European Journal of Operational Research xxx (2008) xxx–xxx

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Discrete Optimization

A study of heuristic combinations for hyper-heuristic systems for the uncapacitated examination timetabling problem

N. Pillay [a,*], W. Banzhaf [b]

[a] School of Computer Science, University of KwaZulu-Natal, King Edward Road, Pietermaritzburg Campus, Pietermaritzburg, KwaZulu-Natal 3201, South Africa
[b] Department of Computer Science, Memorial University of Newfoundland, St. John's, NL, Canada A1B 3X5

A R T I C L E   I N F O

Article history:
Received 12 January 2007
Accepted 18 July 2008
Available online xxxx

Keywords:
Timetabling
Heuristics

A B S T R A C T

Research in the domain of examination timetabling is moving towards developing methods that generalise well over a range of problems. This is achieved by implementing hyper-heuristic systems to find the best heuristic or heuristic combination to allocate examinations when constructing a timetable for a problem. Heuristic combinations usually take the form of a list of low-level heuristics that are applied sequentially. This study proposes an alternative representation for heuristic combinations, namely, a hierarchical combination of heuristics. Furthermore, the heuristics in each combination are applied simultaneously rather than sequentially. The study also introduces a new low-level heuristic, namely, highest cost. A set of heuristic combinations of this format have been tested on the 13 Carter benchmarks. The quality of the examination timetables induced using these combinations are comparable to, and in some cases better than, those produced by hyper-heuristic systems combining and applying heuristic combinations sequentially.

1. Introduction

Numerous methodologies including graph-based techniques, constraint-based methods, local search, ant algorithms and meta-heuristics such as simulated annealing and genetic algorithms, have been applied to the examination timetabling problem (Qu et al., 2008). While previous attempts in this domain have been aimed at developing techniques that produce the best results for one or more data sets, currently research is being directed at creating methods that generalise well so as to promote the movement of systems out of the research lab and into educational institutions (McCollum, 2007). This has led to the development of hyper-heuristics systems (Burke et al., 2003; Ross, 2005) which produce heuristics or a combination of heuristics to be used in the generation of examination timetables.

A combination of heuristics usually takes the form of a list of low-level heuristics. The study presented in this paper concentrates on construction heuristics. In this context each heuristic is applied in order to choose one or more examinations to schedule next during timetable construction. The study presented in this paper tests the effect of combining low-level heuristics hierarchically and applying them simultaneously instead of sequentially. The low-level heuristics comprising each combination includes the standard graph heuristics, namely, largest degree, largest enrolment, largest weighted degree and saturation degree, and a new heuristic, highest cost. A set of four heuristic combinations constructed and applied according to this new approach were used to generate solutions to the 13 Carter benchmark problems. The quality of the timetables produced by these combinations were found to be just as good as and in some cases better than the timetables constructed using heuristic combinations induced by hyper-heuristic systems representing and applying heuristic combinations sequentially. Thus, the main contribution of this study is a new approach to combining and applying low-level heuristics that can be employed by hyper-heuristic systems and a new low-level heuristic, highest cost.

The following section defines the examination timetabling problem. Section 3 provides an overview of examination timetabling research and discusses some of the studies that have made a contribution to this field. Previous studies employing the use of heuristic combinations in the sequential construction of solutions to the examination timetabling problem are summarised in Section 4. Section 5 describes the four heuristic combinations and the overall system implemented to generate solutions to the examination timetabling problem. The methodology employed to evaluate these four heuristic combinations is presented in Section 6. Section 7 discusses the performance of the four heuristic combinations on the set of 13 Carter benchmarks. The results obtained by the four combinations on the Carter benchmark set is compared to that of other heuristic combinations produced by hyper-heuristic systems, the results of some of the earlier examination timetabling problem studies and methods that have obtained the best results in this

* Corresponding author. Tel.: +27 33 2605644; fax: +27 33 2605648.
E-mail addresses: pillayn32@ukzn.ac.za (N. Pillay), banzhaf@cs.mun.ca (W. Banzhaf).

0377-2217/$ - see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.ejor.2008.07.023

field. A summary of the findings of this study and future extensions of this research is presented in Section 8.

2. The examination timetabling problem

The examination timetabling problem involves scheduling a given set of exams in a given number of examination sessions so as not to violate any *hard constraints* and to minimise the *soft constraints* violated (Qu et al., 2008).

Hard constraints are those constraints that must be met by the timetable in order for the timetable to be feasible. For example, there should not be any clashes, i.e. students should not be required to write two exams at the same time. Soft constraints are constraints that we would like the timetable to satisfy, but can be broken if necessary. For example, students' examinations must be well spaced over the timetable; examinations with a large number of students must be scheduled early in the examination timetable. It is highly unlikely that all the soft constraints will be met by a timetable as these are usually contradictory. Timetabling systems attempt to minimise the number of soft constraints violated. It is evident from the literature (Burke et al., 1996; Burke and Petrovic, 2002; McCollum, 2007; Qu et al., 2008; Schaerf and Di Gaspero, 2006; Ranson and Ahmadh, 2007) that the hard and soft constraints differ considerably from one examination timetabling problem to the next and are institution dependant. For example, some versions of the problem may require the number of examination sessions used to be minimised; some examinations may have to be scheduled simultaneously or after other exams; special requirements of students, such as religious requests, may need to be catered for.

An examination timetabling problem may be *capacitated* or *uncapacitated*. In the capacitated version of the problem venue allocation is taken into consideration and the number of students writing examinations in a particular venue and session must not exceed the capacity of the venue. This is treated as a hard constraint.

The research presented in this paper focuses on the uncapacitated examination timetabling problem. The following section provides an overview of previous studies that have made a contribution to the domain of uncapacitated examination timetabling.

3. Previous work

A vast amount of research has been conducted in the domain of examination timetabling with various methodologies being applied to the uncapacitated examination timetabling problem in an attempt to produce better quality timetables. Burke and Petrovic (2002), McCollum (2007) and Qu et al. (2008) provide an overview of the research in this field. This section describes those methods cited in the literature as making a contribution to the field.

Research in this domain was initiated by Carter et al. (1996) with the EXAMINE system which was used to generate timetables for real world problems. The EXAMINE system basically employed a sequential construction method to allocate examinations. The examinations were firstly ordered according to their difficulty using one of five low-level heuristics, namely, largest degree, saturation degree, largest weighted degree, largest enrolment and random ordering. The examinations were allocated sequentially in order. In the case of conflicts, i.e. an allocation would result in a clash, backtracking was performed to reallocate examinations so as to remove the clash. In some cases this process resulted in some exams being moved to a waiting list and reallocated at a later stage. EXAMINE was successfully applied to a set of randomly generated test problems and 13 real world problems. These 13 real world problems have become known as the Carter benchmarks and new developments in the field are usually tested on this set of problems. The saturation degree heuristic was found to produce the best computational time (i.e. the time taken to find a feasible solution), while the largest degree followed by the largest enrolment produced the best solution costs (i.e. cost of the soft constraint).

One of the earlier meta-heuristics methods applied to the examination timetabling problem was the Tabu search used by Di Gaspero and Schaerf (2001) to explore a space of graphs with the nodes representing the examinations and edges joining conflicting exams. Node weights specified the number of students writing the examination and edge weights the number of students involved in a particular clash. The Tabu search was successfully applied to 12 of the Carter benchmarks.

The system implemented by Caramia et al. (2008) appears to have made a major contribution to this domain and has produced the best quality timetables for a number of the Carter benchmarks. The system is composed of a greedy scheduler, a penalty decreaser and a penalty trader. The greedy scheduler allocates examinations according to their priority value, namely, the number of conflicts, to a clash-free timeslot that produces the minimum soft constraint cost. The soft constraint costs are further reduced by the penalty decreaser and the penalty trader. The penalty trader introduces a new timeslot for this purpose.

Casey and Thompson (2003) have used the GRASP system to generate solutions to the Carter benchmarks. This system takes a two-phased approach to the examination timetabling problem. The first phase produces a feasible timetable by allocating the examinations, which are sorted in order according to a low-level heuristic, to a clash-free slot. In the case that a clash-free timeslot cannot be found, backtracking is performed to rearrange the scheduled examinations so as to remove the clash. During the second phase simulated annealing is used to optimise the soft constraint costs.

Merlot et al. (2003) also take a hybrid approach to the examination timetabling problem. The system uses constraint programming to create an initial solution. This initial solution is refined by firstly applying simulated annealing and then hill-climbing. At the end of the hill-climbing phase a greedy heuristic is used to schedule any unallocated examinations. This system performed comparably to other methodologies applied to the Carter benchmarks.

More recent studies in the field include the three variations of very large-scale neighbourhood search proposed by Meyers and Orlin (2007) and the evaluation of a genetic algorithm using a linear linkage encoding representation by Ulker et al. (2007). Of the later studies those that are most relevant to the research presented in this paper are the Flex-Deluge algorithm implemented by Burke and Bykov (2006a), the variable neighbourhood search and genetic algorithm hybrid system tested by Burke et al. (2006b), the application of ant colonisation to the examination timetabling problem by Eley (2007) and the Ahuja–Orlin large neighbourhood search employed by Abdullah et al. (2007).

The Flex-Deluge algorithm is a variation of the Great Deluge algorithm which incorporates a form of hill-climbing. This system has been successfully applied to the Carter benchmarks and has outperformed other methodologies on some of the data sets.

Burke et al. (2006b) have implemented a variation of the variable neighbourhood search (VNS) which uses a genetic algorithm to select the most suitable subset of neighbourhoods, from a set of 23 such neighbourhoods, for the examination timetabling problem at hand. The VNS is applied to a feasible timetable which is constructed by allocating examinations in order, according to the largest degree heuristic, to a feasible period. The main aim behind using the genetic algorithm is to produce a system that can generalise well over a range of examination timetabling problems. This system was applied to the Carter benchmarks and produced the best results cited thus far for two of the datasets.

Eley (2007) applies the MMAS (Max–min ant system) ant colonisation algorithm to the uncapacitated examination timetabling problem. The first phase of the algorithm is comprised of a number of iterations during which the ants create feasible timetables. During this phase examinations are scheduled according to the lowest saturation degree. Hill-climbing is then used to further improve the quality of the best timetable found in phase 1. The results produced by this system are comparative to other methods applied to the Carter benchmarks.

Abdullah et al. (2007) employ a variation of the Ahuja–Orlin algorithm to find solutions to the Carter benchmarks. The algorithm begins by dividing the examinations, according to the saturation degree heuristic, into cells with each cell corresponding to a timeslot. The cells form input to a cyclic exchange process to produce a very large-scale neighbourhood. A network flow optimisation technique, which employs the use of an improvement graph to identify the most appropriate neighbourhood to move to, is used to search the very large-scale neighbourhood. The system performed well on the Carter benchmarks although in some cases the runtime was high.

The studies discussed thus far have focussed on improving the quality of timetables generated for the Carter benchmarks. A more recent direction of research in the domain of examination timetabling is aimed at developing systems that generalise well over a range of problems rather than producing the best results for a few problems. This has led to the development of hyper-heuristic systems for the examination timetabling problem. Burke et al. (2003) and Ross (2005) provide an overview of hyper-heuristics in general. Hyper-heuristic systems automatically produce a problem-specific heuristic or a combination of low-level heuristics, which is used to allocate examinations in a particular order. The heuristics comprising a hyper-heuristic have varied from low-level heuristics for timetable construction, improvement heuristics or neighbourhood move operators (Kendall and Hussin, 2005), strategies for heuristic selection and move acceptance (Bilgin et al., 2007) and hill-climbers (Ersoy et al., 2007). The study presented in this paper focuses on construction heuristics and hence the discussion that follows is limited to those studies that use constructive heuristics or a combination of constructive and other types of heuristics. It is evident from the literature surveyed that hyper-heuristic systems employ one of two approaches in deciding on which constructive heuristic to apply next. The first either identifies or adapts an existing heuristic to be applied at each stage of the construction process, while the second approach optimises a search space of heuristic combinations, i.e. lists of low-level heuristics.

Studies employing the first approach include Burke et al. (2006c), Burke and Newall (2004), Yang and Petrovic (2004), Kendall and Hussin (2005) and Ross et al. (2004). In the study conducted by Burke and Newall heuristic values are iteratively adapted based on their performance. Initially, one of five low-level heuristics is used to order the examinations. The initial heuristic value assigned to each examination is modified on the next iteration if either the examination could not be scheduled as a result of a clash or the soft constraint costs associated with scheduling the exam exceeds a preset limit. Burke et al. and Yang and Petrovic use case-based reasoning to determine which heuristic to apply to order examinations during each stage of the construction process. A case base of previously solved examination timetabling problems is maintained. When solving a new examination timetabling problem a similarity measure is used to decide which heuristic to apply next. The heuristic is used as is or adapted if necessary. Yang and Petrovic use a Tabu search to explore the space of similarity measures and the Great Deluge algorithm to search the timetable space. Instead of a case-based approach Kendall and Hussin use a Tabu search and Ross et al. apply a steady state genetic algorithm to decide on which heuristic to apply next. In the study conducted by Kendall and Hussin hyper-heuristics include both construction heuristics and neighbourhood heuristics.

The study presented in this paper focuses on the second approach taken by hyper-heuristic systems, i.e. the generation of combinations of low-level construction heuristics. The following section provides an overview of previous studies investigating the generation of heuristic combinations.

4. Heuristic combinations and the examination timetabling problem

Hyper-heuristic systems producing combinations of low-level heuristics employ an optimisation technique to search the space of heuristic combinations. The search is driven by the cost of the timetable constructed using each combination. This section provides an overview of previous work producing heuristic combinations of low-level heuristics.

The idea of optimising a search space of heuristic combinations to find solutions to the examination timetabling problem was introduced by Burke et al. (2005) who employed a Tabu search to explore the space of combinations of the largest degree and saturation degree heuristics. The initial list is composed of only the saturation degree heuristic. The Tabu search is then applied to this initial list to generate a heuristic combination that produces the best quality timetable. At each stage of the process the performance of the heuristic combination is evaluated by assessing the quality of the timetable constructed using the combination. The timetable is constructed by applying each heuristic in the list sequentially to schedule three examinations. The system was applied to a set of randomly generated problems and four of the Carter benchmarks.

Asmuni et al. (2004) implement a fuzzy logic expert system to combine heuristics. The fuzzy expert system uses a form of exhaustive search to fine tune the fuzzy terms. The fuzzy logic system produces combinations of two of the following heuristics: largest degree, saturation degree and largest enrolment. The combination output by the system is in the form of a fuzzy weight indicating the difficulty of scheduling the particular examination. Thus, each heuristic in the combination is applied simultaneously as a single value. A sequential construction method sorts the examinations in decreasing order based on their fuzzy weight values and the examinations are allocated sequentially in this order. Each exam is allocated to the period which produces the minimum penalty. In the case of clashes deallocation and reallocation of examinations is performed. The fuzzy system was tested on 12 of the Carter benchmarks. The combination of two heuristics was found to produce better results than using single heuristics to order examinations. The best results were produced by a combination of the saturation degree and largest enrolment heuristics. This combination produced the best results for 11 of the 12 benchmarks. The performance of this system was also found to be comparable to that of other methodologies applied to the Carter benchmarks.

Qu and Burke (2005) apply a hybrid variable neighbourhood search (VNS) to the heuristic space to identify heuristic lists that produce high quality examination timetables. Each list is essentially a combination of different low-level heuristics. The low-level heuristics used are colour degree, largest degree, largest enrolment, largest weighted degree, saturation degree and random ordering. The VNS uses one of two neighbourhood sets. The first is obtained by randomly changing two to five heuristics in different parts of the list (VNS1). The second involves randomly changing two to five heuristics in sequence, i.e. a block of heuristics (VNS2). The heuristic list output by the VNS is used to construct a timetable by applying each heuristic sequentially to order the examinations not yet scheduled. The most difficult exam is allocated to the period with the minimum penalty and the next heuristic in the list is applied to the remaining examinations and the process is repeated. The system was tested on 11 of the Carter benchmarks. The results obtained are comparable to that cited in the literature.

Burke et al. (2007) employ a similar approach to induce sequences of low level heuristics. In this study a Tabu search is used instead of a VNS to search the heuristic space for a heuristic list that produces the best quality timetable. Each list is comprised of two or more of the following low-level heuristics: least saturation degree, largest colour degree, largest degree, largest weighted degree, largest enrolment and random ordering. Each heuristic in the list is used to schedule two examinations. The initial heuristic list for all experiments is composed of only the saturation degree heuristic. The system was applied to 11 of the Carter benchmarks. Three runs were performed for each problem. The performance of this system comes close to those producing the best results in this domain.

The following section describes the heuristic combinations and the overall system implemented in the study presented in this paper.

5. Heuristic combinations and overall system

The main aim of the study presented in this paper is to test a new approach that can be used by hyper-heuristic systems to combine and apply low-level heuristics. This section describes the new approach and the overall system implemented to test it. Note that a hyper-heuristic system has not been implemented to generate heuristic combinations of this format. The heuristic combinations are tested individually. If this study reveals that the performance of these combinations created using the new approach show potential, future work will focus on implementing a hyper-heuristic system employing an optimisation method, such as genetic programming, to explore the space of heuristic combinations of this format.

This study differs from previous work in that the heuristics are combined hierarchically into heuristic combinations and the heuristics are applied simultaneously. Combinations of the following low-level heuristics are used in this study:

- Largest degree (LD) – The number of conflicts an examination is involved in. The exam with the largest number of conflicts is scheduled first.
- Largest enrolment (LE) – The number of students enrolled for the course. The examination with the largest student enrolment is scheduled first.
- Largest weighted degree (LWD) – The examination with the largest number of conflicting students is scheduled first.
- Saturation degree (SD) – The number of remaining periods that an exam can be allocated to without causing a clash, i.e. the number of feasible periods. A smaller value indicates an examination that is more difficult to schedule.
- Highest cost (HC) – The soft constraint cost of scheduling an examination given the current state of the timetable. The cost of scheduling each examination is calculated using the function in Fig. 1. The *weight* function used in Fig. 1 is defined in Fig. 2. The examination with the highest cost is scheduled first.

The heuristics LD, LE and LWD can be calculated prior to the construction process and remain static throughout the process. The values of SD and HC are dependant on the current status of the timetable and have to be recalculated after each exam allocation.

In this study each combination consists of p primary heuristics and a secondary heuristic, with $p \geqslant 2$. When ordering examinations the values of all p primary heuristics in the combination are compared for each examination, i.e. a vector or Pareto comparison is performed. The p heuristics are combined using logical operators. For example, suppose that SD and HC are chosen as primary heuristics. A comparison using these heuristics is illustrated in Fig. 3. One of the primary heuristics is defined as the **priority** heuristic. This heuristic is given priority in conflict situations. For example, in Fig. 3 SD is the priority heuristic. Thus, in the case where $e1.sd < e2.sd$ and $e1.hc < e2.hc$, $e1$ is scheduled first, i.e. SD is given priority over HC. In this example a secondary heuristic is not used.

A secondary heuristic is used to break ties during the comparison process. If a secondary heuristic is not specified examinations involved in a tie are scheduled sequentially. Fig. 4 lists an example of a Pareto comparison for a combination of primary heuristics SD and HC and secondary heuristic LD. Note that SD is the priority heuristic in this combination. The use of a secondary heuristic and a priority heuristic allows for a hierarchical rather than a sequential comparison.

The main aim of this study is to test this new approach to combining heuristics. If the results obtained are promising future work will investigate automatically generating the different combinations of primary, secondary and priority heuristics and inducing the most effective heuristic combination for a particular problem domain by applying genetic programming to search the space of heuristic combinations. Preliminary experimentation with the different low-level heuristics have revealed that SD reduces the computational time associated with inducing feasible timetables while HC helps to minimise the soft constraint cost. Although it may take longer to compute the SD heuristic compared to other heuristics, the SD results in a feasible solution being found quicker than the other graph heuristics.

Thus, it was decided to study the following heuristic combinations to assess the effectiveness of the new approach:

- SD–HC – SD and HC are defined as the primary heuristics, with SD defined as the priority heuristic. Secondary heuristics are not used in this case. In the case of a tie, exams are allocated sequentially.
- SD–HC(LD) – SD and HC are defined as the primary heuristics, with SD defined as the priority heuristic. LD is used as the secondary heuristic.
- SD–HC(LWD) – SD and HC are defined as the primary heuristics, with SD defined as the priority heuristic. LWD is defined as the secondary heuristic.
- SD–HC (LE) – SD and HC are defined as the primary heuristics, with SD defined as the priority heuristic. LE is used as the secondary heuristic.

Each timetable is constructed by firstly applying the heuristic combination to order the examinations using a Pareto comparison of the primary heuristics in the combination and secondary heuristic (if defined) to break ties. The overall process is depicted in Fig. 5.

Each examination is allocated to the minimum penalty period, i.e. a clash-free period that produces the minimum soft constraint cost. If there is more than one such period, the period is randomly chosen from the set of minimum cost periods. The algorithm for determining the best period is illustrated in Fig. 6 with the corresponding cost function defined in Fig. 7. The *weight* function referred to in Fig. 7 is the same function defined in Fig. 2. Note that there are no mechanisms built into the system to ensure that a feasible timetable is constructed.

Please cite this article in press as: Pillay, N., Banzhaf, W., A study of heuristic combinations for hyper-heuristic systems ..., European Journal of Operational Research (2008), doi:10.1016/j.ejor.2008.07.023

```
function calc_hc( exam e)
begin
  hc = 0
  for each exam e_j other than e
  begin
    if(e_j has students in common with e and e_j has already been scheduled)
    begin
      for each period p
      begin
        dist = the absolute value of the distance between p and the period e_j
               has been allocated to
        cost = weight(dist) * the number of students common to both exams
        hc= hc + cost
      endfor
    endfor
    return hc
end
```

Fig. 1. Pseudo code for calculating the HC heuristic for examination *e*.

```
procedure weight (dist d)
begin
  case of d
    1: return 16
    2: return 8
    3: return 4
    4: return 2
    5: return 1
    default : return 0
  endcase
end
```

Fig. 2. Weight function.

```
compare(exam e1, exam e2)
begin
  if (e1.sd == e2.sd and e1.hc != e2.hc)
    if (e1.hc > e2.hc)
     schedule e1 first
    else
     schedule e2 first
  else if( e1.hc == e2.hc and e1.sd != e2.sd)
    if(e1.sd > e2.sd)
      schedule e2 first
    else
      schedule e1 first
  else if (e1.sd < e2.sd and e1.hc > e2.hc)
    schedule e1 first
  else if (e1.sd > e2.sd and e1.hc < e2.hc)
    schedule e2 first
  else if (e1.sd < e2.sd and e1.hc < e2.hc)
    schedule e1 first
  else if(e1.sd > e2.sd and e1.hc > e2.hc)
    schedulee2 first
  else
    either examination can be scheduled first
end
```

Fig. 3. A Pareto comparison for the combination of SD and HC, with SD defined as the priority heuristic.

```
compare(exam e1, exam e2)
begin
  if (e1.sd == e2.sd and e1.hc != e2.hc)
    if (e1.hc > e2.hc)
     schedule e1 first
    else
     schedule e2 first
  else if( e1.hc == e2.hc and e1.sd != e2.sd)
    if(e1.sd > e2.sd)
      schedule e2 first
    else
      schedule e1 first
  else if (e1.sd < e2.sd and e1.hc > e2.hc)
    schedule e1 first
  else if (e1.sd > e2.sd and e1.hc < e2.hc)
    schedule e2 first
  else if (e1.sd < e2.sd and e1.hc < e2.hc)
    schedule e1 first
  else if(e1.sd > e2.sd and e1.hc > e2.hc)
    schedulee2 first
  else
    if(e1.ld > e2.ld)
      schedule e1 first
    else if(e1.ld < e2.ld)
      schedule e2 first
    else
     either examination can be scheduled first
end
```

Fig. 4. A Pareto comparison for the combination with primary heuristics SD and HC, SD is the priority heuristic and LD the secondary heuristic.

Once a timetable is constructed it is not optimised to further reduce the soft constraint cost. If more than one minimum cost period exists for a particular examination, the period the examination is allocated to is randomly chosen from the list of possible candidates. This introduces an element of selection noise into the process, i.e. if a different period is chosen this could lead to a better quality timetable being constructed. To overcome this problem n distinct timetables are constructed and the timetable with the

```
1. Order the examinations according to the heuristic
   combination
2. While there are examinations that have not been
   scheduled
     a) Allocate the most difficult exam to the
        minimum penalty period.
     b) Order the remaining examinations according
        to the heuristic combination.
   EndWhile
```

Fig. 5. Algorithm for constructing a timetable.

```
For 1 to n
  Repeat
    Construct a timetable using the algorithm in Figure 5
  Until the timetable is different from those already constructed
EndFor
Return the timetable with the lowest proximity cost as a solution
```

Fig. 8. Algorithm implemented by the overall system.

minimum cost is reported as the solution. In this study a value of *1000* was used for *n*. The overall approach is depicted in Fig. 8.

The following section outlines the methodology employed to test the performance of the four heuristic combinations.

6. Problem description and experimental setup

The different heuristic combinations were tested on real world examination timetabling problems, namely, the 13 Carter benchmarks listed in Table 1. Please note that there are two versions of some of these benchmarks (Qu et al., 2008) and the "I" indicates which version has been used.

The Carter set of benchmarks is generally used to test different approaches to the uncapacitated examination timetabling problem (Schaerf and Di Gaspero, 2006). The hard constraint for this set of benchmarks is that no student must be scheduled to write more than one examination at the same time, i.e. there must be no clashes. The soft constraint for this data set requires the examinations to be well spaced. The proximity cost function in Eq. (1), defined by Carter et al. (1996), is used to assess the quality of a timetable in terms of how well the examinations are spread. We aim to minimise the cost of this function for each examination timetabling problem.

$$\frac{\sum w(|e_i - e_j|)N_{ij}}{S}, \tag{1}$$

where:

(1) $|e_i$ minus $e_j|$ is the distance between the periods of each pair of examinations (e_i,e_j) with common students.
(2) N_{ij} is the number of students common to both examinations.
3) S is the total number of students.
4) it $w(1) = 16$, $w(2) = 8$, $w(3) = 4$, $w(4) = 2$ and $w(5) = 1$, i.e. the smaller the distance between periods the higher the weight allocated. Note for $n > 5$, $w(n) = 0$.

The system was implemented in Java using JDK 1.4.2 and simulations were run on an Apple iMac with a 2.16 MHz Intel Core 2 Duo processor and 1 GB of memory. Ten runs, each using a differ-

```
1.   For each period p
       if assigning e to p causes a clash then
         the cost assigned to p is the maximum double value
       else
         calculate the cost of scheduling exam e in period p using the
         functions defined in Figure 7
2.   Sort the periods in ascending order according to the calculated cost.
3.   Choose the period p with lowest cost for exam e.  If there is more
     than one period with the lowest cost, the period p is randomly
     chosen from the set of periods with the lowest cost.
```

Fig. 6. Algorithm for finding the best period *p* for examination *e*.

```
function calc_cost( exam e, period p, total number of students n)
begin
  cost = 0
  for each exam e_j other than e
  begin
    if(e_j has students in common with e and e_j has already been scheduled)
    begin
      dist = the absolute value of the distance between p and the period e_j
             has been allocated to
      ecost = weight(dist) * the number of students common to both exams
      cost = cost + ecost
   endfor
   return cost/n
end
```

Fig. 7. Pseudo code for the function used to calculate the cost of scheduling exam *e* in period *p*.

Please cite this article in press as: Pillay, N., Banzhaf, W., A study of heuristic combinations for hyper-heuristic systems ..., European Journal of Operational Research (2008), doi:10.1016/j.ejor.2008.07.023

Table 1
Carter benchmarks

Problem	Institution	Periods	No. of exams	No. of students	Density of conflict matrix
car-f-92 I	Carleton University, Ottawa	32	543	18,419	0.14
car-s-91 I	Carleton University, Ottawa	35	682	16,925	0.13
ear-f-83 I	Earl Haig Collegiate Institute, Toronto	24	190	1125	0.27
hec-s-92 I	Ecole des Hautes Etudes Commerciales, Montreal	18	81	2823	0.42
kfu-s-93	King Fahd University of Petroleum and Minerals, Dharan	20	461	5349	0.06
lse-f-91	London School of Economics	18	381	2726	0.06
pur-s-93 I	Purdue University, Indiana	43	2419	30,029	0.03
rye-s-93	Ryerson University, Toronto	23	486	11,483	0.08
sta-f-83 I	St Andrew's Junior High School, Toronto	13	139	611	0.14
tre-s-92	Trent University, Peterborough, Ontario	23	261	4360	0.18
uta-s-92 I	Faculty of Arts and Sciences, University of Toronto	35	622	21,266	0.13
ute-s-92	Faculty of Engineering, University of Toronto	10	184	2749	0.08
yor-f-83 I	York Mills Collegiate Institute, Toronto	21	181	941	0.29

Table 2
Performance of the different heuristic combinations

Problem	SD–HC	SD–HC(LD)	SD–HC(LWD)	SD–HC(LE)
car-f-92 I	Best: 4.33 Mean: 4.38 Time: 7 minutes	Best: **4.28** Mean: 4.36 Time: 7 minutes	Best: 4.31 Mean: 4.35 Time: 7 minutes	Best: 4.31 Mean: 4.35 Time: 7 minutes
car-s-91 I	Best: 5.08 Mean: 5.13 Time: 10 minutes	Best: 5.08 Mean: 5.12 Time: 10 minutes	Best: **4.97** Mean: 5.10 Time: 10 minutes	Best: **4.97** Mean: 5.10 Time: 10 minutes
ear-f-83 I	Best: 38.74 Mean: 39.55 Time: 2 minutes	Best: **36.86** Mean: 37.22 Time: 2 minutes	Best: 36.87 Mean: 37.16 Time: 2 minutes	Best: 37.11 Mean: 37.19 Time: 2 minutes
hec-s-92 I	Best: **11.85** Mean: 11.85 Time: 11 minutes	Best: 12.41 Mean: 12.41 Time: 29 minutes	Best: 12.09 Mean: 12.09 Time: 18 minutes	Best: 12.09 Mean: 12.09 Time: 16 minutes
kfu-s-93	Best: 15.61 Mean: 15.62 Time: 8 minutes	Best: 15.56 Mean: 15.58 Time: 6 minutes	Best: **14.62** Mean: 14.62 Time: 5 minutes	Best: **14.62** Mean: 14.62 Time: 5 minutes
lse-f-91	Best: **11.14** Mean: 11.14 Time: 2 minutes	Best: 11.20 Mean: 11.22 Time: 2 minutes	Best: 11.15 Mean: 11.18 Time: 2 minutes	Best: **11.14** Mean: 11.15 Time: 2 minutes
pur-s-93 I	Best: **4.73** Mean: 4.78 Time: 1 hour 20 minutes	Best: 4.75 Mean: 4.80 Time: 1 hour 27 minutes	Best: 4.74 Mean: 4.77 Time: 1 hour 26 minutes	Best: 4.74 Mean: 4.78 Time: 1 hour 26 minutes
rye-s-93	Best: 9.76 Mean: 9.80 Time: 4 minutes	Best: 10.3 Mean: 10.49 Time: 5 minutes	Best: **9.65** Mean: 9.70 Time: 5 minutes	Best: 9.69 Mean: 9.76 Time: 5 minutes
sta-f-83 I	Best: 159.62 Mean: 159.68 Time: 28 seconds	Best: 158.34 Mean: 158.46 Time: 30 seconds	Best: 158.45 Mean: 158.54 Time: 33 seconds	Best: **158.33** Mean: 158.53 Time: 30 seconds
tre-s-92	Best: 8.54 Mean: 8.57 Time: 2 minutes	Best: 8.84 Mean: 8.90 Time: 2 minutes	Best: 8.5 Mean: 8.54 Time: 2 minutes	Best: **8.48** Mean: 8.52 Time: 2 minutes
uta-s-92 I	Best: 3.47 Mean: 3.54 Time: 9 minutes	Best: **3.4** Mean: 3.43 Time: 9 minutes	Best: 3.41 Mean: 3.44 Time: 9 minutes	Best: 3.41 Mean: 3.44 Time: 9 minutes
ute-s-92	Best: 29.36 Mean: 29.37 Time: 2 minutes	Best: 29.50 Mean: 29.50 Time: 2 minutes	Best: 29.07 Mean: 29.09 Time: 1 minute	Best: **28.88** Mean: 28.88 Time: 1 minute
yor-f-83 I	Best: 41.88 Mean: 41.88 Time: 2 hours 30 minutes	Best: 41.91 Mean: 41.91 Time: 2 hours 30 minutes	Best: **40.74** Mean: 41.52 Time: 2 minutes	Best: 40.75 Mean: 41.21 Time: 2 minutes

ent random number generator seed, were performed for each of the 13 benchmarks. The results obtained are presented in the next section.

7. Results and discussion

This section discusses the performance of the four heuristic combinations on the 13 Carter benchmarks. Section 7.1 discusses the results obtained for each of the heuristic combinations listed in Section 5. In Section 7.2 the performance of the hierarchical combinations of heuristics presented in this paper is compared to other approaches combining heuristics. Section 7.2 also compares the results obtained to that of some of the initial studies in this domain and that of methods cited in the literature as producing the best quality timetable for at least one of the 13 Carter benchmarks.

7.1. Performance of the different heuristic combinations

Table 2 lists the best cost, the mean cost and the approximate computational time for each heuristic combination when applied to the Carter benchmarks. The cost for each timetable is calculated using the proximity cost function defined in Eq. (1) (Section 6). The best solutions generated are accessible from http://saturn.cs.unp.ac.za/~nelishiap/et/heuristics.htm. The proximity costs of each of these solutions are highlighted in Table 2. The mean cost for a problem is the average of the proximity cost of the best solution obtained for each of the ten runs performed. The overall system did not include mechanisms to ensure that feasible timetables were produced. However, the heuristic combinations produced feasible timetables for all the benchmarks. Generally different heuristic combinations were found to perform best on different data sets. The combination of SD and HC as primary heuristics, with SD as the priority heuristic, and LE as the secondary heuristic produced the best overall results, performing just as good as or better than the other heuristic combinations for 6 of the 13 Carter benchmarks.

The computational time for each heuristic combination is more or less the same for all problems except the York Mills Collegiate Institute data set. There is a marked difference between the time taken by the SD–HC and SD–HC(LD) combinations when compared to the computational time of the SD–HC(LWD) and SD–HC(LE) combinations. A more detailed study of the performance of these heuristic combinations was conducted to account for this difference.

The first two heuristic combinations require more time as a result of having to perform additional iterations to find distinct timetables whereas the latter combinations find timetables, different from the timetables already constructed, on the first iteration most of the time. Thus, the heuristic combinations using LWD or LE as a secondary heuristic tend to explore more of the search space for the York Mills data set than the combination without a secondary heuristic or the combination that uses LD as a secondary heuristic.

Please cite this article in press as: Pillay, N., Banzhaf, W., A study of heuristic combinations for hyper-heuristic systems ..., European Journal of Operational Research (2008), doi:10.1016/j.ejor.2008.07.023

Table 3
The best results obtained by the heuristic combinations and hyper-heuristic systems

Problem	Heuristic combinations	Tabu search (2005)	Fuzzy logic expert system	VNS	Tabu search (2007)
car-f-92 I	**4.28**	–	4.56	4.7	4.84
car-s-91 I	**4.97**	–	5.29	5.4	5.41
ear-f-83 I	**36.86**	45.60	37.02	37.29	38.19
hec-s-92 I	11.85	–	**11.78**	12.23	12.72
kfu-s-93	**14.62**	–	15.81	15.11	15.76
lse-f-91	**11.14**	–	12.09	12.71	13.15
pur-s-93 I	4.73	–	–	–	–
rye-s-93	**9.65**	–	10.35	–	–
sta-f-83 I	158.33	158.2	160.42	158.8	**158.19**
tre-s-92	**8.48**	–	8.67	8.67	8.85
uta-s-92 I	**3.4**	4.52	3.57	3.54	3.88
ute-s-92	28.88	35.40	**27.78**	29.68	31.65
yor-f-83 I	40.74	–	40.66	43.0	**40.13**

The following section compares the performance of the heuristic combinations applied in this study to those induced by other approaches used to combine heuristics.

7.2. Comparison with previous studies

This section compares the performance of the heuristic combinations used in this study to those produced by hyper-heuristic systems and applied to the same version of the Carter benchmarks. These studies include:

- The Tabu search applied by Burke et al. (2005) to the space of combinations of the largest degree and saturation degree heuristics.
- The fuzzy logic expert system employed by Asmuni et al. (2004).
- The VNS system used by Qu and Burke (2005) to search the heuristic search space to obtain the heuristic combination that produces the best quality timetable.
- The Tabu search implemented by Burke et al. (2007).

These studies are described in detail in Section 4. The best proximity cost obtained by these methods and the heuristic combinations used in the study presented in this paper are listed in Table 3. The best costs are highlighted. Note that the four studies that the heuristic combinations are compared to have employed a search to find the best heuristic combination, i.e. the space of heuristic combinations has been optimised whereas no optimisation is performed by the system presented in this paper. It is evident from Table 3 that the results obtained by the new approach used to combine and apply low-level heuristics, incorporating the use of HC, are comparable to that produced by hyper-heuristic methods. For 8 of the Carter benchmarks this approach has produced the best proximity costs when compared to those of heuristic combinations induced by the hyper-heuristic systems.

Although the study presented in this paper focuses on producing a methodology that generalises well over a spectrum of problems rather than a problem specific method that produces good results for one or more datasets, the performance of this method is compared to the best results cited in the literature for the Carter benchmarks to assess the potential of this methodology. The following studies have been cited in the literature as making a contribution to the field and have been applied to the same version of the Carter benchmarks used in this study:

- The EXAMINE system implemented by Carter et al. (1996).
- The Tabu search applied by Di Gaspero and Schaerf (2001).
- The sequential construction and backtracking methodologies employed by Caramia et al. (2008).

Table 4
The results obtained by methods cited as making a contribution to the field for the Carter benchmarks

Problem	Carter et al. (1996)	Di Gaspero and Schaerf (2001)	Caramia et al. (2008)	Merlot et al. (2003)	Yang and Petrovic (2004)	Burke and Bykov (2006a)	Burke et al. (2006b)	Eley (2007)	Abdullah et al. (2007)
car-f-92 I	6.2	5.2	6.0	4.3	3.93	4.42	**3.9**	4.8	4.4
car-s-91 I	7.1	6.2	6.6	5.1	4.50	**3.74**	4.6	5.7	5.2
ear-f-83 I	36.4	45.7	**29.3**	35.1	33.71	32.76	32.8	36.8	34.9
hec-s-92 I	10.8	12.4	**9.2**	10.6	10.83	10.15	10.0	11.3	10.3
kfu-s-93	14.0	18.0	13.8	13.5	13.82	**12.96**	13.0	15.0	13.5
lse-f-91	10.5	15.5	**9.6**	10.5	10.35	9.83	10.0	12.1	10.2
pur-s-93 I	3.9	–	**3.7**	–	–	–	–	5.4	–
rye-s-93	7.3	–	**6.8**	–	8.53	–	–	10.2	8.7
sta-f-83 I	161.5	160.8	158.2	157.3	158.35	157.03	156.9	157.2	159.2
tre-s-92	9.6	10.0	9.4	8.4	7.92	**7.75**	7.9	8.8	8.4
uta-s-92 I	3.5	4.2	3.5	3.5	3.14	**3.06**	3.2	3.8	3.6
ute-s-92	25.8	27.8	**24.4**	25.1	25.39	24.82	24.8	27.7	26.0
yor-f-83 I	41.7	41.0	36.2	37.4	36.53	**34.84**	34.9	39.6	36.2

Table 5
A comparison of the results obtained by the heuristic combinations and best result from Table 4

Problem	Heuristic combinations	Best result cited	Difference
car-f-92 I	4.28	3.9	0.38
car-s-91 I	4.97	3.74	1.23
ear-f-83 I	36.86	29.3	7.56
hec-s-92 I	11.85	9.2	2.65
kfu-s-93	14.62	12.96	1.66
lse-f-91	11.14	9.6	1.54
pur-s-93 I	4.73	3.7	1.03
rye-s-93	9.65	6.8	2.85
sta-f-83 I	158.33	134.9	23.43
tre-s-92	8.48	7.75	0.73
uta-s-92 I	3.4	3.06	0.34
ute-s-92	28.88	24.4	4.48
yor-f-83 I	40.74	34.84	5.9

- The hybrid system used by Merlot et al. (2003).
- The hybrid case-based reasoning system implemented by Yang and Petrovic (2004).
- The Flex-Deluge algorithm implemented by Burke and Bykov (2006a).
- The ant colonisation approach tested by Eley (2007).
- The variable neighbourhood search and genetic algorithm hybrid implemented by Burke et al. (2006b).
- The Ahuja–Orlin algorithm employed by Abdullah et al. (2007).

A detailed description of these studies is provided in Section 3. Table 4 lists the results obtained in each of these studies and Table 5 tabulates the difference in the best results from Table 4 and the results produced by the heuristic combinations. It is evident from Tables 4 and 5 that even though the method described in this paper only performs the construction phase and not an improvement phase the results produced by this methodology is comparable to the best results cited in the literature for the Carter benchmarks. Furthermore, this method has produced better results than some of the methodologies on a number of the benchmarks and outperformed the Tabu search on all of the benchmarks.

The results presented in this section show the effectiveness and potential of hierarchical heuristic combinations as a general methodology for producing good quality solutions to the uncapacitated examination timetabling problem. Future work will investigate automating the process of generating heuristic combinations so as to explore more of the heuristic search space and hence investigate the performance of more combinations. This will also allow for the heuristic combinations to be tailored to each problem domain.

8. Conclusion and future work

Hyper-heuristics systems choosing construction heuristics generally search a space of heuristics or a combination of low-level heuristics to produce the best heuristic or combination to allocate examinations during timetable construction. The study presented in this paper focuses on heuristic combinations. The main aim of this study is to test a new approach that can be used by hyper-heuristic systems to combine and apply low-level heuristics. In previous work heuristic combinations usually consist of a list of low-level heuristics that are applied sequentially. In this study the low-level heuristics are combined hierarchically and applied simultaneously. The study also presents a new low-level heuristic, namely, highest cost.

Four heuristic combinations, constructed and applied using the new approach, were tested on the set of 13 Carter benchmarks. In all cases feasible timetables were generated. The results obtained clearly indicate the potential of combining and applying low-level heuristics in this manner. The performance of these combinations have been found to be comparable to heuristic combinations produced by hyper-heuristic systems in previous studies and in a number of instances have outperformed the heuristic combinations generated by these systems. Furthermore, the quality of the timetables generated by this new approach is within range of the best quality timetables cited in the literature even though the system implemented in this study does not include an improvement phase to further minimise the soft constraint cost. Thus, the main contributions of this study are a new means of combining and applying low-level heuristics that can be used by hyper-heuristic systems, namely, the hierarchical combination of heuristics defining primary, secondary and priority heuristics and the simultaneous application of heuristics using a Pareto comparison as well as a new low-level heuristic, highest cost. The next step would be to implement a hyper-heuristic system to generate heuristic combinations of this form. Such a system would apply an optimisation technique such as Tabu search to explore the space of hierarchically combined low-level heuristics and identify the best combination for the problem at hand.

Future work will investigate using genetic programming (Koza, 1992) for this purpose. This process will begin with a randomly created initial population of hierarchically combined heuristics with each element consisting of different Pareto comparisons and primary, secondary, and priority heuristics. Each element of the population, i.e. heuristic combination, will be a parse tree comprised of elements of the function and terminal sets. As is evident from Figs. 3 and 4 each heuristic combination is comprised of if-statements, logical operators and the different low-level heuristics. Thus, the function set will basically consist of if-then–else statements and logical operators while the terminal set will essentially be the set of low-level heuristics. Primary, secondary and priority heuristics will be randomly chosen when creating each individual of the initial population. This population will then be iteratively refined, by applying genetic operators, crossover and mutation, to the fitter elements of the population, to obtain the best heuristic combination for the problem at hand. Tournament selection will be used to choose the parents of each generation.

The fitness of each individual will be the quality of the timetable constructed using the individual.

Acknowledgements

The authors would like to thank the reviewers for their helpful comments and suggestions.

References

Abdullah, S., Ahmadi, S., Burke, E.K., Dror, M., 2007. Investigating Ahuja–Orlin's large neighbourhood search for examination timetabling. OR Spectrum 29 (2), 351–372.

Asmuni, H., Burke, E.K., Garibaldi, J.M., 2004. Fuzzy multiple ordering criteria for examination timetabling. In: Burke, E.K., Trick, M. (Eds.), Selected Papers from the 5th International Conference on the Theory and Practice of Automated Timetabling (PATAT 2004) – The Theory and Practice of Automated Timetabling V, Lecture Notes in Computer Science, vol. 3616. Springer, Berlin, pp. 147–160.

Bilgin, B., Ozcan, E., Korkmaz, E.E., 2007. An experimental study on hyper-heuristics and exam timetabling. In: Burke, E.K., Rudova, H. (Eds.), Practice and Theory of Automated Timetabling VI: Selected Papers from the 6th International Conference, PATAT 2006, Lecture Notes in Computer Science, vol. 3867. Springer, Berlin, pp. 394–412.

Burke, E.K., Elliman, D.G., Ford, P.H., Weare, R.F., 1996. Examination timetabling in British Universities – a survey. In: Burke, E.K., Ross, P. (Eds.), Selected Papers from the 1st International Conference on the Practice and Theory of Automated Timetabling, Lecture Notes in Computer Science, vol. 1153. Springer, pp. 76–92.

Burke, E.K., Petrovic, S., 2002. Recent research directions in automated timetabling. European Journal of Operational Research 140 (2), 266–280.

Burke, E., Hart, E., Kendall, G., Newall, J., Ross, P., Schulenburg, S., 2003. Hyper-heuristics: An emerging direction in modern research technology. In: Handbook of Metaheuristics. Kluwer Academic Publishers, 2003, pp. 457-474 (Chapter 16).

Please cite this article in press as: Pillay, N., Banzhaf, W., A study of heuristic combinations for hyper-heuristic systems ..., European Journal of Operational Research (2008), doi:10.1016/j.ejor.2008.07.023

Burke, E.K., Newall, J.P., 2004. Solving examination timetabling problems through adaptation of heuristic orderings. Annals of Operations Research 129, 107–134.

Burke, E.K., Dror, M., Petrovic, S., Qu, R., 2005. Hybrid graph heuristics with a hyper-heuristic approach to exam timetabling problems. In: Golden, B.L., Raghavan, S., Wasil, E.A. (Eds.), The Next Wave in Computing, Optimization, and Decision Technologies – Conference Volume of the 9th Informs Computing Society Conference. Springer, pp. 79–91.

Burke, E.K., Bykov, Y., 2006a. Solving exam timetabling problems with the Flex-Deluge algorithm 2006a. In: Burke, E.K., Rudova, H. (Eds.), Proceedings of the International Conference on the Theory and Practice of Automated Timetabling (PATAT 2006), pp. 370–372.

Burke, E.K., Eckersley, A., McCollum, B., Petrovic, S., Qu, R., 2006b. Hybrid variable neighbourhood approaches to university exam timetabling. Technical Report NOTTCS-TR-2006-2, School of Computer Science and Information Technology, University of Nottingham, UK.

Burke, E.K., Petrovic, S., Qu, R., 2006c. Case-based heuristic selection for timetabling problems. Journal of Scheduling 9 (2), 115–132.

Burke, E.K., McCollum, B., Meisels, A., Petrovic, S., Qu, R., 2007. A graph-based hyper-heuristic for educational timetabling problems. European Journal of Operational Research 176, 177–192.

Caramia, M., Dell'Olmo, P., Italiano, G., 2008. Novel local-search-based approaches to university examination timetabling. INFORMS Journal of Computing 20 (1), 86–99.

Carter, M.W., Laporte, G., Lee, S.Y., 1996. Examination timetabling: Algorithmic strategies and applications. The Journal of the Operational Research Society 47 (3), 373–383.

Casey, S., Thompson, J., 2003. GRASPing the examination scheduling problem. In: Burke, E.K., De Causmaecker, P. (Eds.), The Practice and Theory of Automated Timetabling IV: Proceedings of the 4th International Conference on the Practice and Theory of Automated Timetabling, vol. 2740. Springer, Berlin, pp. 232–246.

Di Gaspero, L., Schaerf, A., 2001. Tabu search techniques for examination timetabling. In: Selected Papers from the 3rd International Conference on the Theory and Practice of Automated Timetabling. Lecture Notes in Computer Science, vol. 2079. Springer, Berlin, pp. 104–117.

Eley, E., 2007. Ant algorithms for the exam timetabling problem. In: Burke, E.K., Rudova, H. (Eds.), Practice and Theory of Automated Timetabling VI: Selected Papers from the 6th International Conference, PATAT 2006, Lecture Notes in Computer Science, vol. 3867. Springer, Berlin, pp. 364–382.

Ersoy, E., Ozcan, E., Uyar, S., 2007. Memetic algorithms and hill-climbers. In: Baptiste, P., Kendall, G., Kordon, A.M., Sourd, F. (Eds.), Proceedings of the 3rd Multidisciplinary International Conference on Scheduling: Theory and Applications. Paris, France, pp. 159–166.

Kendall, G., Hussin, N.M., 2005. An investigation of a Tabu search based on hyper-heuristics for examination timetabling. In: Kendall, G., Burke, E.K., Petrovic, S. (Eds.), Selected Papers form Multidisciplinary Scheduling: Theory and Applications, pp. 309–328.

Koza, J.R., 1992. Genetic Programming I: On the Programming of Computers by Natural Selection. MIT Press.

McCollum, B., 2007. A perspective on bridging the gap between research and practice in university timetabling. In: Burke, E.K., Rudova, H. (Eds.), Practice and Theory of Automated Timetabling VI, Lecture Note in Computer Science, vol. 3867. Springer, pp. 3–23.

Merlot, L.T.G., Boland, N., Hughes, B.D., Stuckey, P.J., 2003. A hybrid timetable for the examination timetabling problem. In: Burke, E.K., De Causmaecker, P. (Eds.), The Practice and Theory of Automated Timetabling IV: Proceedings of the 4th International Conference on the Practice and Theory of Automated Timetabling, Lecture Notes in Computer Science, vol. 2740. Springer, Berlin, pp. 207–231.

Meyers, C., Orlin, J.B., 2007. Very large-scale neighborhood search techniques in timetabling problems. In: Burke, E.K., Rudova, H. (Eds.), Practice and Theory of Automated Timetabling VI: Selected Papers from the 6th International Conference, PATAT 2006, Lecture Notes in Computer Science, vol. 3867. Springer, Berlin, pp. 24–39.

Qu, R., Burke, E.K., 2005. Hybrid neighbourhood hyper-heuristics for exam timetabling problems 2005. In: Proceedings of the MIC2005: The Sixth Metaheuristics International Conference, Vienna, Austria, August 2005.

Qu, R., Burke, E.K., McCollum, B., Merlot, L.T.G., Lee, S.Y., 2008. A survey of search methodologies and automated approaches for examination timetabling. Journal of Scheduling, in press. doi:10.1007/s10951-008-0077-5.

Ranson, D., Ahmadh, S., 2007. An extensible modelling framework for the examination timetabling problem. In: Burke, E.K., Rudova, H. (Eds.), Practice and Theory of Automated Timetabling VI: Selected Papers from the 6th International Conference, PATAT 2006, Lecture Notes in Computer Science, vol. 3867. Springer, Berlin, pp. 383–393.

Ross, P., Marin-Blazquez, J.G., Hart, E., 2004. Hyper-heuristics applied to class and exam timetabling. In: Proceedings of the 2004 Congress on Evolutionary Computation (CEC 2004), pp. 1691–1698.

Ross, P., 2005. Hyper-heuristics. In: Burke, E.K., Kendall, G. (Eds.), Search Methodologies: Introductory Tutorials in Optimization and Decision Support Methodologies. Kluwer Academic Publishers, pp. 529–556. Chapter 17.

Schaerf, A., Di Gaspero, L., 2006. Measurability and reproducibility in timetabling research: State-of-the-art and discussion. In: Burke, E.K., Rudova, H. (Eds.), Proceedings of the International Conference on the Theory and Practice of Automated Timetabling (PATAT 2006), pp. 53–62.

Ulker, O., Ozcan, E., Korkmaz, E.E., 2007. Linear linkage in grouping problems: Applications on graph coloring and timetabling. In: Burke, E.K., Rudova, H. (Eds.), Practice and Theory of Automated Timetabling VI: Selected Papers from the 6th International Conference, PATAT 2006, Lecture Notes in Computer Science, vol. 3867. Springer, Berlin, pp. 347–363.

Yang, Y., Petrovic, S., 2004. A Novel Similarity Measure for Heuristic Selection in Examination Timetabling. In: Burke, E.K., Trick, M. (Eds.), Selected Papers from the 5th International Conference on the Theory and Practice of Automated Timetabling (PATAT 2004) – The Theory and Practice of Automated Timetabling V, Lecture Notes in Computer Science, vol. 3616. Springer, Berlin, pp. 247–269.