A Study of the Practical and Tutorial Scheduling Problem

Nelishia Pillay

School of Mathematics, Statistics and Computer Science, University of KwaZulu-Natal

+27 33 2605644

pillayn32@ukzn.ac.za

Abstract: The practical and tutorial allocation problem is a problem encountered at tertiary institutions and essentially involves the allocation of students to practical or tutorial groups for the different courses the student is enrolled in. Practical and tutorial scheduling for first year courses is becoming more and more challenging as the number of permissible course combinations and student numbers increase at tertiary institutions, and while this has previously been done manually and independently for each course, this is no longer feasible. The paper firstly presents a formal definition of the practical and tutorial scheduling problem. Low-level construction heuristics for this domain are defined and a heuristic approach for solving this problem is proposed. A tool namely, PRATS, incorporating this approach is described. The performance of PRATS on six sets of real-world data is discussed. The paper also reports on a hyper-heuristic implemented to automatically generate low-level construction heuristics and compares the performance of the generated heuristics to the human intuitive heuristics used.

Keywords: educational timetabling, construction heuristics, practical and tutorial scheduling, hyper-heuristic

1. Introduction

Educational timetabling essentially encompasses university course timetabling (McCollum et al. 2008), university examination timetabling (Qu et al. 2009) and school timetabling (Pillay 2013). Initially, the practical and tutorial scheduling problem formed part of the university course timetabling problem, with a period or afternoon session set aside for the practical or tutorial for a course. However, with time one session was found not to be sufficient for certain courses and practical and tutorial allocations were done separately from course timetabling for these courses, with more than one session scheduled for certain courses and students choosing the session that would suit them best. These selections were done manually and usually independently for the different courses. As the number of permissible course combinations, students and hence practical/tutorial

sessions per course has increased at first year level, this is no longer feasible. As with the other types of educational timetabling which were initial done manually, as the complexity of these problems grew manual timetable construction was no longer an option and methods for automating the process were sought, this is now true of practical and tutorial allocations as well. The paper presents a description of the problem and a heuristic approach to solve this problem. Low-level construction heuristics are derived based on human intuition and evaluated for this purpose. A hyper-heuristic was implemented to automatically derive low-level construction heuristics for the domain and the evolved heuristics are compared to the human intuitive heuristics used to solve the problem. The proposed approach was used to solve this problem for the College of Agriculture, Engineering and Science at the University of KwaZulu-Natal. The performance of the heuristics on six sets of data corresponding to three semesters is presented and discussed.

The following section defines the practical and tutorial scheduling problem. Section 2 describes the heuristic approach proposed to solve the problem and the tool PRATS developed for use by administrators for practical and tutorial allocation. The genetic programming system implemented to evolve heuristics is explained in section 3. Section 4 discusses the performance of the heuristic approach and the evolved heuristics on the six real-world data sets. A summary of the findings of the study and future work is outlined in section 5.

2. The Practical and Tutorial Scheduling Problem (PTSP)

Science courses offered by the College of Agriculture, Science and Engineering have a three hour weekly practical or tutorial. Subjects such as Chemistry, Physics and Computer Science have a practical and courses like Mathematics and Statistics a tutorial. While the other educational timetabling problems, namely, school, university course and examination timetabling aims at scheduling events in timetable periods and venues, the practical and tutorial scheduling problem involves the allocation of practical/tutorial periods to each of the courses a student is registered for.

The practical/tutorial scheduling problem can be considered to be a version of the student sectioning problem (Muller et al., 2007; Muller et al., 2010; Murray et al., 2007) with the following differences:

- All students are pre-registered.
- The problem does not include the scheduling of lectures for students. This is done by the university administration using the block system and cannot be changed. The lectures are generally scheduled during the morning until lunch time and practicals and tutorials in the afternoon commencing at 14:10 (with one practical/tutorial timetable session in the morning at 09:35).
- Student preferences are not taken into consideration but rather department preferences. The department for each course specifies a set of practical/tutorial sessions for the course and students are allocated into the sessions so as to prevent clashes.
- Once the allocations to practicals/tutorials are made this can only be changed if the curriculum of the student changes resulting in a clash.

The student sectioning problem has essentially been solved by allocating students according to priorities, based on their preferences, and performing intelligent backtracking to resolve clashes (Muller et al. 2010). The research presented in this paper focuses on the derivation of heuristics for construction of solutions to this problem, both human intuitive and automatically generated.

The allocations for PTSP need to be made so as to ensure that:

- All courses for each student must be allocated a practical or tutorial period.
- The capacities for each practical/tutorial period must not be exceeded.
- Each student must not be scheduled to attend more than one practical in the same period, i.e. there must be no clashes.
- Grouping requirements - Students registered for certain degrees must attend practicals/tutorials in the same session for one or more courses. For example, all Engineering students must attend Physics on a Monday afternoon and Chemistry on a Tuesday afternoon.

Each year is comprised of two semesters and a practicial/tutorial schedule has to be created for each semester. The College has two campuses, with each campus requiring a different schedule. Each problem instance is defined in terms of:

- The courses offered by the College, practicals/tutorial sessions for each course and the capacity for each session. An example is illustrated in Table 1.

- The grouping requirements specifying the degrees and the session allocations for each course. Table 2 provides an example of this data, e.g. all Geological Sciences students (BSGLS) must have a Chemistry practical on a Tuesday, Geography on a Wednesday, a Mathematics tutorial on a Thursday and Geology on a Friday.
- Student registrations - Student number, degree, College (some schools in the College offer services courses for degrees offered by other Colleges) and courses for which each student is registered.

  The following section proposes a method for solving the PTSP.

Table 1. Example of Course Details

Course	Mon	Tues	Wed	Wed(am)	Thurs	Fri
BIOL101	186	186	186	186		
BIOL103						187
BIOL195				192		
CHEM110	96	192	192	192	192	120
CHEM195						72
COMP100					250	
GEOG110			184			
GEOL101	90				51	90
MATH130		408	378			
MATH140						72
MATH150		350		750		350
MATH195				60		
PHYS110			159			
PHYS131	240	320			320	
PHYS139	42					
PHYS195			25			
STAT130	350			350	350	

3. Solving the PTSP

This section firstly presents the low-level heuristics and method used to solve the practical and tutorial scheduling problem. This is followed by a description of the tool developed, employing the method outlined in section 3.1.

Table 2. Example of Group Requirements

Degree	Mon	Tues	Wed	Wed(am)	Thurs	Fri
B-MDSC		PHYS131	BIOL101		CHEM110	MATH150
BMDSCP		PHYS131	BIOL101		CHEM110	MATH150
BOPT	PHYS139	MATH150	CHEM110			BIOL103
B-PHAR	PHYS131	MATH150	CHEM110			BIOL103
B-PHYS		PHYS131				BIOL103
BS-ENS	PHYS131	CHEM110			GEOL101	MATH150
BSGLS		CHEM110	GEOG110	MATH150	PHYS131	GEOL101
BSCF	GEOL101					
BSCA	GEOL101					
BSLES	GEOL101					
BSAPC	GEOL101					
BSBLS	GEOL101					
BSCMB	GEOL101					

3.1 Heuristic Approach for Solving the PTSP

Low-level construction heuristics have proven to be effective in creating solutions to educational timetabling problems. These heuristics have essentially been used to order events, e.g. examinations, lessons, in order of difficulty to schedule. For example in the domain of examination timetabling graph colouring heuristics, e.g. largest degree, largest weighted degree, largest enrollment , largest colour degree and saturation degree (Carter et al., 1996) have been used. Usually, one heuristic is used to order examinations for scheduling. For example, the largest degree heuristic is the number of clashes that an examination can be involved in and a higher value indicates a more difficult exam to schedule. In some instances the low-level heuristic is used to create an initial solution which is further improved using optimization methods such as variable neighbourhood search, tabu search and evolutionary algorithms (Qu et al. 2009). In Carter et al. (1996) a low-level heuristic is used to create an initial timetable which is improved using backtracking. These graph colouring heuristics are also used for university course timetabling. Similar heuristics are used for timetable construction for the domain of school timetabling (Pillay, 2013). Given the crucial role that low-level construction heuristics have played in educational timetabling it was felt that the first step to solving the PTSP would be to define low-level construction heuristics for this domain.

The problem essentially involves assigning practical and tutorial sessions for each student. For this domain it was decided to view this as the student being the entity to assign and thus heuristics for assessing the difficulty of scheduling each student was investigated. This was defined in terms of the practicals/tutorials that had to be scheduled for the student. Two heuristics were defined for ordering students for practical/tutorial allocation:

- Allocation degree - the number of allocations for the student, e.g. if the student is registered for three courses that require practical/tutorial allocations the allocation degree will be 3. Priority is given to students with a higher allocation degree. It is anticipated that students requiring more allocations will be more difficult to schedule as there is a greater chance of clashes.
- Total opts - This heuristic is the sum of the options for each practical/tutorial that needs to be scheduled for the student. The reasoning behind this is that the less options there are available the more difficult it will be to schedule the practicals/tutorials for the student.

These heuristics are static and remain constant throughout the allocation process. These are calculated for each student at the beginning of the allocation process. In addition to heuristics for choosing which student to perform allocations for first, a low-level heuristic, namely option degree, for selecting which practical/tutorial to schedule first was also defined. This heuristic is the number of options, i.e. sessions, for each practical/tutorial. Practicals/tutorials with fewer options are given priority to be scheduled. The final low-level heuristic defined is to choose the session to schedule the practical in. The session with the higher capacity amongst the feasible sessions, i.e. sessions with sufficient capacity that do not result in clashes, is chosen. Both these heuristics are dynamic and need to be updated during the allocation process as allocations will change the number of options and capacities. Table 3 provides a summary of the low-level heuristics.

Table 3. Low-level Heuristic Summary

Category	**Heuristics**	**Type**
Student allocation heuristics	• Allocation degree • Total opts	Static
Course heuristics	• Option degree	Dynamic
Session heuristics	• Capacity degree	Dynamic

The allocation process begins by sorting the students according to one of the student allocation heuristics. The allocation for each student starts with sorting the practicals/tutorials to be scheduled according to the option degree. A period is assigned to the practical/tutorial with the lowest option degree using the capacity degree heuristic. If the degree the student is registered for and practical/tutorial is one of the group requirements, the specified session is allocated. The allocated practical/tutorial is removed and the list to be scheduled resorted according to the option degree as the value of the option degree may change in some cases as a result of the allocation. If a period cannot be found as a result of insufficient capacity or if the allocation will result in a clash, the allocation is not made and the number of unallocated practicals/tutorials is incremented. A list of clashes and insufficient capacities for the different courses is maintained and reported together with the number of unallocated practicals/tutorials at the end of the allocation process. The following section describes the tool developed to be used by administrators for practical/tutorial allocation. The algorithm is illustrated in Figure 1.

3.2 PRATS (Practical and Tutorial Scheduler)

The importance of working with administrators that will be using the system and bridging the gap between research and practice is emphasized in McCollum (2007). Thus, the tool implementing the method describe in section 3.1 for practical/tutorial allocation was developed in consultation with the College Director: College Professional Services and the College Academic Services. Prior to the implementation of PRATS the practical/tutorial allocation for first year was done manually. The schedule was stored in an Excel spreadsheet which was given to administrators to update in cases where students wished to change allocations as a result of any changes in curriculum for the particular student.

```
begin
 for 1 to no_of_students
  Sort practicals/tutorials to be allocated
   while(there are practicals/tutorials to allocate)
    if the practical/tutorial is part of  a group requirement
     allocate specified session
    else
     sort the feasible sessions for allocation
     schedule the practical/tutorial in the session with the maximum capacity
     if  a session cannot be found for the practical/tutorial
      increment the number of unallocated practicals/tutorials
      update the insufficient capacities/clashes list
     endif
   endif
  endwhile
endfor
 Report the number of unallocated practicals/tutorials
 Report insufficient capacities
 Report clashes
end
```

Figure 1. Practical/tutorial allocation algorithm

The spreadsheet contained:

- The courses and capacities for each practical/session given the current allocations.
- Practical and tutorial allocations for the students registered for first year Science modules in the College.
- Formulae linking the student allocations and capacities for each practical/tutorial session so that any de-allocation/reallocation of practicals/tutorials for a particular student results in the capacity for the session/s involved being automatically updated.

It was requested that the software developed should solve the practical and tutorial allocation problem and output a spreadsheet with the same format as the that of the spreadsheet schedule that was previously created manually. The tool

PRATS was developed for this purpose. Input to for each problem instance includes:

- A spreadsheet of all student registrations for first year Science courses in the College (DMI file).
- A spreadsheet listing all first year Science courses offered by the College and the practical/tutorial sessions for each course and corresponding capacities.
- The spreadsheet listing the group requirements for specific degrees (Groups file).

Figure 2 illustrates a PRATS session. The user is required to load the input files and choose the option to create a schedule, specifying the file the schedule must be stored in.

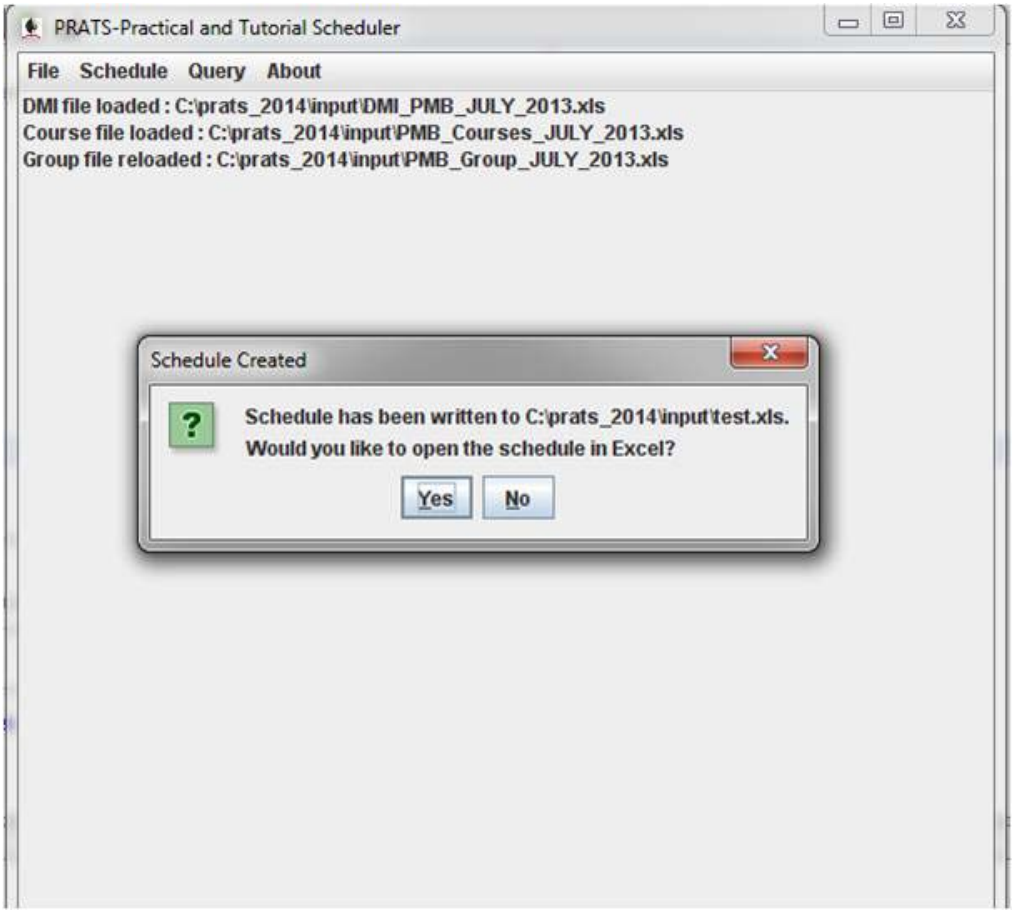

Figure 2. Example of a PRATS session

PRATS produces an Excel spreadsheet file with the following tabs:

- The first tab lists the courses, practical/tutorial sessions and capacities as well as the allocations for all students and includes formulae linking student allocations and student capacities.

- The second tab lists all the courses and additional capacity needed for courses where the number of students registered exceeds the total capacity of all sessions for the course practical/tutorial.
- The third tab list the clashes that cannot be resolved due to a student being registered for two courses that have the same practical/tutorial session with no other options, and the number of students with this clash. An example is illustrated in Table 4. This usually results when students are registered for incorrect combinations. Another reason for this is that the combination was not anticipated, in which case the practical/tutorial for one of the sessions may have to be rescheduled. The subsequent tabs in the spreadsheet list the students for each clash so that they can be contacted and informed. An example is illustrated in Table 5.

Table 4: Example of Spreadsheet Tab Listing Courses with Practical/Tutorial Clashes

Clash	First Course	Second Course	Period	No. of Students
Clash1	MATH196	COMP102	Thurs	1
Clash2	STAT140	COMP106	Wed(am)	3
Clash3	PHYS120	BIMI120	Wed	5
Clash4	MATH144	MATH130	Fri	4
Clash5	ENVS120	BIMI120	Wed	18

Table 5: Example of Student Data for a Clash

Student Number	Surname	Name	Degree
212500323	Mhlongo	Slungile Sunshine	BSCA
213510964	Wanda	Ayanda	BSAPC
212540361	Mvelase	Bonokwakhe Sthembiso	BSCA

The following section describes the performance of PRATS on six sets of data for the College of Agriculture, Engineering and Science.

4. PRATS Performance

During the development phase of PRATS it was tested on data for the first semester of 2013 for which manual schedules had been created. Details of the data sets are listed in Table 6.

Table 6: Data Sets for Semester 1 2013

Data Set	No. of Courses	No. of Sessions	No. of Students	No. of Group Requirements
PMB_S1_2013	18	6	1223	0
WSTVL_S1_2013	17	6	2771	28

It was found that the application of each of the student allocation heuristics separately was not effective. The concept of primary and second heuristics used in Pillay et al. (2009) was introduced in this study. The students to be allocated are firstly sorted for scheduling using the primary heuristic. In the case of ties a secondary heuristic is applied to the tied students to determine the ordering. The combination of the allocation degree as the primary heuristic and total opts as a secondary heuristic was found to be the most effective.

PRATS allocated all practicals/tutorials for all students within the specified capacities for PMB_S1_2013. A clash between two courses, namely Nutrition and Geography with two students registered for this combination was reported. This combination is not permitted as generally Dietetics students register for Nutrition and Geography does not form part of the programme. All student practical/tutorials were also scheduled for WSTVL_S1_2013. The capacity for BIOL103 was exceeded by 1, however it was found that the student in question was registered for both BIOL101 (which was the module required for the programme the student was registered for) and BIOL103. The student was advised to deregister from BIOL103. Three clashes were reported by PRATS. The combinations causing the clashes were not permitted and the students notified.

PRATS was used to schedule the first year practicals/tutorials for the second semester of 2013. The details of both problem instances are listed in Table 7.

Table 7: Data Sets for Semester 2 2013

Data Set	**No. of Courses**	**No. of Sessions**	**No. of Students**	**No. of Group Requirements**
PMB_S2_2013	22	6	1483	5
WSTVL_S2_2013	20	6	2939	9

PRATS performed all the required allocations for PMB_S2_2013 within the specified capacities. Two clashes resulting from illegal combinations was reported. PRATS was not able to allocate the practicals/tutorials for all students for WSTVL_S2_2013. Further investigation revealed that the practical/tutorials not scheduled were those that had only one session option. As students with the highest number of allocations required were given priority for scheduling, this resulted in those students with courses having just one session for the practical/tutorial being scheduled later in the allocation process at which stage

there was insufficient capacity for these allocations. This resulted in the introduction of a third student allocation heuristic, namely, the one-option heuristic which gives students with at least one course with one practical/tutorial session priority for allocation. The combination of one-opt as the primary heuristic and allocation degree as the secondary heuristic resulted in all practicals/tutorials being scheduled. Four clashes resulting from students registering for illegal combinations were reported and the students notified.

PRATS was also used to generate schedules for the first semester of 2014. The data sets for both campuses are presented in Table 8.

Table 8: Data Sets for Semester 1 2014

Data Set	**No. of Courses**	**No. of Sessions**	**No. of Students**	**No. of Group Requirements**
PMB_S1_2014	13	6	1436	1
WSTVL_S1_2014	12	6	3548	14

Both the heuristic combinations, i.e. allocation degree as a primary heuristic and total opts as the secondary heuristic and one-option as a primary heuristic and allocation degree as a secondary heuristic are able to allocate all practicals and tutorials for all students. For PMB_S1_2014 one illegal course combination was found and the student notified while there were no clashes for WSTVL_S1_2014.

The runtime of the heuristic approach employed by PRATS is on average a second for all problem instances. Given that this study has revealed that the most appropriate combination of student allocation heuristics used is problem dependent, the option of automating the generation of the student allocation heuristic to use was investigated. This essentially involves implementing a hyper-heuristic for the generation of low-level construction heuristics (Burke et al. 2013). As the best heuristic to use is problem dependent, the generated heuristic is disposable. From previous work done in this domain, it is evident that genetic programming has chiefly been employed for the induction of construction low-level heuristics (Burke et al. 2009; Burke et al. 2013). A genetic programming hyper-heuristic was implemented and tested on the six problem instances. The following section describes the hyper-heuristic and discusses its performance on the six data sets.

5. Generative Constructive Hyper-Heuristic

This section describes the hyper-heuristic used to evolve low-level construction heuristics for the practical and tutorial scheduling problem. Section 5.1 describes the genetic programming system implemented and section 5.2 discusses the performance of the hyper-heuristic in solving the PTSP.

5.1 Genetic Programming Hyper-Heuristic

The hyper-heuristic employs genetic programming to evolve low-level construction heuristics. Genetic programming is an evolutionary algorithm that explores a program space to identify a program which when executed will produce an optimal or near optimal solution (Koza 1992). The generational genetic programming algorithm illustrated in Figure 3 was implemented.

```
Create initial population
Repeat
  Evaluate the population
  Select parents
  Apply genetic operators to parents
Until the termination criteria are met
```

Figure 3. Genetic programming algorithm

Each element of the population is a parse tree representing a low-level construction heuristic. Each parse tree is comprised of elements from the function the terminal sets. The function includes the following operators:

- Arithmetic operators: +, -, *, /. The division operator is protected division which returns a value of 1 if the denominator is zero.
- The *if* operator which performs the standard if-then-else function and has an arity of 3.
- Arithmetic logical operators: <, >, <=, >=, ==, !=. These operators perform the standard arithmetic logical operations. These operators can only be included in the subtree representing the first child of the *if* operator as this is the branch of the tree representing the condition that must be met.

The terminal set is comprised of the low-level student allocation heuristics defined in section 3.1:

- a - Represents the allocation degree heuristic.

- b - Represents the one-option heuristic.
- c - Represents the total opts heuristic.

Figure 4 illustrates examples of population elements. Each element is created by firstly choosing an element from the function set to ensure that trivial trees are not created. The tree is constructed using the grow method (Koza 1992) which involves the random selection of elements from the function and terminal set until the maximum specified depth is reached at which point only elements from the terminal set are chosen.

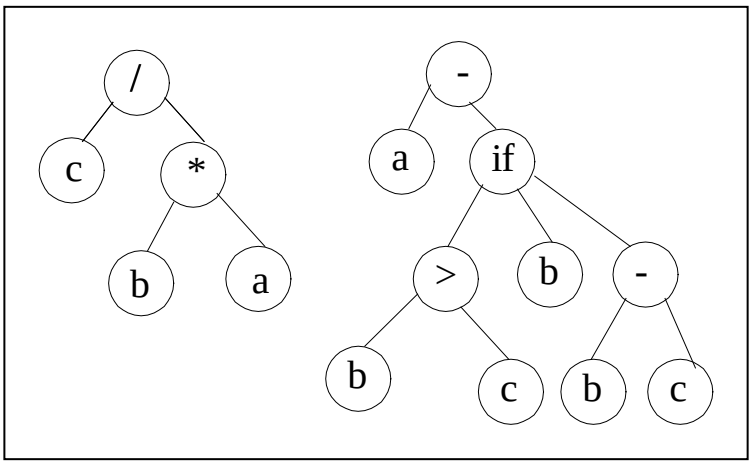

Figure 4. Examples of parse trees representing low-level construction heuristics

Each individual is evaluated by using the individual to create a schedule. This achieved by applying the heuristic to each student producing a numerical value/heuristic which represents the difficulty of scheduling the practicals/tutorials for that student. The students are sorted in descending order according to the heuristic and the allocations for each student performed in order. The fitness of an individual is the number of unallocated practicals and tutorials in the created schedule.

Tournament selection (Koza 1992) is used to choose parents for regeneration. This selection method essentially involves randomly selecting *t* elements of the population and returning the fittest individual as a parent. Selection is with replacement so an individual can be chosen as a parent more than once.

The standard mutation and crossover operators (Koza 1992) are used for regeneration. The mutation operator replaces a randomly selected subtree in the copy of the parent with a newly created subtree. Crossover randomly selects a subtree in each of two parents and the subtrees are swapped to create two offspring.

Performance of the hyper-heuristic on the six data sets is discussed in the following section.

5.2 Hyper-Heuristic Performance

The parameter values used are listed in Table 9. These values were determined empirically.

Table 9. Parameter values

Parameter	**Values**
Population size	50
Maximum tree depth	4
Tournament size	4
Crossover %	50%
Mutation %	50%
Mutation depth	3

Due to the stochastic nature of genetic programming ten runs were performed for each problem. The average runtime for the hyper-heuristic is five seconds. All runs performed were successful at producing schedules with all practicals and tutorials for all students allocated, within the specified capacities, for the six problem instances. Furthermore, heuristics producing the optimal schedule were found in the initial population and further optimization was not needed. The heuristics evolved for each seed were different and there did not appear to be any similarities between the evolved heuristics producing optimal solutions. This will be researched further as part of future work.

6. Conclusion and Future Work

This paper introduces the practical and tutorial scheduling problem and a heuristic approach to solve this problem. Five low-level construction heuristics have been defined for this problem based on human intuition and categorized as student allocation heuristics, course heuristics and session heuristics. The study revealed that it was necessary to combine the student allocation heuristics as primary and secondary heuristics in order to find a solution to the problem. The heuristic approach was incorporated into a tool PRATS which was able to find feasible solutions to six real-world problem instances. It was also found that different

student allocation heuristic combinations may be needed to find solutions to different problems. This led to the implementation of a hyper-heuristic to automatically generate a student allocation heuristic. This approach appeared to be effective, finding solutions to all six problems. Future work will investigate further examination of the generated heuristics to identify possible patterns or similarities in the functions. In addition to this the reusability of the optimal heuristics generated will also be investigated.

Acknowledgements. This work is based on the research supported in part by the National Research Foundation of South Africa for the Grant CSUR13091742778. Any opinion, finding and conclusion or recommendation expressed in this material is that of the author(s) and the NRF does not accept any liability in this regard.

7. References

Burke, E.K., Hyde, M., Kendall, G., Ochoa, G., Ozcan, E. &Woodard, J. (2009) Exploring Hyper-Heuristic Methodologies with Genetic Programming. *Computational Intelligence*, 6, 177-201.

Burke, E.K., Hyde, M., Kendall, G., Ochoa, G., Ozcan, E. & Qu, R. (2013) Hyper-Heuristics: A Survey of the State of the Art. *Journal of the Operational Research Society*, 1-30.

Carter MW, Laporte G, Lee SY (1996) Examination Timetabling: Algorithmic Strategies and Applications. *Journal of the Operational Research Society*, 47(3), 373-383.

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

McCollum, B. (2007) A Perspective on Bridging the Gap between Research and Practice in University Timetabling. Practice and Theory of Automated Timetabling VI (eds. E.K.Burke and H.Rudova), Lecture Notes in Computer Science Volume 3867, Springer 2007, pp 3-23.

McCollum, B., McMullan, P., Paechter, B., Lewis, R., Schaerf, A., DiGapsero, L., Parkes, A.J., Qu, R . & Burke, E.K. (2008). Setting the research agenda in automated timetabling: The second international timetabling competition. *INFORMS Journal of Computing*, 22(1), 120–130.

Muller, T. & Murray, K. (2010) Comprehensive Approach to Student Sectioning. *Annals of Operational Research*, 181, 249-269.

Muller, T., Murray, K. & Schluttenhofer, S. (2007) University Course Timetabling and Student Sectioning System. In Proceedings of the International Conference on Automated Planning and Scheduling, pp. 1-4.

Murray, K. & Muller, T. (2007) Real-Time Student Sectioning. In Proceedings of the 3rd Multidisciplinary International Scheduling: Theory and Applications, pp. 1-3.

Pillay, N. & Banzhaf, W. (2009) A Study of Heuristic Combinations for Hyper-Heuristic Systems for the Uncapacitated Examination Timetabling Problem. *European Journal of Operational Research*, 197, 482-491.

Pillay, N. (2013) A Survey of School Timetabling. Annals of Operations Research, February 2013, DOI: 10.1007/s10479-013-1321-8.

Qu, R., Burke, E.K., McCollum, B., Merlot, L.T.G. & Lee, S.Y. (2009) A Survey off Search Methodologies and Automated System Development for Examination Timetabling. *Journal of Scheduling*, 12(1), 55–89.