A Genetic Programming Approach to the Generation of Hyper-Heuristics for the Uncapacitated Examination Timetabling Problem

Nelishia Pillay[1] and Wolfgang Banzhaf[2]

[1] School of Computer Science, Univesity of KwaZulu-Natal, Pietermaritzburg Campus, Pietermaritzburg, KwaZulu-Natal, South Africa
pillayn32@ukzn.ac.za
[2] Department of Computer Science, Memorial University of Newfoundland, St. John's, NL A1B 3X5, Canada
banzhaf@cs.mun.ca

Abstract. Research in the field of examination timetabling has developed in two directions. The first looks at applying various methodologies to induce examination timetables. The second takes an indirect approach to the problem and examines the generation of heuristics or combinations of heuristics, i.e. hyper-heuristics, to be used in the construction of examination timetables. The study presented in this paper focuses on the latter area. This paper presents a first attempt at using genetic programming for the evolution of hyper-heuristics for the uncapacitated examination timetabling problem. The system has been tested on 9 benchmark examination timetabling problems. Clash-free timetables were found for all 9 nine problems. Furthermore, the performance of the genetic programming system is comparable to, and in a number of cases has produced better quality timetables, than other search algorithms used to evolve hyper-heuristics for this set of problems.

Keywords: hyper-heuristics, genetic programming, examination timetabling.

1 Introduction

Research applying evolutionary algorithms to the domain of examination timetabling has generally focused on using an evolutionary algorithm to evolve a timetable that meets the hard constraints and minimizes the soft constraints of a specific examination timetabling problem. Ross et al. [15] and Burke et al. [5] suggest that evolutionary algorithms would be more effective at inducing hyper-heuristics that can be used to construct the timetable rather than evolving the actual timetable. The main contribution of this paper is an evaluation of genetic programming as a means of evolving hyper-heuristics for the uncapacitated examination timetabling problem (ETP). This study takes a similar approach to that employed by Cowling et al. [10], in that the hyper-heuristics evolved by the GP system are in the form of a sequence of low-level heuristics. In the study presented in this paper each heuristic sequence is composed of one or more of the following low-level heuristics: largest degree, largest

J. Neves, M. Santos, and J. Machado (Eds.): EPIA 2007, LNAI 4874, pp. 223–234, 2007.

weighted degree, largest enrollment, saturation degree and highest cost. During the timetable construction process, each heuristic is used to schedule *n* examinations, where *n* is the total number of examinations divided by the number of heuristics in the sequence. The GP system was used to generate hyper-heuristics for 9 benchmark problems. The system produced feasible timetables for all 9 problems. Furthermore, the quality of the timetables produced by the system was comparative to, and in a number cases better than, those induced by other hyper-heuristic systems.

The following section provides an overview of the examination timetabling problem and previous work investigating the generation of hyper-heuristics for the uncapacitated examination timetabling problem. Section **3** proposes a GP system for the induction of hyper-heuristics for the uncapacitated ETP. The overall methodology employed in the study is presented in section **4** and the performance of the GP-based hyper-heuristic system on the 9 benchmarks is discussed in section **5**. Section **6** summarizes the findings of the study and describes future extensions of the project.

2 The Uncapacitated Examination Timetabling Problem (ETP) and Hyper-Heuristics

This section provides an overview of the uncapacitated examination timetabling problem and previous work investigating the generation of hyper-heuristics for this domain.

2.1 The Uncapacitated Examination Timetabling Problem (ETP)

The ETP requires *n* examinations to be scheduled in *m* timeslots so as to satisfy the hard constraints of the problem and minimize the soft constraints violated [14]. The hard constraints of a problem must be met in order for the timetable to be feasible. The following are examples of hard constraints:

- There must be no clashes, i.e. two students cannot be scheduled to write two or more examinations during the same period.
- Room capacities for each sitting must not be exceeded.

Soft constraints are often contradictory and we aim to minimize the number of soft constraints violated. For example:

- Examinations are well-spaced for each group of students.
- Scheduling examinations with large enrollments earlier in the examination period.

The hard and soft constraints usually differ for each institution. The uncapacitated version of the problem does not require room capacities to be catered for. The following section describes previous studies researching the induction of hyper-heuristics for the uncapacitated ETP.

2.2 Hyper-Heuristics and the ETP

Hyper-heuristics are heuristics that are used to choose one or more low-level heuristics for a particular problem ([5], [15] and [16]). These selections do not rely on domain knowledge and hence provide a more general solution to the ETP. The first attempt at using evolutionary algorithms to evolve instructions for constructing an examination timetable rather than inducing the actual timetable was the study conducted by Terashima-Marin et al. [17] to generate solutions to the capacitated ETP. Each element of the population is a combination of one of three timetable construction strategies, condition/s indicating when to change strategy as well as heuristics for deciding which examination to allocate next and which timeslot to assign an examination to. This section provides an overview of previous work investigating the generation of hyper-heuristics for the uncapacitated examination timetabling problem. The studies described are those that are most relevant to that presented in the paper, i.e. the methodologies employed generate combinations of low-level heuristics and have been tested on the same set of benchmarks.

Asmuni et al. [2] have used a fuzzy expert system to induce hyper-heuristics for the uncapacitated ETP. The hyper-heuristic is in the form of a fuzzy weight combining two of the following low-level heuristics: largest degree, largest enrollment and largest saturation degree. An exhaustive search is used to fine tune the fuzzy terms. The examinations are sorted in descending order according to their fuzzy weight and scheduled in this order. Each examination is allocated to the minimum penalty slot. In the case of clashes examinations are de-allocated and re-scheduled so as to remove the clash.

Qu et al. [13] use a variable neighborhood search (VNS) to search the space of hyper-heuristics for the uncapacitated ETP. The hyper-heuristic output by the VNS is used to construct the timetable. Each hyper-heuristic is a sequence of two or more low-level heuristics (color degree, largest degree, largest enrollment, largest weighted degree, saturation degree and random ordering). Each low-level heuristic in the sequence is used to decide which examination should be scheduled next. A heuristic maybe used to schedule one or more examinations. In each case the examination is allocated to the minimum penalty slot. During the search process the VNS uses one of two neighborhood sets, namely, VNS1 and VNS2. VNS1 randomly changes 2 to 5 heuristics in a sequence, whereas VNS2 randomly changes 2 to 5 heuristics in a subsequence.

Kendall et al. [11] and Burke et al. [7] have implemented Tabu searches to generate hyper-heuristics for the uncapacitated ETP. In the study conducted by Kendall et al. the process begins by creating an initial solution by allocating examinations according to the largest degree or saturation degree heuristics. This initial solution may not be complete in that all examinations may not be scheduled. The neighborhood of the initial solution is than examined so as to improve the timetable. The Tabu search is used to determine which low-level heuristic to apply next. One of four types of low-level heuristics can be applied, namely, heuristics to select and schedule an exam; heuristics to move an exam; a heuristic to swap exams and a heuristic to remove an exam. The Tabu inactive heuristic that produces the best

improvement is applied next. The refinement process continues until either a time limit has been exceeded or there are no more improvements.

The Tabu search employed by Burke et al. [7] is used to search the hyper-heuristic search space in order to identify a heuristic sequence 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 enrollment 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 following section presents a GP system for the generation of hyper-heuristics. Section **5** compares the performance of the GP-based hyper-heuristic system and the hyper-heuristic systems described in this section on a set of 9 benchmark problems.

3 Evolving Hyper-Heuristics

This section describes the GP system that has been implemented to evolve hyper-heuristics for the uncapacitated ETP. Genetic programming systems generally produce a program which when executed provides a solution to the problem at hand ([3] and [12]). In this study the program is a sequence of low-level heuristics which specify the order in which examinations should be scheduled when constructing an examination timetable. The GP system uses the generational control model and a run is terminated once the maximum number of generations has been reached. The hyper-heuristic that has produced the timetable with lowest hard constraint and soft constraint costs during the run is returned as the solution.

3.1 Representation and Initial Population Generation

Each element of the population is a string of variable length composed of characters representing one of the following low-level heuristics:

- Largest degree (l) – The examination with the largest number of conflicts is scheduled first.
- Largest enrollment (e) – The examination with the largest student enrollment is scheduled first.
- Largest weighted degree (w) – The examination with the largest number of students involved in clashes is scheduled first.
- Saturation degree (s) – The examination with the least number of feasible (i.e. will not result in a clash) timeslot options is scheduled first.
- Highest cost (h) - The examination with the highest proximity cost is scheduled first. The pseudo-code of the function for calculating the proximity cost is listed in **Figure 1** and the weight function used in calculating the proximity cost is defined in **Figure 2**.

An example of a hyper-heuristic is *hsseel*. The hyper-heuristic strings are randomly created during initial population generation. A limit is set on the maximum length of each hyper-heuristic.

```
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
      endif
    endfor
  return cost/n
end
```

Fig. 1. Pseudo-code for the proximity cost

```
function 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

3.2 Evaluation and Selection

Each element of the population is evaluated by using the hyper-heuristic to construct a timetable. Each low-level heuristic in the hyper-heuristic is used to schedule n examinations, where n is the total number of examinations divided by the length of the hyper-heuristic. Each examination is allocated to the minimum penalty slot, i.e. the slot that does not cause a clash and results in the lowest proximity cost. The raw fitness of a hyper-heuristic is calculated by applying equation (1) to the timetable constructed using the hyper-heuristic:

$$(\text{number\_of\_clashes}+1)*\text{proximity\_cost} \quad (1)$$

The proximity cost is a measure of how well the examinations are spaced and is calculated using equation (2) in section **4**. Tournament selection is used to choose the parents of the next generation.

3.3 Genetic Operators

The mutation and crossover operators are used to create the next generation. A limit is not set on the length of the offspring produced by the genetic operators. The mutation operator changes a randomly chosen heuristic. This process is illustrated in **Figure 3**.

1. Randomly choose a heuristic to change in the parent ***h****ssess*

2. Replace the chosen heuristic with a randomly selected heuristic (from *h*, *s*, *l*, *e*, *w*)

   w*ssess*

Fig. 3. The mutation process

The crossover operator is depicted in **Figure 4**.

1. Randomly choose crossover points in both the parents

   P_1: *lwesh* P_2: *sshhle*

2. Swap the fragments at the crossover points to create offspring

   O_1: *lwshhle* O_2: *sesh*

3. Return the fitter of the two offspring

Fig. 4. The crossover process

The crossover operator randomly selects crossover points in each of the chosen parents. Two offspring are created by swapping the fragments at the crossover points. The fitter of the two offspring is returned as the result of the operation.

4 Experimental Setup

The GP-based hyper-heuristic system was tested on 9 of the Carter benchmarks [9]. These benchmarks are data sets for real-world exam timetabling problems from various universities and high schools. The characteristics of the data sets used are listed in **Table 1**. The density of the conflict matrix is to some extent an indication of the problem difficulty and is calculated to be the ratio of the number of examinations involved in clashes and the total number of exams.

The hard constraint for this problem is that no student must be scheduled to write two or more examinations at the same time, i.e. there must be no clashes. The soft

Table 1. Carter benchmarks

Data Set	Institution	Periods	No. of Exams	No. of Students	Density of Conflict Matrix
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
rye-s-93	Ryerson University, Toronto	23	486	11483	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
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

constraint requires the examinations to be well-spaced. The soft constraint cost is referred to as the proximity cost and is calculated using the following equation:

$$\frac{\sum w(|e_i - e_j|)N_{ij}}{S} \tag{2}$$

where:

1) $|e_i - e_j|$ 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) 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 that this equation calculates the proximity cost of a complete timetable whereas the pseudo-code in **Figure 1** calculates the cost of scheduling a particular examination in a partially constructed timetable, given the examinations that have already been allocated to a timeslot at that point. For problems with different hard and soft constraints the function for calculating the raw fitness will be different from that defined in section **3.2**.

The GP parameter values are listed in **Table 2**. These values have been obtained empirically by performing test runs for each of the 9 data sets.

Table 2. GP Parameters

Population size	500
No. of generations	50
Maximum initial length	5
Tournament size	10
Mutation rate	40%
Crossover rate	60%

The system was implemented in Java using JDK1.4.2 and simulations were run on a Windows XP machine with an Intel Pentium M with 512 MB of RAM.

5 Results and Discussion

Due to the randomness associated with GP and hence the possibility of selection noise[1] ten runs were performed for each data set. The duration of a run ranges from about 15 minutes for the smaller data sets to about 4 hours for the larger data sets. The system evolved feasible timetables for all 9 problems. **Table 3** lists the average proximity costs and the best individual and its proximity cost for each of the data sets. The timetables constructed for each data set using the best hyper-heuristic obtained can be found at http://saturn.cs.unp.ac.za/~nelishiap/et/hyper_heuristics.htm.

The saturation degree (s) and the highest cost (h) heuristics occur most frequently in the best hyper-heuristics found for each of the data sets. **Figure 5** illustrates the distribution of calls to low-level heuristics in the best hyper-heuristic found for all data sets. Note that the saturation degree, highest cost and largest degree heuristics are invoked in the best hyper-heuristics for almost all the data sets with the highest cost and saturation degree occurring most frequently. The saturation degree heuristic appears to speed up the construction of clash-free timetables while the highest cost heuristic reduces the soft constraint cost.

We compare the performance of the GP system to other methodologies applied to generating hyper-heuristics for the uncapacitated ETP. **Table 4** lists the performance of the GP system and other methods employed to generate hyper-heuristics (details of

Table 3. Performance of the GP system on the Carter benchmarks

Data Set	Average Proximity Cost	Best Proximity Cost	Best Hyper-Heuristic
ear-f-83 I	36.94	36.74	hsshsssshslssh
hec-s-92 I	11.64	11.55	hhlssll
kfu-s-93	14.25	14.22	hhslssssll
lse-f-91	10.97	10.90	hsshshshshhhshhshshhhllhhhww
rye-s-93	9.39	9.35	hssshshshees
sta-f-83 I	158.35	158.22	hsehshehshhhhshhshwhswl
tre-s-92	8.51	8.48	hssshshshees
ute-s-92	27.61	26.65	hhssshsslessshssslshshssssesllllsswssh
yor-f-83 I	41.82	41.57	ssshh

[1] Please note that the reason for performing more than one run is not to show statistical significance.

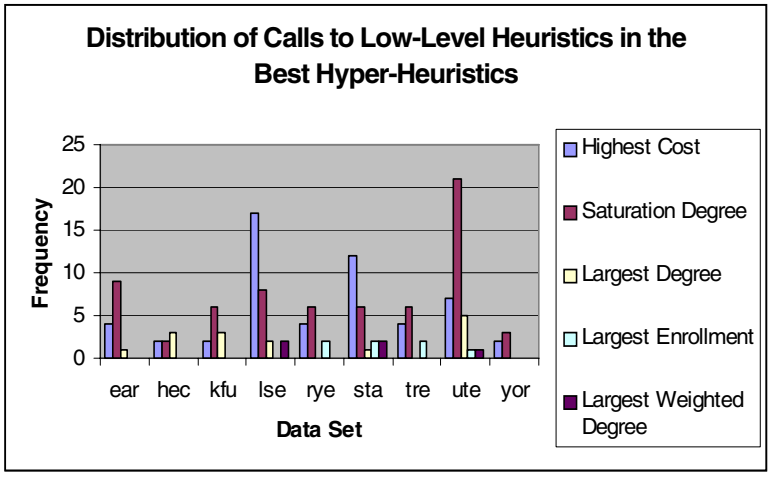

Fig. 5. Distribution of calls to low-level heuristics

these systems are provided in section **2.2**) for the same set of benchmarks. The best soft constraint cost obtained for each of the data sets is highlighted. These methodologies are:

- FES – The fuzzy expert system implemented by Asmuni et al. [2].
- VNS – The variable neighborhood search implemented by Qu et al. [13].
- TS1 – The Tabu search applied by Kendall et al. [11]. In this study each simulation was run for 4 hours.
- TS2 – The Tabu search implemented by Burke et al. [7].

Note that the methodologies that the GP system is being compared to employ very different search mechanisms from that used by the system and a direct comparison of the parameters used is therefore not feasible.

It is evident from **Table 4** that the performance of genetic programming is comparable to the other search methods used to generate hyper-heuristics. The quality

Table 4. The performance of the GP system and other methodologies used to induce hyper-heuristics for the uncapacitated ETP

Data Set	GP	FES	VNS	TS1	TS2
ear-f-83 I	**36.74**	37.02	37.29	40.18	38.19
hec-s-92 I	**11.55**	11.78	12.23	11.86	12.72
kfu-s-93	**14.22**	15.81	15.11	15.84	15.76
lse-f-91	**10.90**	12.09	12.71	-	13.15
rye-s-93	**9.35**	10.35	-	-	-
sta-f-83 I	158.22	160.42	**139.3**	157.38	141.08
tre-s-92	8.48	8.67	8.67	**8.39**	8.85
ute-s-92	**26.65**	27.78	29.68	27.60	31.65
yor-f-83 I	41.57	40.66	43.0	-	**40.13**

of the timetables constructed using the hyper-heuristics induced by the GP system is within the range of those produced by other hyper-heuristic systems. Furthermore, the GP-based hyper-heuristic system has outperformed the other hyper-heuristic systems on 6 of the benchmarks.

The main aim of the study presented in this paper is to evaluate GP as a means of generating hyper-heuristics for the uncapaciated ETP. Hence, emphasis is on producing a general method rather than a method that produces the best results on the the set of benchmarks. However, for completeness we compare the performance of the GP-based hyper-heuristic system to that of methodologies that have been cited in the literature as producing the best results for these benchmarks. **Table 5** compares the performance of the GP system to the following studies:

- The sequential construction and backtracking methodologies employed by Caramia et al. [8].
- The hybrid case-based reasoning system implemented by Yang et al. [18].
- The Ahuja-Orlin algorithm employed by Abdullah et al. [1].
- The Flex-Deluge algorithm implemented by Burke et al. [6].

Note that the system presented in this paper only performs the construction phase for timetable induction while the other methods listed in **Table 5** also include an improvement phase aimed at reducing the proximity cost of feasible timetables produced during the construction phase. Despite this the results produced by the GP system are still within range of the best results produced for the data sets.

Table 5. A comparison of the results obtained by the GP system and the best results cited for the benchmarks

Problem	GP	Caramia et al., 2001	Yang et al. 2004	Abdullah et al., 2004	Burke et, al. 2006	Difference
ear-f-83 I	36.74	**29.3**	33.71	34.84	32.76	7.44
hec-s-92 I	11.55	**9.2**	10.83	10.28	10.15	2.35
kfu-s-93	14.22	13.8	13.82	13.46	**12.96**	1.26
lse-f-91	10.90	**9.6**	10.35	10.24	9.83	1.30
rye-s-93	9.35	**6.8**	8.53	8.7	-	2.55
sta-f-83 I	158.22	158.2	**151.52**	159.28	157.03	6.7
tre-s-92	8.48	9.4	7.92	8.13	**7.75**	0.73
ute-s-92	26.65	24.4	25.39	**24.21**	24.82	2.44
yor-f-83 I	41.57	36.2	36.53	36.11	**34.84**	6.73

6 Conclusion and Future Work

The study presented in this paper is a first attempt at evaluating genetic programming for the purpose of inducing hyper-heuristics for the uncapacitated ETP. A GP system was implemented to induce hyper-heuristics for this problem and was tested on 9 of the Carter benchmarks. This study has revealed the potential of genetic programming as a means of evolving hyper-heuristics for the uncapacitated ETP. The GP system

generated hyper-heuristics that produced feasible examination timetables with soft constraint costs within the range of other search methods employed for this purpose. Furthermore, the GP-based hyper-heuristic system outperformed the other hyper-heuristic systems on 6 of the 9 problems.

One of the drawbacks of this system is the runtime for larger data sets. For example, the runtime for the *kfu-s-93* set is approximately three and a half hours and just over four hours for the *rye-s*-93 data set. Future work will address improving the runtime of the overall system and testing it on additional benchmarks and problems with different hard and soft constraints. In this study a limit was not set on the size of the offspring produced and this did not appear to result in bloating. However, a closer look needs to be taken into the effect of not using such a limit and the overall effect of introns and bloat in this domain. Future extensions of this study will also investigate evolving programs that apply each heuristic more than once, e.g. h4s2, will apply the highest cost heuristic four times and the saturation degree heuristic twice.

Acknowledgments. The authors would like to thank the reviewers for their helpful comments and suggestions. This material is based on work financially supported by the National Research Foundation (NRF) of South Africa.

References

1. Abdullah, S., Ahmadi, S., Burke, E.K., Dror, M.: Investigating Ahuja-Orlin's Large Neighbourhood Search for Examination Timetabling. Technical Report NOTTCS-TR-2004-8, School of CSiT, University of Nottingham, U.K (2004)
2. Asmuni, H., Burke, E.K., Garibaldi, J.M.: Fuzzy Multiple Ordering Criteria for Examination Timetabling. In: Burke, E.K., Trick, M.A. (eds.) PATAT 2004. LNCS, vol. 3616, pp. 147–160. Springer, Heidelberg (2005)
3. Banzhaf, W., Nordin, P., Keller, R.E., Francone, F.D.: Genetic Programming - An Introduction - On the Automatic Evolution of Computer Programs and its Applications. Morgan Kaufmann Publishers, Inc., San Francisco (1998)
4. Burke, E.K., Petrovic, S.: Recent Research Directions in Automated Timetabling. European Journal of Operational Research 140(2), 266–280 (2002)
5. Burke, E., Hart, E., Kendall, G., Newall, J., Ross, P., Schulenburg, S.: Hyper-Heuristics: An Emerging Direction in Modern Research. In: Handbook of Metaheuristics, ch. 16, pp. 457–474. Kluwer Academic Publishers, Dordrecht (2003)
6. Burke, E.K., Bykov, Y.: Solving Exam Timetabling Problems with the Flex-Deluge Algorithm. In: Burke, E.K., Rudova, H. (eds.) Proceedings of PATAT 2006, pp. 370–372 (2006)
7. Burke, E.K., McCollum, B., Meisels, A., Petrovic, S., Qu, R.: A Graph-Based Hyper-Heuristic for Educational Timetabling Problems. European Journal of Operational Research 176, 177–192 (2007)
8. Caramia, M., Dell'Olmo, P., Italiano, G.: New Algorithms for Examination Timetabling. In: Näher, S., Wagner, D. (eds.) WAE 2000. LNCS, vol. 1982, pp. 230–241. Springer, Heidelberg (2001)
9. Carter, M.W., Laporte, G., Lee, S.Y.: Examination Timetabling: Algorithmic Strategies and Applications. Journal of the Operational Research Society 47(3), 373–383 (1996)

10. Cowling, P., Kendall, G., Han, L.: An Investigation of a Hyperheuristic Genetic Algorithm Applied to a Trainer Scheduling Problem. In: Proceedings of Congress on Evolutionary Computation (CEC), Hilton Hawaiian Village Hotel, Honolulu, Hawaii, May 12 -17, 2002, pp. 1185–1190 (2002) ISBN 0-7803-7284-2
11. Kendall, G., Mohd Hussin, N.: An Investigation of a Tabu Search Based on Hyper-Heuristics for Examination Timetabling. In: Proceedings of MISTA (Multidisciplinary International Conference on Scheduling) 2003, Nottingham, UK (2003)
12. Koza, J.R.: Genetic Programming I: On the Programming of Computers by Means of Natural Selection. MIT Press, Cambridge (1992)
13. Qu, R., Burke, E.K.: A Hybrid Neighbourhood Hyper-Heuristic for Exam Timetabling Problems. In: Proceedings of MIC 2005: The 6th Metaheuristics International Conference, Vienna, Austria (2005)
14. Qu, R., Burke, E., McCollum, B., Merlot, L.T.G., Lee, S.Y.: A Survey of Methodologies and Automated Approaches for Examination Timetabling. Technical Report NOTTCS-TR-2006-4 (2006), http://www.cs.nott.ac.uk/TR-cgi/TR.cgi
15. Ross, P., Hart, E., Corne, D.: Some Observations about GA-based Exam Timetabling. In: Burke, E.K., Carter, M. (eds.) PATAT 1997. LNCS, vol. 1408, pp. 115–129. Springer, Heidelberg (1998)
16. Ross, P.: Hyper-heuristics. In: Burke, E.K., Kendall, G. (eds.) Search Methodologies: Introductory Tutorials in Optimization and Decision Support Methodologies, ch. 17, pp. 529–556. Kluwer, Dordrecht (2005)
17. Terashima-Marin, H., Ross, P., Valenzuela-Rendon, M.: Evolution of Constraint Satisfaction Strategies in Examination Timetabling. In: Banzhaf, W., et al. (eds.) Proceedings of GECCO 1999: Genetic Programming and Evolutionary Computation Conference, pp. 635–642. Morgan Kaufmann, San Francisco (1999)
18. Yang, Y., Petrovic, S.: A Novel Similarity Measure for Heuristic Selection in Examination Timetabling. In: Burke, E.K., Trick, M. (eds.) PATAT 2004. LNCS, vol. 3616, pp. 247–269. Springer, Heidelberg (2005)