Evolving Construction Heuristics for the Curriculum Based University Course Timetabling Problem

Nelishia Pillay
School of Mathematics, Statistics and Computer Science
University of KwaZulu-Natal
South Africa
Email: pillayn32@ukzn.ac.za

***Abstract*—In solving combinatorial optimization problems construction heuristics are generally used to create an initial solution which is improved using optimization techniques like genetic algorithms. These construction heuristics are usually derived by humans and this is usually quite a time consuming task. Furthermore, according to the no free lunch theorem different heuristics are effective for different problem instances. Ideally we would like to derive construction heuristics for different problem instances or classes of problems. However, due to the time it takes to manually derive construction heuristics it is generally not feasible to induce problem instance specific heuristics. The research presented in the paper forms part of the initiative aimed at automating the derivation of construction heuristics. Genetic programming is used to evolve construction heuristics for the curriculum based university course timetabling (CB-CTT) problem. Each heuristic is a hierarchical combination of problem characteristics and a period selection heuristic. The paper firstly presents and analyses the performance of known construction heuristics for CB-CTT. The analysis has shown that different heuristics are effective for different problem instances. The paper then presents the genetic programming approach for the automated induction of construction heuristics for the CB-CTT problem and evaluates the approach on the ITC 2007 problem instances for the second international timetabling competition. The evolved heuristics performed better than the known construction heuristics, producing timetables with lower soft constraint costs.**

Keywords - automatic heuristic derivation, genetic programming, university course timetabling, hyper-heuristics

I. Introduction

When applying a metaheuristic such as simulated annealing, tabu search or genetic algorithms to solve combinatorial optimization problems, a construction heuristic is usually used to create initial solutions which the metaheuristic than optimizes. Using a construction heuristic has proven to be more effective than randomly creating an initial solution in that less optimization is needed to improve the initial solution [1]. Furthermore, different heuristics are more effective for different problem instances. These construction heuristics are manually derived. This is a time consuming process and usually a set of construction heuristics is derived for a particular problem domain rather than per problem instance. For example, construction heuristics used for the one dimensional bin packing problem include the first fit, best fit and next fit heuristics [2]. Similarly, for the domain of university examination timetabling the saturation degree, largest degree and largest enrolment heuristics are commonly used [3]. The research presented in this paper aims at automating the process of construction heuristic derivation, thereby facilitating the induction of problem instance specific heuristics. Genetic programming [4] is employed for the automatic evolution of heuristics.

Research in this area was initiated by the study conducted by Burke et al. [5] which employed genetic programming to evolve construction heuristics for the one dimensional bin packing problem. The function set was comprised of arithmetic and conditional operators and the terminal set characteristics of the problem e.g. bin capacity, item size and fullness of the bin. The best performing evolved heuristic performed the function of the first fit heuristic. Drake et al. [6] use the same approach to evolve construction heuristics for the multidimensional knapsack problem. Pillay et al. [7] [8] induce heuristics that combine existing construction heuristics for the examination timetabling problem hierarchically and apply them simultaneously. In [8] genetic programming is used to achieve this by using if-then-else statements to apply more then one existing heuristic simultaneously. The condition of the if - then-else statement combines the heuristics with arithmetic logical and logical operators with the corresponding actions being the examination to allocate next. Bader-El-Den et al. [9] implement a variation of genetic programming, namely, grammatical evolution to evolve construction heuristics for the examination timetabling problem. Existing low-level heuristics are decomposed into their basic components which are recombined, together with period selection heuristics, using a grammar.

The research presented in this paper uses genetic programming to evolve construction heuristics for a new domain, namely, curriculum based university course timetabling. Furthermore, rather than combining problem characteristics with arithmetic and conditional operators as has been done in previous work for the one dimensional bin packing and university examination timetabling domains, genetic programming is used to combine both existing heuristics as well as characteristics of the problem domain hierarchically together with a period selection heuristic. As this is an initial study combining heuristics in this way the aim is to get an indication of the feasibility and effectiveness of using such combinations. Hence the number of heuristics and problem characteristics

978-1-5090-0623-6/16/$31.00

used and combined are small and based on the outcome of this study will be extended in future work. The following section introduces the curriculum based university course timetabling problem and provides an analysis of existing heuristics for solving the problem. The genetic programming approach implemented for solving the problem is described in section IV. Section V discusses the performance of the heuristics evolved by the genetic programming approach and compares it with that of the manually derived heuristics generally used for this problem domain. The results of this study and future work are summarised in section VI.

II. Curriculum Based Course Timetabling

As with educational timetabling in general, curriculum based course timetabling (CB-CTT) involves allocating lessons for different courses to timetable periods so as to satisfy certain hard constraints which must be met in order for the timetable to be feasible. Two hard constraints that must be generally met for all timetabling problems is that all lessons must be scheduled and that there must no conflicts, i.e. students, teachers and venues must not be scheduled more than once in a timetable period. In the curriculum based course timetabling problem the curricula compiled by the university define the potential conflicts [10]. Each curriculum is comprised of a set of courses. Each course has a set number of lessons per week, taught by a particular lecturer. In addition to satisfying hard constraints, timetable allocations must be also be made to meet certain soft constraints. However, it is not possible to meet all soft constraints as these are usually contradictory and the aim is to minimize the number of soft constraints violated. The number of soft constraints violated is a measure of the quality of the timetable. The study presented in this paper uses the benchmark data set for the curriculum based course timetabling track of the second international timetabling competition (ITC 2007) which is described in section II-A below. An analysis of known construction heuristics for the domain of curriculum based course timetabling is provided in section II-B.

A. ITC 2007 Benchmark Data Set

The benchmark set provides 21 CB-CTT problem instances. The details of the instances described in [11] are depicted in Table I. The conflicts are listed per lecture and is calculated to be the number of pairs of potential conflicts divided by the total number of lectures.

For each problem instance a set of periods which lectures for the course cannot be scheduled in is specified. The hard constraints for the benchmark problem set are:

- Lecture allocations - The required number of lectures must be scheduled in the timetable.
- Conflicts - Lectures for courses in a curriculum must be scheduled in different periods.
- RoomOccupancy - Each room must only be scheduled once in a period.
- Availabilities - Each teacher must not be scheduled more than once in a period.

TABLE I
ITC 2007 CB-CTT Problem Instances

Instance	Courses	Rooms	Curricula	Days	Periods	Conflicts
comp01	30	6	14	6	5	13.2
comp02	82	16	70	5	5	7.97
comp03	72	16	68	5	5	8.17
comp04	79	18	57	5	5	5.42
comp05	56	9	139	6	6	21.7
comp06	108	18	70	5	5	5.24
comp07	131	20	77	5	5	4.48
comp08	86	18	61	5	5	4.52
comp09	76	18	75	5	5	6.64
comp10	115	18	67	5	5	5.03
comp11	30	5	13	9	5	13.8
comp12	88	11	150	6	6	13.9
comp13	82	9	66	5	5	5.61
comp14	85	17	60	5	5	6.87
comp15	72	16	68	5	5	8.17
comp16	108	20	71	5	5	5.13
comp17	99	1	70	5	5	5.5
comp18	47	9	52	6	6	13.3
comp19	74	16	66	5	5	7.45
comp20	121	19	78	5	5	5.06
comp21	94	18	78	5	5	6.09

The soft constraints are:

- RoomCapacity -The number of students in a course allocated to a room for a lecture must not exceed the capacity of the room.
- MinimumWorkingDays - The lectures for a course must be distributed over the specified minimum number of working days.
- CurriculumCompactness - Lectures for courses of a curriculum should be scheduled adjacent to each other on each day.
- RoomStability - All the lectures for a course should be scheduled in the same venue.

B. An Analysis of Construction Heuristics for the CB-CTT

An analysis of the performance of existing course timetabling construction heuristics in creating initial solutions for the CB-CTT problem was conducted. Construction heuristics for selecting which lecture to schedule next are essentially a measure of the difficulty of scheduling the lecture. The lectures will be sorted according to the heuristic and allocated in order. The following 3 lecture selection heuristics are commonly used for this domain [3] and will hence be evaluated:

- Largest enrolment (LE) - The lectures involving the largest number of students are scheduled first.
- Largest degree (LD) - The lectures that have the largest number of potential clashes are given priority to be scheduled.
- Saturation degree (SD) - The lectures with the least number of feasible periods in the timetable at the current point of construction are given priority.

The largest enrolment and largest degree heuristics are static heuristics and are calculated before the timetable construction process and remain the same throughout the process. The

saturation degree heuristic is a dynamic heuristic and its value is updated during the timetable construction process. The initial saturation degree is the total number of periods for the week. Two versions for the largest degree heuristic were considered. The first is the number of courses a course has potential conflicts with, i.e. the number of courses in the curricula it forms part of. The second is the number of lectures that it has potential clashes with, i.e. the total number of lectures for the courses in the curricula that it belongs to. The latter proved to be more effective and has been used in the analysis. Three options were considered for deciding which period in the timetable to schedule the chosen lecture in:

- First Period (FP) - The lecture is scheduled in the first feasible period in the timetable. A feasible period is one that does not result in a hard constraint violation when the lecture is allocated to it.
- Random Period (RP) - The period is randomly chosen from all the feasible periods in the timetable.
- Minimum Penalty Period (MPP) - The soft constraint cost for each of the feasible periods in the partially created timetable is calculated. The lecture is scheduled in the period with the minimum soft constraint cost. It was found that curriculum compact constraint was not a good reflection of the soft constraint cost in the partially constructed timetable in the initial stages as all the courses in the particular curriculum were not allocated as yet. Hence this was not weighted by 2 as is in calculating the soft constraint cost [11].

A timetable for each problem instance is created by sorting the courses to be scheduled according to the chosen heuristic, i.e. largest enrolment, largest degree or saturation degree. The lectures for each course are then scheduled sequentially in order. The period that a lecture is scheduled in is chosen using either the first period, random period or minimum penalty period heuristic. If a feasible period is not found for the lecture, it is not allocated and the hard constraint cost is incremented. In the case of the saturation degree heuristic, after each timetable allocation the saturation degree of the courses in the same curricula are updated and the courses resorted for allocation.

Table II lists the hard and soft constraint cost obtained for each combination of the course selection and period selection heuristic. The first row for each problem instance lists the hard constraint cost and the second row the soft constraint cost. From Table II it is evident that the combination of the saturation degree heuristic together with minimum penalty period heuristic performs the best for a majority of the problem instances. However, it can be seen that for some of the problem instances other combinations are more effective. For example, the random period with largest degree and saturation degree produce the best results for comp06 and comp07 respectively. Similarly, the largest enrolment heuristic with the first period heuristic is the most effective for comp20. From this analysis it can be seen that different heuristic combinations are effective for different problem instances. The following section describes the genetic programming approach to automate the induction of construction heuristics based on this analysis.

III. Automating Heuristic Derivation Using Genetic Programming

This section presents the genetic programming approach for automating the derivation of construction heuristics. The genetic programming algorithm is generational [4] and begins by creating an initial population that is iteratively refined via the processes of evaluation, selection and regeneration over a set number of generations. Each element of the population represents a construction heuristic and combines the problem characteristics together with a period selection heuristic. The problem characteristics form the terminal set and include the following:

- Saturation degree (s) - This is the saturation degree heuristic defined in the previous section.
- Number of students (u) - The number of students enrolled in the course. This is equivalent to the largest enrolment (LE) heuristic defined in the previous section.
- Largest degree (d) - This is the largest degree heuristic defined in the previous section.
- Lectures (l) - The number of lectures that must be allocated for a course.
- Unavailabilities (v) - The number of periods, i.e. day and period combination, which the lectures for the course must not be scheduled in.
- Minimum number of working days (c) -The minimum number of working days over which the lectures of a course must be distributed.
- Room degree (r) - The number of rooms with a capacity greater than or equal to the number of students enrolled for the course.

The period heuristics defined in the previous section, i.e first period (f), random period (r) and minimum penalty period (m) are also included in the terminal set.The function set consists of two elements, namely, ”#” and ”%”. The ”#” is used to combine characteristics of the problem. A minimum of 2 and a maximum of 3 problem characteristics can be combined. The ”%” is used to combine the characteristics with one of the period selection heuristics. Examples of heuristics are illustrated in Fig. 1. The first individual combines 2 problem characteristics, namely, saturation degree and the number of lectures, together with the minimum penalty period heuristic. In applying this heuristic all the courses will firstly be sorted in ascending order according to the saturation degree heuristic. In cases where two courses have the same saturation degree the number of courses will be used to break ties, i.e. the course with the higher number of lectures will be given priority. The period to schedule each lecture for a course is chosen using the minimum penalty period heuristic. In application of the second individual all the courses are firstly sorted in descending order according to the number of unavailabilities for each course. In the case of courses with the same number of unavailabilities the largest degree is used to to break ties. If there are courses with the same largest degree the minimum number of working

TABLE II
PERFORMANCE COMPARISON OF CONSTRUCTION HEURISTICS IN TERMS OF HARD CONSTRAINT COST (FIRST ROW) AND SOFT CONSTRAINT COST (SECOND ROW)

Instance		**FP**			**RP**			**MPP**	
	LE	LD	SD	LE	LD	SD	LE	LD	SD
comp01	12	8	11	5	5	11	4	**4**	8
	337	335	320	216	224	235	196	**193**	59
comp02	8	7	2	4	3	0	4	5	**0**
	749	716	658	866	881	899	843	885	**437**
comp03	9	6	0	0	2	0	0	4	**0**
	656	643	600	744	811	811	741	775	**328**
comp04	3	2	0	0	0	0	0	0	**0**
	646	666	702	687	729	720	702	748	**320**
comp05	6	11	**2**	3	3	1	3	4	4
	747	750	**784**	1539	1498	1452	1579	1562	654
comp06	2	8	4	1	1	**0**	2	2	4
	927	956	979	849	915	**926**	969	950	356
comp07	5	11	7	1	**0**	0	1	0	1
	1110	1041	1075	1062	**1038**	1119	1098	1054	559
comp08	0	0	0	0	0	0	0	1	**0**
	747	704	705	771	800	798	757	851	**346**
comp09	3	0	0	2	0	0	0	1	**0**
	698	708	701	868	873	839	788	895	**383**
comp10	5	3	8	0	0	0	0	0	**0**
	989	894	901	931	932	891	969	936	**456**
comp11	0	12	0	0	2	5	**0**	1	1
	263	275	272	243	248	270	**221**	228	34
comp12	7	14	0	5	9	0	6	5	**0**
	901	925	1000	1483	1619	1735	1701	1476	**839**
comp13	5	4	0	0	0	0	0	0	**0**
	712	678	727	756	818	808	806	754	**328**
comp14	4	4	0	1	3	0	1	0	**0**
	696	754	752	739	757	781	808	774	**324**
comp15	9	6	0	2	0	0	2	3	**0**
	656	463	600	782	751	862	805	803	**328**
comp16	6	7	2	4	4	0	0	0	**0**
	963	946	925	954	954	872	931	953	**338**
comp17	15	10	0	0	3	0	1	4	**0**
	962	898	916	948	897	885	880	963	**371**
comp18	1	0	0	0	0	0	0	0	**0**
	441	447	427	721	677	632	690	657	**256**
comp19	3	11	0	6	3	3	2	2	**0**
	669	730	639	828	713	721	715	776	**333**
comp20	6	15	17	**0**	3	1	2	6	4
	1085	1037	1062	**1109**	1006	993	1048	1042	495
comp21	13	5	0	2	2	0	2	4	**0**
	864	856	904	986	960	987	1013	984	**411**

days is used as the tie breaker. In the second individual the random period heuristic is used to select periods to allocate lectures to. An individual is created by randomly selecting arguments for "%" and "#" nodes.

Each element of the population is evaluated by using it to create a timetable. The timetable is created using the same procedure described in the previous section. The fitness of an individual is the product of the hard constraint cost of the timetable constructed using the heuristic plus one and the soft constraint cost. The fitness is minimized. Tournament selection [4] is used to choose parents for creation of the next generation.

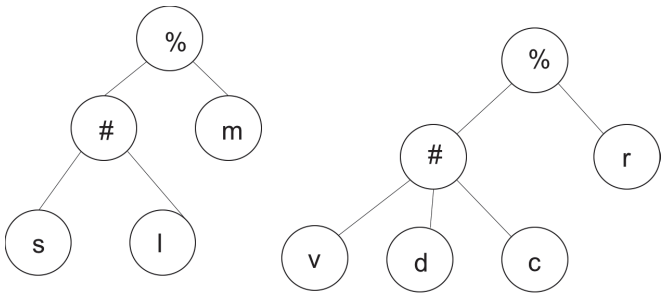

Fig. 1. Examples of Individuals in the Population

Mutation and crossover are used to create the offspring of each generation. These are the standard parse tree operators presented in [4]. The mutation operator randomly selects a mutation point in a copy of the parent. If the mutation point is the root a new individual is created. If "#" is at the mutation point, a new subtree is created and replaces the subtree rooted at "#". In the case of the leaf nodes, the node is replaced with a randomly selected node of the same type, i.e. characteristic or

TABLE III
GP PARAMETER VALUES

Parameter	Value
Population size	500
No. of generations	15
Tournament size	4
Mutation rate	0.5
Crossover rate	0.5

period selection heuristic. In order to ensure that the crossover operator produces syntactically correct offspring all the nodes are assigned a type. There are four types, namely, *comb*, *combh* for "%" and "#" respectively, *char* for the characteristic of a problem, and *psh* for a period selection heuristic. A crossover point is firstly selected in the first parent. A crossover point of the same type is selected in the second parent. The subtrees rooted at both crossover points are swapped to create two offspring.

Trial runs were performed to determine the parameter values to use for experiments. These are listed in Table III. Population sizes of 300, 500 and a 1000 were tested. A smaller population size did not sufficiently represent the search space and a higher population size did not improve the performance of the genetic programming approach. The algorithm was found to converge within 15 generations. Tournament sizes of 2, 4 and 10 were tested with 4 proving to be the most appropriate. Genetic operator application rates [4] in the range 0 to 1, with steps of 0.1 were tested. An equal balance between exploration and exploitation was found to be sufficient.

The genetic programming approach has been implemented in Java and all simulations have been run on an HP Z80 workstation with Windows 7. Due to the stochastic nature of genetic programming 10 runs, each with a different random number seed, were conducted for each problem instance. The performance of the evolved heuristics are discussed in the following section.

IV. RESULTS AND DISCUSSION

Table IV lists the best hard constraint cost, soft constraint cost and the heuristics evolved over the 10 runs. The genetic programming approach generated heuristics that produced feasible timetables for all the problem instances accept comp01. It is anticipated that the reason for this could be that problem characteristics specific to comp01 have not been taken into consideration. Upon further examination it is found that the lectures not scheduled were due to a room of sufficient capacity not being available in a feasible period. These courses require a room capacity greater than 30 and there are only two venues of the required capacity available and hence these courses are competing with a number of other courses for these venues. Although the number of rooms with sufficient capacity is included as a problem characteristic, this is a static value and what is probably needed is a dynamic value, the number of available rooms for a course at the current point of timetable construction. This will be investigated as part of future work.

TABLE IV
PERFORMANCE OF THE GENETIC PROGRAMMING APPROACH

Instance	Hard Constraint Cost	Soft Constraint Cost	Evolved Heuristic
comp01	4	63	%#vudm
comp02	0	388	%#slm %#slrm
comp03	0	280	%#slum
comp04	0	278	%#scum
comp05	0	1005	%#suur
comp06	0	355	%#svcm
comp07	0	419	%#sum
comp08	0	292	%#cusm
comp09	0	342	%#scum
comp10	0	343	%#sdcm
comp11	0	40	%#vssm %#vsum
comp12	0	802	%#scvm
comp13	0	273	%#scdm
comp14	0	305	%#lsdm
comp15	0	280	%#slum
comp16	0	283	%#svum
comp17	0	368	%#svum
comp18	0	219	%#svum
comp19	0	316	%#svcm
comp20	0	519	%#vdsm
comp21	0	405	%#svm %#rsvm

The runtimes ranged from 1 minute 15 seconds to 27 minutes for the bigger problem instances such as comp07. For all of the problem instances, with an exception of comp05, the best evolved heuristic was the same or one of two heuristics (comp02, comp11 and comp21) for all 10 runs. Different evolved heuristics produced the best results on each of the 10 runs for comp05. However, all the best heuristics used random period as the period selection heuristic and the first characteristic in all the heuristics was the saturation degree heuristic.

It is interesting to note that the heuristic producing the best results is the same for both problem instances comp03 and comp15. Both these problem instances have the same number of conflicts per lecture (see Table I in section II). Similarly, the same best heuristic has been evolved for problem instance comp04 and comp09; comp06 and comp19 and comp16 and comp17. The similarities betweens these pairs of problem instances is not evident and will also be investigated as part of future work.

The performance of the evolved heuristics is compared to that of the existing heuristics for this domain presented in section II-B. As the aim of this research is to automate the process of deriving construction heuristics the performance of the evolved heuristics is compared to that of manually derived heuristics and not other optimization techniques or state of the art approaches as this is not a fair comparison as heuristics are not expected to achieve the same level of optimization but to provide an initial solution that can be further optimized. A more effective construction heuristic is one that would produce a timetable with a lower hard constraint or soft constraint cost if the heuristics compared achieved feasibility,

TABLE V
PERFORMANCE COMPARISON OF THE MANUALLY DERIVED HEURISTICS AND THE EVOLVED HEURISTICS

Instance	A	B	C	D	E	F
comp01	LD+MPP	4	193	%#vudm	4	63
comp02	SD+MPP	0	437	%#slm	0	388
comp03	SD+MPP	0	328	%#slum	0	280
comp04	SD+MPP	0	320	%#scum	0	280
comp05	SD+RP	1	1452	%#suur	0	1005
comp06	LD+RP	0	926	%#svcm	0	355
comp07	LD+RP	0	1038	%#lsum	0	419
comp08	SD+MPP	0	346	%#cusm	0	292
comp09	SD+MPP	0	383	%#scum	0	342
comp10	SD+MPP	0	456	%#sdcm	0	343
comp11	LE+MPP	0	221	%#vssm	0	40
comp12	SD+MPP	0	839	%#scvm	0	802
comp13	SD+MPP	0	328	%#scdm	0	273
comp14	SD+MPP	0	324	%#lsdm	0	305
comp15	SD+MPP	0	328	%#slum	0	280
comp16	SD+MPP	0	338	%#svum	0	283
comp17	SD+MPP	0	371	%#svum	0	368
comp18	SD+MPP	0	256	%#csdm	0	219
comp19	SD+MPP	0	333	%#svcm	0	316
comp20	LE+RP	0	1109	%#vdsm	0	519
comp21	SD+MPP	0	411	%#svm	0	405

thereby reducing the amount of further optimization needed. Table V compares the performance of best performing known construction heuristics (presented in section II-B) and the best evolved heuristics for each of the problem instances. The table lists:

- A - the best performing manually derived heuristics.
- B - the hard constraint cost of the best performing manually derived heuristics.
- C - the soft constraint cost of the best performing manually derived heuristics.
- D - the best performing evolved heuristic.
- E - the hard constraint cost of the best performing evolved heuristic.
- F - the soft constraint cost of the best performing evolved heuristic.

From Table V it is evident that the evolved heuristics perform better than the manually derived heuristics, producing timetables with a lower soft constraint cost and hence requiring less further optimization. For problem instances comp02 to comp04, comp09, comp10, comp12, comp13, comp15 to comp17, comp19 and comp21, the saturation degree heuristic combined with the minimum penalty period heuristic performed the best in the analysis of the manually created construction heuristics. The best evolved heuristics for these problem instances also use the minimum penalty period heuristic and the sequence of problem characteristics begins with the saturation degree heuristic. The use of other problem characteristics to deal with ties differs for each of these problem instances, i.e. the most effective characteristics are problem instance specific, and has improved the results. For problem instance comp05 saturation degree together with random period produced the best results but was not able to achieve feasibility. The heuristics evolved on all ten runs produced feasible heuristics and all ten heuristics included the random period heuristic and the sequence of characteristics began with saturation degree. Different problem characteristics were found to be effective for dealing with ties on each of the ten runs. It is interesting to note that for the two problem instances where the largest enrolment heuristic produced the best results, namely, comp11 and comp20 the best evolved heuristic began with the unavailabilities as the first characteristic in the sequence of problem characteristics. There is a notable decrease in the soft constraint cost of the timetables produced by the evolved heuristics. For problem instance comp07 the best performing heuristic was the largest degree heuristic and the best evolved heuristic also applied the largest degree heuristic first followed by saturation degree and number of students to deal with ties. In the case of problem instance comp06 largest degree together with random period produced the best result, however the best performing evolved heuristic did not include saturation degree or random period. For instances comp14 and comp18 the best performing period selection heuristic is also included in the best performing evolved heuristic. However, the first characteristic in the problem characteristic sequence is not the same as the best performing heuristic, namely, saturation degree. In both these cases the saturation degree is used as a second characteristic to break ties with the primary problem characteristic in comp14 being the number of lectures and in comp18 the minimum number of working days.

V. CONCLUSION

The aim of the research presented in this paper was to investigate the automated derivation of construction heuristics for the curriculum based course timetabling problem. To the author/s knowledge this is the first examination of automated design of construction heuristics for the curriculum based course timetabling problem. Furthermore, in previous work automating the induction of construction heuristics, problem characteristics or existing/manually derived heuristics are combined with arithmetic and selection operators. This study investigates a different approach which uses genetic programming to combine problem characteristics and existing heuristics hierarchically and combining this sequence further with a period selection heuristic. Each heuristic is comprised of a sequence of length 2 or 3 of problem characteristics and a period selection heuristic. The first characteristic in the sequence is regarded as the primary characteristic and courses are sorted according to this value. The second characteristic is used to break ties in the case of courses which have the same value for the primary characteristic. Similarly, the third heuristic is used to break ties where courses have the same value for the second heuristic. The evolved heuristics performed better than the existing manually derived heuristics producing timetables with lower soft constraint costs.

It is interesting to note that for a number of the problem instances the primary characteristic in the best evolved heuristic was the same as the existing heuristic producing the best result for the problem instance. However, the remaining heuristics in the sequence used to break ties enabled the evolved heuristic to

perform better. It was also noted that the remaining heuristics in the sequence differed for each problem instance. For some problem instances the evolved heuristics producing the best results did not include any existing heuristic characteristics and the improvement in results can be attributed to the introduction of new characteristics which the existing heuristics did not cater for.

The study has also revealed how important it is to represent the characteristics of the problem domain as widely as possible. The characteristic of room availability for courses was represented as a static value, namely, the number of rooms of sufficient capacity for each course. However, it was found that this was not effective and instead a dynamic value of the number of rooms available at the current point of timetable construction per period would be more appropriate. It is not an easy task to determine the most suitable set of problem characteristics to use and this is one of the challenges of automated construction heuristic design. Currently this set is determined by trial and error. Future work will investigate this further, including identifying techniques to extract relevant problem characteristics.

For some problem instances the best performing evolved heuristic was the same. For two such instances the conflicts ratio per lecture was the same and this could be used as a criterion for reusing evolved heuristics. For the remaining 3 pairs of problem instances with the same best evolved heuristic it was not obvious what the similarity in problem instances are. This will be investigated further as part of future work. The aim is to find a technique that will determine the similarities of problem instances so that similar instances can be placed in classes and heuristics can be evolved per class rather than for each problem instance.

The main aim of this study was to test an alternative approach to combining heuristics and problem characteristics using genetic programming, hence the number of problem characteristics and the number of heuristics/characteristics combined have been kept small. For some instances exhaustive search possibly could have been used explore the search space but the aim was to firstly test GP to get an indication of the effectiveness of such combinations and the practicality of using genetic programming for this purpose. Future work will examine using more heuristics/characteristics, increasing the length of the problem characteristic sequence to greater than 3 and including a sequence of period selection heuristics, rather than a single period selection heuristic, which will be applied in the same way as the problem characteristics sequence. Other options for period selection heuristics will also be looked at. Furthermore, this approach to automated construction heuristic design using genetic programming will also be applied to other combinatorial optimization problems.

REFERENCES

[1] N. Pillay and W. Banzhaf, "An informed genetic algorithm for the examination timetabling problem," *Applied Soft Computing*, vol. 10, pp. 457–467, March 2010.

[2] A. Scholl, R. Klein, and C. Jurgens, "Bison: A fast bison: A fast hybrid procedure for exactly solving the one-dimensional bin packing problem," *Computers and Operations Research*, vol. 24, no. 7, pp. 626–645, 1997.

[3] R. Qu, E. Burke, B. McCollum, L. Merlot, and S. Lee, "A survey of search methodologies and automated system development for examination timetabling," *Journal of Scheduling*, vol. 12, pp. 55–89, 2009.

[4] J. Koza, Ed., *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, 1st ed. United States of America: MIT, 1992.

[5] E. Burke, M. Hyde, G. Kendall, and J. Woodward, "Scalability of evolved online bin packing heuristics," in *Proceedings of the Congress of Evolutionary Computation (CEC 2007)*. IEEE Press, 2007, pp. 2530–2537.

[6] J. Drake, M. Hyde, K. Ibrahim, and E. Ozcan, "A genetic programming hyper-heuristic for the multidimensional knapsack problem," *Kybernetes*, vol. 9, no. 10, pp. 1500 – 1511, 2014.

[7] N. Pillay and W. Banzhaf, "A study of heuristic combinations for hyper-heuristic systems for the uncapacitated examination timetabling problem," *European Journal of Operational Research*, vol. 197, pp. 482–491, 2009.

[8] N. Pillay, "Evolving hyper-heuristics for the uncapacitated examination timetabling problem," in *Proceedings of the 4th Multidisciplinary International Scheduling Conference: Theory and Application (MISTA 2009)*, 2009, pp. 409–422.

[9] M. Bader-El-Den, R. Poli, and S. Fatima, "Evolving timetabling heuristics using grammar-based genetic programming hyper-heuristic framework," *Memetic Computing*, vol. 1, pp. 205–219, 2009.

[10] A. Bonutti, F. De Cesco, L. Di Gaspero, and A. Schaerf, "Benchmarking curriculum-based course timetabling: formulations, data formats, instances, validation, visualization, and results," *Annals of Operations Research*, vol. 194, no. 1, pp. 59–70, April 2012.

[11] B. McCollum, P. McMullan, B. Paechter, R. Lewis, A. Schaerf, L. DiGaspero, A. Parkes, R. Qu, and E. Burke, "Setting the research agenda in automated timetabling: The second international timetabling competition." *INFORMS Journal of Computing*, vol. 22, no. 1, pp. 120–130, 2008.