighbourhoods

Exam timetabling problems quite often differ greatly in their structures, so different neighbourhoods suit different problems. Increasing the number of neighbourhoods tested and taking note of which neighbourhoods are most efficient when optimising different problems can give a good indication of this.

With this aim, we increase the number of neighbourhoods to 23 (Table 1) to cover a higher number of possible neighbourhoods when using the VNS-GA technique. This represents a good set of the neighbourhoods that have been used in exam timetabling. This number can be easily extended due to the intelligent selection of GA upon the neighbourhoods within VNS. 'Type A' and 'Type B' represent the Kempe chain move with the first exam selected at random from the 5% and 20% of the total exams that give the highest penalty, respectively.

Table 1. Neighbourhoods defined for VNS-GA

N_i ID	**Neighbourhood structures**
1.	Single random Kempe chain move
2.	Swap two exams
3. 5. 7. & 9.	Move two, three, four, and five exams at random
4. 6. 8. & 10.	Make two, three, four, and five random Kempe chain moves
11. & 12.	Make one Kempe chain move (Type A & Type B)
13. & 14.	Make two Kempe chain moves (Type A & Type B)
15. & 16.	Make three Kempe chain moves (Type A & Type B)
17. & 18.	Make four Kempe chain moves (Type A & Type B)
19. & 20.	Make five Kempe chain moves (Type A & Type B)
21.	Move a whole timeslot
22.	Swap timeslots
23.	Randomly order timeslots

4 Results

4.1 VNS Results

The most widely used benchmark problems were first presented by Carter et al. [24] for the uncapacitated exam timetabling problem. Over the years, two versions of the datasets have circulated using the same name, and this has caused some confusion in comparing results from different approaches in the literature [54]. We have renamed the datasets and discussed the issue in more detail in [50]. A web site was also set up to include all the different datasets with new distinct names, together with an API evaluation function at
http://www.cs.nott.ac.uk/∼rxq/data.htm. The new naming conventions presented in [50] are used here.

In these problem instances, the number of timeslots is fixed *a priori* and the aim is to spread clashing exams around the timetable as much as possible. Graph density is calculated based on the conflict matrix C for the problem, where $c_{e_{sp}}$ = 1 if exam e_i conflicts with exam e_j (i.e. have students in common), or $c_{e_{sp}}$ = 0 otherwise. It is the ratio between the number of elements of value "1" to the total number of elements in the conflict matrix. 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 results for these problems are reported as the average penalty per student. The key features of these problems are given in Table 2.

Data Set	No. of exams	No. of students	No. of enrolments	Graph Density	No. of periods
car91	682	16925	56877	0.13	35
car92	543	18419	55522	0.14	32
ear83 I	190	1125	8109	0.27	24
hec92 I	81	2823	10632	0.42	18
kfu93	461	5349	25113	0.06	20
lse91	381	2726	10918	0.06	18
sta83 I	139	611	5751	0.14	13
tre92	261	4360	14901	0.18	23
uta92 I	622	21267	58979	0.13	35
ute92	184	2750	11793	0.08	10
you83 I	181	941	6034	0.29	21

Table 2. Characteristics of the uncapacitated benchmark problems [24]

Results from a variety of methods in the literature during the years are presented in Table 3. These different techniques have been successful on these problems but no single approach outperformed 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	Carter et al. (1996) [24]	Di Gaspero & Schaerf (2001) [28]	Caramia et al. (2001) [22]	Burke & Newall (2003) [18]	Merlot et al. (2003) [41]	Yang et al. (2004) [59]
car91 I	7.1	6.2	6.6	4.7	5.1	4.5
car92 I	6.2	5.2	6.0	4.1	4.3	3.93
ear83 I	36.4	45.7	**29.3**	37.05	35.1	33.7
hec92 I	10.8	12.4	**9.2**	11.54	10.6	10.83
kfu93	14.0	18.0	13.8	13.9	13.5	13.82
lse91	10.5	15.5	**9.6**	10.8	10.5	10.35
sta83 I	161.5	160.8	158.2	168.73	157.3	158.35
tre92	9.6	10.0	9.4	8.35	8.4	7.92
uta92 I	3.5	4.2	3.5	3.2	3.5	3.14
ute92	25.8	29.0	**24.4**	25.83	25.1	25.39
yor83 I	41.7	41.0	**36.2**	37.28	37.4	36.35

Table 3. Selected results from the literature on the uncapacitated benchmark problems from [24] (best results given in bold)

Data Set	Eley (2007) [26]	Abdullah et al. (2004) [2]	Burke & Newall (2004) [19]	Qu & Burke (2007) [48]	Burke et al. (2004) [8]	Burke et al. (2007) [16]	Burke & Bykov (2007) [7]
car91 I	5.2	5.2	5.0	5.45	4.8	5.36	**4.42**
car92 I	4.3	4.4	4.3	4.5	4.2	4.53	**3.74**
ear83 I	36.8	34.9	36.2	36.15	35.0	37.92	32.76
hec92 I	11.1	10.3	11.6	11.38	10.8	12.25	10.15
kfu93	14.5	13.5	15.0	14.75	13.7	15.2	**12.96**
lse91	11.3	10.2	11.0	10.85	10.4	11.33	9.83
sta83 I	157.3	159.2	161.9	157.2	159.1	158.19	**157.03**
tre92	8.6	8.4	8.4	8.79	8.3	8.92	**7.75**
uta92 I	3.5	3.6	3.4	3.55	3.4	3.88	**3.06**
ute92	26.4	26.0	27.4	26.68	25.7	28.01	24.82
you83 I	39.4	36.2	40.8	42.2	36.7	41.37	**34.84**

Table 3. (cont.) Selected results from the literature on the uncapacitated benchmark problems from [24] (best results given in bold)

Table 4 compares the results produced by the modified basic VNS algorithm from different initialisations with the best reported solutions from Table 3. Values in italics represent the best solution found between the two initialisation techniques, Basic VNS uses neighbourhoods 1-8 from Table 1, all using random move selection in step 2 (a) of the algorithm in Figure 1. A t-test between the basic VNS with random and greedy initialisations was carried out, for which the p-value for each dataset is also given in Table 4. For most datasets, the p-value for the results of two approaches is less than 0.2, indicating the results between them are siginificantly different.

Data	Best reported	Basic VNS-RI		Basic VNS-GI		p-value
Set	solution	Best	Average	Best	Average	
car91	**4.42**	5.10	5.29	*5.07*	5.24	4.2E-06
car92	**3.74**	4.20	4.39	*4.17*	4.30	2.5E-08
ear83 I	**29.3**	*33.56*	36.33	33.70	36.35	0.01
hec92 I	**9.2**	*10.41*	11.08	-	-	-
kfu93	**12.96**	*13.72*	14.40	13.85	14.54	6.0E-03
lse91	**9.6**	*11.13*	11.70	11.18	11.74	9.4E-08
sta83 I	157.03	***156.86***	157.12	***156.86***	157.15	0.12
tre92	**7.75**	*8.48*	8.88	8.49	8.84	0.22
uta92 I	**3.06**	3.49	3.59	*3.40*	3.49	2.7E-17
ute92	**24.4**	*25.10*	25.94	25.18	26.01	0.11
you83 I	**34.84**	36.80	38.70	*36.77*	38.47	0.067

Table 4. Results from the basic VNS with random (VNS-RI) and greedy (VNS-GI) initialisations

Table 5 gives the results of the Biased VNS when initialised by the two different methods, both use neighbourhoods 1-8 from Table 1 with two additional biased Kempe Chain neighbourhoods, as described in Section 2.2, with a 5% and 20% sample of the exams causing the largest penalty. Again, the best result reported from Table 3 is included for comparison.

The average and best results from 100 runs on a 750MHz Athlon are presented for both the Basic and Biased variants. The stopping condition is 2,500 iterations without improvement. On the smaller datasets, the algorithm generally terminates in the order of 1-2 minutes, whereas runtimes for larger data sets range from 30 minutes to 90 minutes.

It can be seen that the basic implementation of VNS when compared with the other techniques from Table 3, is highly competitive and beats the results each of the other techniques on at least one problem, indicating that it is highly consistent across the range of problems. In particular, it obtained the best results reported in the literature on dataset sta83 I.

Comparing the results of Tables 4 & 5 clearly illustrates an improvement by introducing the two biased neighbourhoods. Results on the two initialisations are mixed although the random initialisation approach slightly outperforms the greedy initialisation across all problems. The p-values from a t-test between these two approaches are

Data Set	Best reported solution	Biased VNS-RI Best	Biased VNS-RI Average	Biased VNS-GI Best	Biased VNS-GI Average	p-value
car91	**4.42**	*5.02*	5.28	5.07	5.12	2.2E-09
car92	**3.74**	4.17	4.34	*4.12*	4.23	3.7E-10
ear83 I	**29.3**	*33.10*	36.00	33.46	35.78	0.29
hec92 I	**9.2**	*10.26*	11.02	-	-	-
kfu93	**12.96**	*13.38*	13.87	*13.38*	14.03	2.89E-04
lse91	**9.6**	*10.66*	11.33	10.93	11.58	4.97E-10
sta83 I	157.03	***156.86***	157.04	***156.86***	157.06	0.92
tre92	**7.75**	*8.35*	8.76	8.39	8.77	0.58
uta92 I	**3.06**	3.47	3.55	*3.39*	3.50	1.77E-04
ute92	**24.4**	*24.86*	25.41	*24.86*	25.43	0.91
yor83 I	**34.84**	36.48	38.33	*36.43*	38.03	1.75E-09

Table 5. Results from the VNS with biased neighbourhoods with random (VNS-RI) and greedy (VNS-GI) initialisations

presented in the table, indicating that the differences between them for 3 datasets are not significant. This implies that VNS has the capability to overcome a seemingly bad initialisation and still produce high quality results.

Table 6 presents the best results obtained by the descent-ascent variation of the Biased VNS [5] compared to those of selected best five techniques in the literature. The approach is ranked the second giving a total penalty across all data sets. This simple sum may be biased heavily by the sta83 I dataset. We further measured across only those 10 datasets besides sta83 I, and VNS is still very competitive (ranked the third) on producing high quality results across all the data sets we tested.

4.2 VNS-GA Results

The main aim of this work is to investigate the potential performance of variants of the VNS algorithm by employing various neighbourhoods rather than to evaluate the GA itself. For this reason, it was deliberately decided not to carry out a deep investigation of GA parameter settings. Initial experiments gave us a broad idea of appropriate parameters in the experiments. Mutation rates of 0.002 and 0.01 were both tested. We use the descent-ascent Biased VNS with the largest degree heuristic with randomisation initialisation (Biased VNS-RI). The VNS-GA hybrid approach that we present is able to intelligently select appropriate neighbourhoods and obtain better results across the different problems that we are addressing.

Table 7 gives the best results found by the VNS-GA algorithm on each data set, compared with the previous best results from the VNS and in the literature (Table 3). The neighbourhoods in VNS which contributed to the best solutions are also given. Across all 11 and 10 well studied problems, the hybrid GA ranked the second (total penalty 302 and 145.1 compared with those in 6) across all the best approaches in

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

Data Set	Abdullah et al. [2]	Caramia et al. [22]	Burke & Bykov [7]	Merlot et al. [41]	Yang et al. [59]	Descent-ascent Biased VNS
car91 I	5.2	6.6	**4.42**	5.1	4.5	4.9
car92 I	4.4	6.0	**3.74**	4.3	3.93	4.1
ear83 I	34.9	**29.3**	32.76	35.1	33.7	33.2
hec92 I	10.3	**9.2**	10.15	10.6	10.83	10.3
kfu93	13.5	13.8	**12.96**	13.5	13.82	13.2
lse91	10.2	**9.6**	9.83	10.35	10.5	10.4
sta83 I	159.2	158.2	157.03	157.3	158.35	**156.9**
tre92	8.4	9.4	**7.75**	8.4	7.92	8.3
uta92 I	3.6	3.5	**3.06**	3.5	3.14	3.3
ute92	26.0	24.4	**24.82**	25.1	25.39	24.9
you83 I	36.2	36.2	**34.84**	37.4	36.35	36.3
Total Penalty (11)	312.2	306.2	**301.36**	310.8	308.28	305.8
Total Penalty (10)	153	148	**144.33**	153.5	149.93	148.9

Table 6. Descent-ascent Biased VNS compared to results from the literature (best results given). “Total Penalty (11)” and “Total Penalty (10)” is the sum of results for all datasets and those except sta83 I

the literature. It can also be easily seen that the VNS-GA improves the descent-ascent Biased VNS with 10 neighbourhoods used previously.

For the same problem, it was observed that a variety in the neighbourhood subsets was evolved from VNS-GA for different seeds of runs due to the fact that some of the neighbourhoods are quite similar to the others. Within the 100 runs, almost every neighbourhood was included on the same dataset, but some were selected far more often than others. Another interesting issue worth of further investigations is to study statistically if similar subsets of neighbourhoods would be evolved for problems with similar characteristics. Based on these analysis knowledge based systems can be built to assist neighbourhood selection within meta-heuristics.

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 process, 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, a further series of tests were undertaken. Table 8 shows the results of running VNS 100 times with the neighbourhood sets suggested in Table 7, compared to those produced by 100 runs of VNS with the full set of 23 neighbourhoods. Average runtimes for the technique are also given in both cases. The results are fairly inconclusive with regard to the merits of using selected neighbourhoods rather than using all 23. In all cases, the difference

Data Set	Best Reported Solution	Best VNS Solution	Best VNS-GA Solution	Neighbourhood Subset For Best Solution
car91 I	**4.42**	4.9	4.6	{1,4-8,11-13,16,17,19-23}
car92 I	**3.74**	4.1	3.9	{1,3-6,8-11,13-17,19-23}
ear83 I	**29.3**	33.1	32.8	{1,3,4,7,11,13-15,17,21-23}
hec92 I	**9.2**	10.3	10.0	{1-4,6,8,10-12,14,16,17,19-22}
kfu93	**12.96**	13.2	13.0	{2,4,6,8-10,12-15,17-20,22}
lse91	**9.6**	10.4	10.0	{2,3,5-8,10,13,15-17,19,20,22,23}
sta83 I	157.03	156.9	**156.9**	Many
tre92	**7.75**	8.3	7.9	{2,4,7-12,15,19,21-23}
uta92 I	**3.06**	3.3	3.2	{1-9,13,18-22}
ute92	24.82	24.9	**24.8**	{1-3,5-10,13-17,19,20,22,23}
you83 I	**34.84**	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

between the two sets of results is small. We have carried out t-test and provided p-values for the comparisons in Table7.

Data Set	VNS with all neighbourhoods			VNS with selected neighbourhoods			p-value
	Best	Average	Time (s)	Best	Average	Time (s)	
car91 I	4.7	**4.9**	2751	4.6	**4.9**	3084	0.36
car92 I	4.0	4.2	1605	3.9	**4.1**	1686	0.027
ear83 I	32.9	34.2	175	32.8	**34.1**	162	0.019
hec92 I	10.2	**10.6**	28	10.0	**10.6**	28	0.11
kfu93	13.2	13.6	633	13.0	**13.4**	673	0.017
lse91	10.1	**10.6**	359	10.0	10.8	345	0.058
tre92	8.3	8.4	244	7.9	**8.2**	218	0.17
uta92 I	3.3	**3.4**	2358	3.2	**3.4**	2040	0.022
ute92	24.9	25.1	67	24.8	**25.0**	73	0.061
you83 I	35.2	**36.4**	124	34.9	36.6	126	0.96

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

The hybrid VNS-GA can provide highly competitive results for a range of problems by adaptively selecting appropriate neighbourhoods. Based on the Biased VNS with all 23 neighbourhoods, the VNS-GA selects the neighbourhoods presented in Table 7, enabling VNS to obtain the best results whereas no other combination of neighbourhoods was able to produce results of that quality.

An analysis of the individual runs of the VNS-GA 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 runtime and solution quality. However, in real

world timetabling, time is not a critical issue as usually the timetables are produced weeks before they are used. This is also the reason why much of the other work in the timetabling literature does not report computational time. In addition, algorithms implemented across different platforms make it impossible to compare the computational time upon the same problems.

5 Conclusions and Future Research Directions

One significant advantage of the Basic Variable Neighbourhood Search (VNS) is that it involves very few parameters apart from the selection of neighbourhoods. 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. The Basic VNS also has a large number of potential improvements. 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 large problems. As mentioned above, this is not a significant issue for real world timetabling, since exams are generally only taken once or twice a year with a fair degree of planning time, it is still an area which deserves future research attention.

The VNS techniques can be adapted more easily than many single neighbourhood techniques and be applied to exam timetabling problems with more complex side constraints. With the increasing complexity of exam timetabling problems in universities, we believe that this general VNS technique can be adapted easily without taking away from the genericity of the method. New neighbourhoods which better reflect constraints in real world environment can be added easily and tested.

The hybrid Genetic Algorithm (VNS-GA) which intelligently selects the neighbourhoods for use within VNS produced very competitive results on benchmark exam timetabling problems which attracted a large research attention in the last decade. The algorithm ranked the second compared with all the best methods in the literature.

A number of methods for improving the performance of the VNS-GA with a particular focus on its consistency have been considered. Most notable amongst these is to run VNS 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. Further tuning of the other parameters of the GA could also significantly improve the consistency of its performance.

At the time of writing, the 2nd International Timetabling Competition (ITC2007) was underway. Authors should refer to [38] for further information about the competition. As part of track one of this competition, a new formulation of the Examination Timetabling Problem was introduced. This formulation models additional constraints which are found in many modern institution. Details of the new formulation can be found at [39]. In relation to the work presented here, the authors intend experimenting with this new formulation and associated datasets released as part of ITC2007. Details of techniques used by competitors can be found in [40].

6 Acknowledgement

We would like to thank the anonymous referees for their helpful and constructive comments in improving this paper.

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. pp 187-210, Springer 2005.
2. S. Abdullah, S. Ahmadi, E. K. Burke and M. Dror. Investigating Ahuja-Orlin's Large Neighbourhood Search Approach for Examination Timetabling. *OR Spectrum*, 29(2), pp 351-372, 2007.
3. S. Abdullah, S. Ahmadi, E.K. Burke, M. Dror and B. McCollum. A Tabu Based Large Neighbourhood Search Methodology for the Capacitated Examination Timetabling Problem. *Journal of Operational Research Society*, 58(11), pp 1494-1502, 2007.
4. R. K. Ahuja, O. Ergun, J. B. Orlin and A. P. Punnen. A Survey of Very Large-scale Neighborhood Search Techniques. *Discrete Applied Mathematics*, 123(1-3), pp 75 - 102, 2002.
5. H. Asmuni, E.K. Burke, J. Garibaldi and B. McCollum. A Novel Fuzzy Approach to Evaluate the Quality of Examination Timetabling. In: E.K. Burke and H. Rudova (eds). (2006). Proceedings of the 6th International Conference on the Practice and Theory of Automated Timetabling. pp 82-102. 2006. ISBN 80-210-3726-1.
6. B. Bilgin, E. Ozcan and E.E. Korkmaz. An Experimental Study on Hyper-heuristics and Exam Timetabling. In: Proceedings of the 6th International Conference on the Practice and Theory of Automated Timetabling, pp 123-140, 2006. ISBN 80-210-3726-1.
7. E.K. Burke and Y. Bykov. Solving Exam Timetabling Problems with the Flex-Deluge Algorithm. In: Proceedings of the 6th International Conference on the Practice and Theory of Automated Timetabling, pp 370-372, 2006. ISBN 80-210-3726-1.
8. E.K. Burke, Y. Bykov, J. P. Newall and S. Petrovic. A Time-predefined Local Search Approach to Exam Timetabling Problems. *IIE Transactions*, 36(6), pp 509-528. 2004.
9. E.K. Burke, Y. Bykov and S. Petrovic. A Multicriteria Approach to Examination Timetabling. In: E.K. Burke and W. Erben (eds). (2001). Practice and Theory of Automated Timetabling: Selected Papers from the 3rd International Conference. Springer Lecture Notes in Computer Science, vol. 2079. pp 118-131. 2001.
10. E.K. Burke, M. Dror, S. Petrovic and R. Qu. Hybrid Graph Heuristics in Hyper-heuristics Applied to Exam Timetabling Problems. In: B.L. Golden, S. Raghavan and E.A. Wasil (eds.): The Next Wave in Computing, Optimization, and Decision Technologies. pp 79-91. Springer, Maryland, Jan 2005.
11. E.K. Burke, D.G. Elliman, P.H. Ford and R.F. Weare. Examination Timetabling in British Universities: a Survey. In: *Practice and Theory of Automated Timetabling I*. Springer Lecture Notes in Computer Science, vol. 1153, pp 76-90. 1996.
12. 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.
13. E.K. Burke, K. Jackson, J. H. Kingston and R. Weare. Automated University Timetabling: The State of the Art. *The computer journal*, 40(9), pp 565-571, 1997.
14. E.K. Burke, J. Kingston and D. de Werra. Applications to Timetabling. In: J. Gross and J. Yellen (eds.), *Handbook of Graph Theory*. Chapman Hall/CRC Press. pp 445-474, 2004.

15. E.K. Burke, S. Petrovic and R. Qu. Case Based Heuristic Selection for Timetabling Problems. *Journal of Scheduling*, 9, pp 115-132, 2006.
16. E.K. Burke, B. McCollum, A. Meisels, S. Petrovic and R. Qu. A Graph-Based Hyper Heuristic for Timetabling Problems. *European Journal of Operational Research*, 176, pp 177-192, 2007.
17. E.K. Burke, J.P. Newall and R.F. Weare. A Memetic Algorithm for University Exam Timetabling. In: E.K. Burke and P. Ross (eds). (1996). Practice and Theory of Automated Timetabling: Selected Papers from the 1st International Conference. Springer Lecture Notes in Computer Science, vol. 1153. pp 241-250. 1996.
18. E.K. Burke and J.P. Newall. Enhancing Timetable Solutions with Local Search Methods. In: *Practice and theory of automated timetabling II*, Springer Lecture Notes in Computer Science, vol. 2740. pp 195-206. 2003.
19. E.K. Burke and J.P. Newall. Solving Examination Timetabling Problems Through Adaptation of Heuristic Orderings. *Annals of Operations Research*, 129, pp 107-134. 2004.
20. E.K. Burke and J.P. Newall. A Multi-stage Evolutionary Algorithm for the Timetable Problem. *IEEE Transactions on Evolutionary Computation*, 3(1), pp 63-74, 1999.
21. E.K. Burke and S. Petrovic. Recent Research Directions in Automated Timetabling. *European Journal of Operational Research*, 140(2), pp 266-280. 2002.
22. 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)*. Springer Lecture Notes in Computer Science, vol 1982, pp 230-241. 2001.
23. M.W. Carter. A survey of practical applications of examination timetabling algorithms. *Operations Research*, 34(2): 193-202.
24. M.W. Carter and G. Laporte. Recent Developments in Practical Examination Timetabling. In: *Practice and Theory of Automated Timetabling I*. Springer Lecture Notes in Computer Science, vol. 1153. pp 373-383. 1996.
25. M.W. Carter, G. Laporte and S.Y. Lee. Examination Timetabling: Algorithmic Strategies and Applications. *Journal of the Operational Research Society*, 47(3), pp 373-383. 1996.
26. S. Casey and J. Thompson. GRASPing the Examination Scheduling Problem. In: *Practice and Theory of Automated Timetabling IV*, Springer Lecture Notes in Computer Science, vol. 2740, pp 232-244. 2003.
27. M. Eley. Ant Algorithms for the Exam Timetabling Problem. In: E.K. Burke and H. Rudova (eds). Practice and Theory of Automated Timetabling: Selected Papers from the 6th International Conference. Springer Lecture Notes in Computer Science, vol. 372-391.
28. L. Di Gaspero. Recolour, Shake and Kick: A Recipe for the Examination Timetabling Problem. In: *Proceedings of the Fourth International Conference on the Practice and Theory of Automated Timetabling*, pp 404-407. Gent, Belgium, August 2002. ISBN 90-806096-1-7.
29. L. Di Gaspero and A. Schaerf. Tabu search techniques for examination timetabling. In: *Practice and Theory of Automated Timetabling III*. Springer Lecture Notes in Computer Science, vol. 2079. pp 104-117. 2001.
30. A. Gaw, P. Rattadilok and R.S. Kwan. Distributed Choice Function Hyperheuristics for Timetabling and Scheduling. In: *Practice and Theory of Automated Timetabling V*, Springer Lecture Notes in Computer Science, vol. 3616, pp 51-67, 2005.
31. 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. pp 165-182, Springer 2005.
32. F. Glover and G. Kochenberger. *Handbook of Metaheuristics*, Kluwer, 2003.
33. L. Han and G. Kendall. Guided Operators for a Hyper-heuristic Genetic Algorithm. In: *Proceedings of the 16th Australian Conference on Artificial Intelligence*, Springer Lecture Notes in Computer Science, vol. 2903, pp 807-820. 2003.

34. P. Hansen and N. Mladenović. Variable Neighbourhood Search: Principles and Applications. *European Journal of Operational Research*, 130, pp 449-467. 2001.
35. 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.
36. G. Kendall and N.M. Hussin. An Investigation of a Tabu Search Based Hyperheuristic for Examination Timetabling. In: Kendall G., Burke E. and Petrovic S. (eds), Selected papers from Multidisciplinary Scheduling; Theory and Applications, pp 309-328, 2005.
37. M.R. Malim, A.T. Khader and A. Mustafa. Artificial Immune Algorithms for University Timetabling. In: E.K. Burke and H. Rudova (eds.): Proceedings of the 6th International Conference on Practice and Theory of Automated Timetabling, pp 234-245. August 2006, Brno, Czech Republic. ISBN 80-210-3726-1.
38. B.G.C. McCollum. Bridging the Gap between Research and Practice. In: Proceedings of the 6th International Conference on the Practice and Theory of Automated Timetabling, Brno, 30th Aug - 1st Sep 2006, pp 15-35. ISBN 80-210-3726-1.
39. B.G.C. McCollum, A. Schaerf, B. Paechter, P. McMullan, R. Lewis, A. Parkes, L. Di Gaspero, R. Qu, E.K. Burke. Setting The Research Agenda in Automated Timetabling: The Second International Timetabling Competition, To appear in *INFORMS Journal on Computing*. doi 10.1287/ijoc.1090.0320
40. B.G.C. McCollum, P. McMullan, E.K. Burke, A.J. Parkes, R. Qu. A New Model for Automated Examination Timetabling, Accepted to post PATAT08 *Annals of OR*, to be published in 2010
41. B.G.C. McCollum, P.J. McMullan, A.J. Parkes, E.K. Burke, S. Abdullah. An Extended Great Deluge Approach to the Examination Timetabling Problem Proceedings of The 4th Multidisciplinary International Conference on Scheduling: Theory and Applications, Dublin, August 2009, pp 424-434.
42. L.T.G. Merlot, N. Boland, B.D. Hughes and P.J. Stuckey. A Hybrid Algorithm for the Examination Timetabling Problem. In: *Practice and Theory of Automated Timetabling IV*, Springer Lecture Notes in Computer Science, vol. 2740. pp 207-231. 2003.
43. N. Mladenović and P. Hansen. Variable Neighbourhood Search. *Computers & Operations Research*, 24(11), pp 1097-1100. 1997.
44. Frank Hutter, D. Babic, H.H. Hoos and A.J. Hu. Boosting Verification by Automatic Tuning of Decision Procedures. Proceedings of the Formal Methods in Computer Aided Design. pp 27-34, 2007. IEEE Computer Society Washington, DC, USA.
45. J.B. Orlin. Very Large-Scale Neighborhood Search Techniques in Timetabling Problems. In: Proceedings of the 6th International Conference on the Practice and Theory of Automated Timetabling, Brno, 30th Aug - 1st Sep 2006, pp 36-52. ISBN 80-210-3726-1.
46. E. Ozcan, N. Bilgin and E.E. Korkmaz. Hill Climbers and Mutational Heuristics in Hyperheuristics. In: Proceedings of the 9th International Conference on Parallel Problem Solving from Nature, pp 202-211, 2006.
47. 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.
48. R. Qu and E.K. Burke. Hybridisations within a Graph Based Hyper-heuristic Framework for University Timetabling Problems. *Journal of Operational Research Society*, 60, pp 1273-1285, 2009.
49. R. Qu and E.K. Burke. Adaptive Decomposition and Construction for Examination Timetabling Problems. Proceedings of the 3rd Multidisciplinary International Scheduling: Theory and Applications, 418-425, Aug, 2007, Paris, France.
50. R. Qu, E.K. Burke and B.G.C. McCollum. Adaptive Automated Construction of Hybrid Heuristics for Exam Timetabling and Graph Colouring Problems. *European Journal of Operational Research*, 198(2), 392-404, 2009,

51. R. Qu, E.K. Burke, B. McCollum, L.T.G. Merlot, and S.Y. Lee. A Survey of Search Methodologies and Automated System Development for Examination Timetabling. *Journal of Scheduling*, 12(1): pp 55-89, 2009.
52. R. Qu, F. He and E.K. Burke. Hybridizing Integer Programming Models with an Adaptive Decomposition Approach for Exam Timetabling Problems. In: The 4th Multidisciplinary International Scheduling: Theory and Applications, pp. 435-446, 10-12 August 2009, Dublin, Ireland.
53. P. Ross, J.G. Marin-Blazquez and E. Hart. Hyper-heuristics Applied to Class and Exam Timetabling Problems. In: Proceedings of the 2004 Congress on Evolutionary Computation (CEC2004), pp 1691-1698. 2004.
54. A. Schaerf. A Survey of Automated Timetabling. *Artificial Intelligence Review*, 13(2), pp 87-127. 1999.
55. A. Schaerf. Measurability and Reproducibility in Timetabling Research: State-of-the-Art and Discussion. In: Proceedings of the 6th International Conference on the Practice and Theory of Automated Timetabling, Brno, 30th Aug - 1st Sep 2006, pp 53-63. ISBN 80-210-3726-1.
56. 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*, pp 635-642. Orlando, Florida, July 13-17, 1999.
57. J. Thompson and K. Dowsland. Variants of Simulated Annealing for the Examination Timetabling Problem. *Annals of Operational Research*, 63, pp 105-128. 1996.
58. J. Thompson and K. Dowsland. A Robust Simulated Annealing Based Examination Timetabling System. *Computers & Operations Research*, 25, pp 637-648. 1998.
59. G.M. White, B.S. Xie and S. Zonjic. Using Tabu Search with Longerterm Memory and Relaxation to Create Examination Timetables. *European Journal of Operational Research*, 153(16), pp 80-91, 2004.
60. Y. Yang and S. Petrovic. A Novel Similarity Measure for Heuristic Selection in Examination Timetabling. In: *Practice and Theory of Automated Timetabling V*. Springer Lecture Notes in Computer Science, vol. 3616. pp 377-396. 2005.