An Analysis of Representations for Hyper-Heuristics for the Uncapacitated Examination Timetabling Problem in a Genetic Programming System

Nelishia Pillay
School of Computer Science
University of KwaZulu-Natal, Pietermaritzburg Campus,
KwaZulu-Natal, South Africa
+27 33 260 5644
pillayn32@ukzn.ac.za

ABSTRACT

Earlier research into the examination timetabling problem focused on applying different methodologies to generate solutions to the problem. More recently research has been directed at developing hyper-heuristic systems for timetable construction. Hyper-heuristic systems are used to decide which examination to schedule next during the timetable construction process and aim at allocating those examinations that are most difficult to schedule first. This study investigates using a genetic programming based hyper-heuristic system to evolve heuristic combinations for the uncapacitated examination timetabling problem. More specifically it presents and evaluates three different representations for heuristic combinations in a genetic programming system. The performance of the genetic programming based system using the different representations is applied to three examination timetabling problems with different characteristics and the performance on these problems is compared. The results obtained are also compared to that of other hyper-heuristic systems applied to the same problems.

Categories and Subject Descriptors

I.2. [**Computing Methodologies**]: – *Artificial Intelligence*

General Terms

Algorithms, Theory

Keywords

Hyper-heuristics, genetic programming, examination timetabling

1. INTRODUCTION

A lot of research has been conducted into evaluating techniques such as Tabu search, constraint programming and simulated annealing as a means of producing the best quality examination timetables for one or more problems of a benchmark set of problems [8]. Hyper-heuristic systems on the other hand are aimed at generalizing over problems for a particular domain rather than finding the best solution for a few problems [2]. Hyper-heuristic systems basically generate a heuristic or heuristic combinations to be applied during the timetable construction process. Tabu search, variable neighbourhood search and case-based reasoning are some of the methods used to implement hyper-heuristic systems for the uncapacitated examination timetabling problem. The study presented in this paper examines the use of genetic programming (GP) as a means of evolving heuristic combinations to be used during the timetable construction process. The representation used by a GP system affects the search process and the hence the success of the system. This study compares the performance of a GP-based hyper-heuristic system using three different representations for heuristic combinations on three real-world examination timetabling problems. The results obtained are also compared to that produced by other hyper-heuristic systems for the same set of problems.

The following section presents the background details of this study. Section 3 describes the GP system and representations implemented and section 4 presents the methodology employed to evaluate the representations. The performance of the GP system with the three different representations is compared in section 5. Section 5 also presents a comparison of the performance of the GP-based system and other hyper-heuristic systems. Section 6 summarizes the findings of the study and discusses future extensions of this work.

2. BACKGROUND

The examination timetabling problem involves scheduling examinations to timetable periods so as to meet the hard constraints of the problem and minimize the soft constraints. Examples of hard constraints include ensuring that each student is only scheduled to write one examination during a period, i.e. there are no clashes; room capacities are not exceeded. The hard constraints of a problem have to be met in order for a timetable to be operable. A timetable that meets all the hard constraints of the problem is referred to as a feasible timetable. The capacitated version of the problem has the additional hard constraint that

SAICSIT 2008, 6 - 8 October 2008, Wilderness Beach Hotel, Wilderness, South Africa

venue capacities are not exceeded for any period whereas the uncapacitated version does not have this constraint.

The soft constraints of a problem are characteristics that we would like our timetable to have, all of which may not be possible, for example ensuring that examinations are well spread for all students; larger examinations are scheduled earlier in the examination session to facilitate marking. Soft constraints are often contradictory and we aim to minimize the soft constraint cost. The quality of an examination timetable is defined in terms of the soft constraint cost. The hard and soft constraints differ from one problem to the next.

The standard sequential method used to construct timetables essentially involves sorting the examinations in order of difficulty and allocating each exam to a feasible period, i.e. a period that does not result in a clash. A low-level heuristic is usually used to estimate the difficulty of each exam. Low-level heuristics generally used for this purpose are the graph colouring heuristics, largest degree, largest weighted degree, largest enrollment, and saturation degree. Such heuristics are referred to as construction heuristics. Hyper-heuristic systems are used to determine which heuristic or heuristic combination should be used in deciding which examination to schedule next. Methods employed by hyper-heuristic systems to generate construction heuristic combinations for the uncapaciated examination timetabling problem include fuzzy logic, Tabu search and variable neighbourhood search.

In the study conducted by Asmuni et al. [1] each heuristic combination takes the form of a fuzzy weight combining two of the low-level heuristics, largest degree, largest enrolment and saturation degree. A fuzzy expert system is used to produce the weight. The list of examinations to be allocated is sorted according to the fuzzy weight value and scheduled in order to the minimum penalty period.

Studies conducted by Burke et al.[3] and Burke et al.[4] have used a Tabu search to explore the space of heuristic combinations. In both these studies the Tabu search is used to search a space comprised of lists of low-level heuristics to find the optimal heuristic list for the problem at hand. Each list is composed of two or more of the saturation degree, largest colour degree, largest degree, largest weighted degree and largest enrolment heuristics and random ordering. In the first study each heuristic is used to schedule three examinations during the timetable construction process and in the second each heuristic is used to allocate two examinations. The performance of each heuristic list is assessed in terms of the quality of the timetable it is used to construct.

Qu et al. [7] employ a variable neighbourhood search (VNS) to search the space of heuristic combinations comprised of two or more of the saturation degree, largest colour degree, largest degree, largest weighted degree and largest enrolment constructive heuristics and random ordering. Each heuristic in the combination is used to schedule one examination to a minimum penalty period.

Section 5 will present a comparison of the performance of the GP-based hyper-heuristic system presented in this study and that of the methods presented in this section on three real-world problems from the Carter benchmark set.

3. GP SYSTEM AND DIFFERENT HYPER-HEURISTIC REPRESENTATIONS

Like genetic algorithms, genetic programming (GP) is based on Darwin's theory of evolution [6]. However, while genetic algorithms generate a solution to a problem, GP systems induce a program which when executed will produce the corresponding solution. Thus, GP systems search a program space rather than a solution space. For example, in the context of examination timetabling a genetic algorithm will search the space of timetables while a genetic programming system will search the space of algorithms for constructing the timetable. In genetic programming systems the evolutionary process begins with an initial population which is iteratively refined to produce a solution program. On each iteration, referred to as a generation, the elements of the population are evaluated using a fitness function and a selection method is used to choose the fitter elements of the population as parents of the next generation. Genetic operators are applied to the chosen parents to create the offspring of the next generation. This process of evaluation, selection and regeneration is repeated until either a solution is found or a set number of generations are completed. This section describes the GP system implemented in this study and the three representations to be tested.

3.1 Representation and Initial Population Generation

The GP system implements the generational control model. Thus, the overall evolutionary process consists of a fixed number of generations. The first generation is referred to as the initial population generation and involves creating a population of programs randomly. Each element of the population is a heuristic combination and is comprised of one or more of the following low-level heuristics:

- Largest degree (l) – The examination with the most number of clashes is scheduled first.
- Largest enrolment (e) – The examinations with the most number of students sitting for the examination is given priority.
- 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 period options is scheduled first. A feasible period is one that does not result in a clash.
- Highest cost (h) – The examination with the highest soft constraint cost, based on the current state of the timetable, is scheduled first.

This study tests the following three representations for combining these low-level heuristics:

- Fixed length heuristic combinations (FHC) – The length of each heuristic combination is fixed to be the number of examinations. For example, if the number of examinations to be scheduled is five the combination *lessh* could be an element of the initial population. In this case the first examination to be scheduled will be chosen based on the largest degree heuristic, the second examination according to the largest enrollment heuristic, the saturation degree will be used to determine the next two exams to be allocated and the highest cost heuristic the last exam.

- Variable length heuristic combinations (VHC) – The length of the heuristic combination is randomly chosen to be between one and a preset maximum length, e.g. *hssl*. Each heuristic is used to schedule one examination. If the length of the heuristic combination is larger than the number of examinations, only the heuristics in the substring of the combination, beginning at the first heuristic, with length equal to the number of examinations are applied. If the heuristic combination contains less heuristics than the number of examinations the heuristics are reapplied in order from the first heuristic in the string.
- N-times heuristic combinations (NHC) – Each low-level heuristic is preceded by the number of times the heuristic must be applied. For example, application of the heuristic combination *2h3w2l* will result in the first two examinations being scheduled according to the highest cost heuristic, the next three according to the largest weighted degree heuristic and the last two exams based on the largest degree heuristic. The sum of the number of applications of each heuristic is equal to the total number of examinations to be scheduled. Thus, in the example the total number of examinations is seven.

For all three representations, the low-level heuristic at each position of a heuristic combination is randomly chosen. In the case of the NHC the number of times each heuristic is applied is also randomly chosen to be between one and a preset maximum value. The initial population consists of *m* heuristic combinations. The value of *m* is a genetic parameter and varies from one problem to the next. At each stage of the evolutionary process each element of the population is evaluated and parents of the next generation are selected. These processes are described in the next section.

3.2 Evaluation and Selection

Each heuristic combination is evaluated by using the combination to construct the corresponding timetable. The fitness of a heuristic combination is a function of the hard and soft constraint cost of the constructed timetable. The fitness function thus differs from one timetabling problem to the next as the hard and soft constraints often vary drastically from one problem to the next. The tournament selection method is used to choose the parents of the next generation. This method involves randomly choosing *t* elements of the population, referred to as the tournament, and returning the fitness element of the tournament as the winner. The winner is a parent of the next generation. In the case of the examination timetabling problem a low fitness value indicates a fitter individual as we aim to minimize the hard and soft constraint cost. Selection is performed with replacement, i.e. an individual maybe chosen as a parent more than once in a generation.

3.3 Genetic Operators

The mutation and crossover operators are applied to the chosen parents to create the offspring of the next generation. The mutation operator creates a copy of the parent and randomly changes a low-level heuristic, or the number of applications of a heuristic in the case of NHC. The position of the heuristic to be changed, referred to as the mutation point, is randomly selected and the heuristic value at this position is changed to be a randomly selected heuristic. For example, if the parent is *shhls* and the mutation point is one, the corresponding offspring maybe *ehhls*.

The crossover operator randomly selects crossover points in copies of two parents. The fragments at the crossover points are swapped to create two offspring. The fitter offspring is returned as the result of the crossover operator. For example, if the parents are *hsels* and *llsses* and the crossover points have been randomly chosen to be *2* and *4*, then the corresponding offspring will be *hses* and *llssels*. In the case of the NHC representation the crossover operator is structure preserving to ensure that the offspring has the correct format, i.e. each low-level heuristic is preceded by the number of applications of the heuristic.

The application rates of both these operators are problem dependant and are hence specified as GP parameters.

4. EXPERIMENTAL SETUP

The performance of the three different representations was tested on three of the Carter benchmark problems [5] listed in **Table 1**.

Table 1. Benchmarks Problems

Data Set	Periods	No. of Exams	No. of Students	Density of Conflict Matrix
hec-s-92 I	18	81	2823	0.42
sta-f-83 I	13	139	611	0.14
ute-s-92	10	184	2749	0.08

These three problems were selected due to the differences in problem characteristics, e.g. the density of the conflict matrix. All three problems are real-world problems. The *hec-s-92I* data set is from the Ecole des Hautes Etudes Commerciales, Montreal, *sta-f-83I* from St Andrew's Junior High School, Toronto and *ute-s-92* from the Faculty of Engineering, University of Toronto. The density of the conflict matrix is the ratio of the number of examinations involved in clashes to the total number of examinations and is a measure of the difficulty of the problem.

The hard and soft constraints for all three problems are the same. The hard constraint 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 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:

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

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.

The fitness function is the number of clashes plus one times the proximity cost defined in equation (1). **Table 2** lists the values of the parameters used in the study. These values were obtained empirically by performing trial runs. A lower population size resulted in the population not representing enough of the search space and a higher value did not improve the success of the system. In all trial runs the GP system converged within a hundred generations.

Eleven different pairs of application rates, i.e. mutation and crossover rates, were tested for the three data sets. Five runs, each using a different initial seed for the random number generator, were performed for each pair. The rates were than ranked according to the average fitness for the five solutions found using the rate. The average of the ranks for each data set was used to determine which rate performed well across all three data sets.

Table 2. GP Parameters

Population size	500
No. of generations	100
Maximum initial length	5
Tournament size	10
Mutation rate	70%
Crossover rate	30%

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 genetic programming and to show statistical significance thirty runs were performed for each data set using each of the representations FHC, VHC and NHC. Feasible solutions were found using all three representations. **Table 3** lists the average and best soft constraint cost obtained by each representation for each of the data sets over thirty runs.

For all three data sets the VHC and NHC representations have produced better results than the FHC representation and VHC appears to have performed better on two of the data sets than NHC. Hypothesis tests were performed to test the statistical significance of these results. The levels of significance, critical values and decision rules used are listed in **Table 4**.

Table 3: Best and Average Soft Constraint Cost for the Three Representations

Data Set	FHC	VHC	NHC
hec-s-92 I	**Best:** 11.44 **Average:** 11.78	**Best:** 11.23 **Average:** 11.46	**Best:** 11.26 **Average:** 11.53
sta-f-83 I	**Best:** 158.54 **Average:** 158.95	**Best:** 157.82 **Average:** 158.32	**Best:** 157.59 **Average:** 158.24
ute-s-92	**Best:** 27.36 **Average:** 27.36	**Best:** 27.13 **Average:** 27.74	**Best:** 27.24 **Average:** 27.78

Table 4: Levels of significance, critical values and decision rules

P	Critical Value	Decision Rule
0.01	2.33	Reject H_o if Z >2.33
0.05	1.64	Reject H_o if Z >1.64
0.10	1.28	Reject H_o if Z >1.28

Table 5 lists the Z-values for each of the data sets for the corresponding hypotheses. In the case of the first two hypotheses, the Z-values are greater than the critical values for all levels of significance, for all three data sets, indicating that the null hypothesis is rejected in all cases. Thus, we can conclude that the results presented in **Table 3** regarding the performance of FHC compared to VHC and NHC is significant at all levels of significance. Similarly, the result that VHC performs better than NHC is statistically significant for *hec-s-92I* for all levels of significance. However, this result is not significant for the data set *ute-s-92*. For the data set *sta-f-83I* the result that NHC performs better than VHC is only significant at the 0.10 level of significance. Based on the statistical tests it can be concluded that the FHC representation is generally not suitable for the examination timetabling domain. Furthermore, it is evident that it cannot be generally concluded that VHC performs better than NHC or vice versa and their performance appears to be problem dependant, i.e. VHC and NHC may perform well for some problem domains and not for others. Future work will test this concept further by applying VHC and NHC to additional data sets with varying characteristics.

For completeness the best results obtained by the GP-based hyper-heuristic system is compared to the performance of the hyper-heuristic systems described in section 2. **Table 6** compares the best result obtained by the GP-based system and the other hyper-heuristic systems. In all cases feasible timetables were generated, thus the values listed are the soft constraint costs, i.e. the proximity cost defined in equation (1) of section 4. The proximity cost represents the quality of the timetables.

Table 5: Hypotheses and Z- values for the comparison of the performance of FHC, VHC and NHC

Hypothesis	Z Value		
	hec-s-92	*sta-f-83*	*ute-s-92*
H_o: $\mu_{FHC} = \mu_{VHC}$ H_a: $\mu_{FHC} > \mu_{VHC}$	8.76	12.07	6.81
H_o: $\mu_{FHC} = \mu_{NHC}$ H_a: $\mu_{FHC} > \mu_{NHC}$	6.07	9.83	5.68
H_o: $\mu_{NHC} = \mu_{VHC}$ H_a: $\mu_{NHC} > \mu_{VHC}$	2.56	-	0.55
H_o: $\mu_{VHC} = \mu_{NHC}$ H_a: $\mu_{VHC} > \mu_{NHC}$	-	1.34	-

Table 6: Comparison of the performance of the GP system and other hyper-heuristic systems

Data Set	GP	Asumni et al. [1]	Burke et al. [3]	Burke et al. [4]	Qu et al. [7]
hec-s-92 I	11.23	11.78	-	12.72	12.23
sta-f-83 I	157.59	160.42	158.2	141.08	139.3
ute-s-92	27.13	27.78	35.40	31.65	29.68

For both the data sets *hec-s-92 I* and *ute-s-92* the GP-based hyper-heuristic system has produced better results than the other hyper-heuristic systems irrespective of the representation used. For the *sta-f-83 I* data set the best result produced by the GP-based system is better than that produced by Asumni et al. [1] and Burke et al. [3] but is not as good as that obtained by Burke et al. [4] and Qu et al. [7], however it is within range of these soft constraint costs.

6. CONCLUSION AND FUTURE WORK

The main aim of the study presented in this paper is to test the effect of different representations on the success of a GP-based hyper-heuristic system for the uncapacitated examination timetabling problem. The GP-based system with the three different representations was tested on three real-world problems from the Carter benchmark sets. This study revealed that the FHC representation does not perform as well as the VHC and the NHC representations for the examination timetabling domain. This result was found to be statistically significant.

The study has also indicated that the performance of the NHC and VHC representations differ from one problem to the other, and which of these two representations is more suitable is problem dependant. This hypothesis will be tested further by applying the GP-based system with both these representations to additional problems. Future work will also examine which representation works better for which type of problem, in terms of the problem characteristics, and whether an expert module can be incorporated into the system to decide on which representation to use based on the problem characteristics such as the density of the conflict matrix, number of students, etc. This will enable the GP-based hyper-heuristic system to generalize further. Investigations into combining the different representations will also be conducted.

This study has also highlighted the potential of using genetic programming as the basis of a hyper-heuristic system for the examination timetable problem. The results produced by the GP-based hyper-heuristic system were found to be comparative to, and in a number of cases better, than that of hyper-heuristic systems employing other methodologies.

7. ACKNOWLEDGMENTS

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

8. REFERENCES

[1] Asumni, H., Burke, E. K., and Garibaldi, J. M. Fuzzy Multiple Ordering Criteria for Examination Timetabling. In: *Burke EK, 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, 2005 , 795-825.

[2] Burke E., Hart E., Kendall G., Newall J., Ross P., and Schulenburg S. Hyper-Heuristics: An Emerging Direction in Modern Research. *In Handbook of Metaheuristics, Chapter 16*. Kluwer Academic Publishers, 2003, 457– 474.

[3] Burke E.K., Dror M., Petrovic S., and Qu R. 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, 2005, 79 - 91.

[4] Burke E.K., McCollum B., Meisels A., Petrovic S., and Qu R. A Graph-Based Hyper-Heuristic for Educational Timetabling Problems. *European Journal of Operational Research, 176*. 2007, 177 - 192.

[5] Carter M. W., Laporte G., and Lee S.Y. Examination Timetabling: Algorithmic Strategies and Applications. *The Journal of the Operational Research Society*, *47*, *3*. 1996, 373 - 383.

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

[7] Qu R., and Burke E.K. Hybrid Neighbourhood HyperHeuristic for Exam Timetabling Problems. In: *Proceedings of the MIC2005: The Sixth Metaheuristics International Conference, Vienna, Austria.* 2005.

[8] Qu R., Burke E.K., McCollum B., Merlot L.T.G., and Lee S.Y. *A Survey of Search Methodologies and Automated Approaches for Examination Timetabling*. Computer Science Technical Report No. NOTTCS-TR-2006-4, UK, 2006.