10th International Conference of the Practice and Theory of Automated Timetabling
PATAT 2014, 26-29 August 2014, York, United Kingdom

Directed Selection using Reinforcement Learning for the Examination Timetabling Problem

Ryan Hamilton-Bryce · Paul McMullan · Barry McCollum

Abstract Traditional heuristic approaches to the Examination Timetabling Problem normally utilize a stochastic method during Optimization for the selection of the next examination to be considered for timetabling within the neighbourhood search process. This paper presents a technique whereby the stochastic method has been augmented with information from a weighted list gathered during the initial adaptive construction phase, with the purpose of intelligently directing examination selection. In addition, a Reinforcement Learning technique has been adapted to identify the most effective portions of the weighted list in terms of facilitating the greatest potential for overall solution improvement. The technique is tested against the 2007 International Timetabling Competition datasets with solutions generated within a time frame specified by the competition organizers. The results generated are better than those of the competition winner in seven of the twelve examinations, while being competitive for the remaining five examinations. This paper also shows experimentally how using reinforcement learning has improved upon our previous technique.

1 Introduction

The challenge of producing acceptable solutions for timetabling problems such as Course and Examination timetabling involves a combination of practical and research based approaches [1]. Due to the complexity of the underlying problems and the potential time requirement in providing acceptable solutions to these problems through the use of discrete methods, over this last few decades research has focused on the use of search based heuristic techniques. A number of review papers on the subject have been published [2], [3]. As

School of EEECS, Queen's University of Belfast, BT7 1NN, Northern Ireland
E-mail: {rhamiltonbryce01, p.p.mcmullan, b.mccollum}@qub.ac.uk

with Course timetabling, progress in research within the area of examination timetabling has been facilitated by the availability of benchmark data sets [4], [5]. Results generated using a wide range of techniques have been reported, with varying levels of success based on both generality of the solver and the time taken to solve [2]. A successful technique can be viewed as one which can produce feasible and workable solutions to a range of differing problems for a given problem domain within a practical timescale.

An examination scheduling track, based on the post-enrolment examination timetabling problem was introduced in the second International Timetabling Competition (ITC2007) [4]. This track introduced a number of real world datasets, drawn from anonymised data from several institutions worldwide. New result sets continue to be validated using the competition's online validation service despite the competition closing almost five years ago. The next timetabling competition to be announced will further develop the problem definition, to further extend the real world aspects of research in this area and to encourage innovation with the development of new problem solvers [6].

Solving the examination timetabling problem generally takes the form of an initial construction phase to produce a feasible solution, and an improvement phase which employ a number different search techniques to find high quality solutions when given specific objectives [7]. It has been observed that for certain construction techniques if construction is continued beyond the point at which a feasible solution has been acquired it is often possible to acquire a better quality for solution on which the improvement phase can operate [8].

Traditionally heuristic based approaches to timetabling problems have utilized a stochastic method for selection of the examination within a neighbourhood search process [9], [10], [11]. This allows for a rapid selection of examinations for the optimization process. It has been shown through experimentation that a link exists between the phases of construction and optimization [7]. It is possible to exploit this link to allow for a useful transfer of information between the phases

Directed Selection Optimization (DSO) exploits co-operation between the phases of construction and optimization. Information gathered and used, in the form of a weighted list, during the construction phase is used to influence and direct examination selection within the subsequent improvement phase. In the improvement phase the weighted list is split into portions, and using reinforcement learning techniques, the portions which show the greatest potential for improvement are preferentially used to influence examination selection. Highest Soft Constraint Optimization (HSCO) is a new optimization heuristic, where examination selection is directed by a weighted list, the values of which are calculated based on an examinations individual soft constraint penalty. Optimization occurs in order from the examination with the highest to lowest penalty.

The remainder of the paper is as follows: Section 2 briefly describes the examination timetabling problem. Section 3 describes the Squeaky Wheel constructor, Directed Selection Optimizer and the Highest Soft Constraint Cost Optimizer. Section 4 describes the experimental environment and time pa-

rameters used during experimentation. Section 5 presents and discusses the results and finally section 6 concludes the paper with a brief discussion on the effectiveness of the technique and potential future research areas

2 The Examination Timetabling Problem

Examination timetabling is a subset of the general timetabling problem, and has been proven to be NP-hard [12]. Examination timetabling involves allocating a set of events (exams), into a number of available resources (timeslots and rooms), subject to a series of constraints. Primarily, there are two types of constraints; hard and soft. Hard constraints must be satisfied for a timetable to be considered feasible, for example an exam cannot be scheduled in a room that is too small for the size of the exam, or a student must not have two exams at the same time. Soft constraints on the other hand represent desirable preferences, which are not required to be satisfied for the timetable to be considered feasible, but may affect the fitness or quality of the resultant solution. For example, while it may not be preferable for a student to have two exams in one day, a timetable can still be considered feasible if this does occur. The main goal when solving this problem is to minimize the number of soft constraint violations, while at the same time maintaining a feasible solution. As it is possible to assign a numeric value to the quality of a timetable based on how well it satisfies the various constraints, it is possible to directly compare two timetables, where the timetable with the lower overall penalty is considered the more acceptable solution.

Examination timetabling, unlike Course Timetabling, is overwhelmingly considered to be a post-enrolment problem. Student enrolment data is generally known at the time of scheduling, allowing for an accurate use of the available resources during the examination period.

The requirements for real-world examination timetabling problems are often unique for each individual institution, with the type and mix of hard and soft constraint options reflecting the preferences of the institution in question. However it is possible to identify a common set of both hard and soft constraints for benchmark and research use.

Carter, et al. introduced a set of 13 benchmark examination datasets in 1996 [5], drawn from three Canadian high schools, five Canadian universities, one American university, one British university and one university in Saudi Arabia. These datasets have been widely tested and used in examination timetabling research [2]. These datasets were supplemented by a series of new datasets, drawn from anonymised data provided by several institutions worldwide, for the 2007 International Timetabling Competition (ITC2007).

	Exams	Students	Periods	Rooms	Conflict Density	Period Hard Constraints	Room Hard Constraints
Exam 1	607	7891	54	7	5.05%	12	0
Exam 2	870	12743	40	49	1.17%	12	2
Exam 3	934	16439	36	48	2.62%	170	15
Exam 4	273	5045	21	1	15.00%	40	0
Exam 5	1018	9253	42	3	0.87%	27	0
Exam 6	242	7909	16	8	6.16%	23	0
Exam 7	1096	14676	80	15	1.93%	28	0
Exam 8	598	7718	80	8	4.55%	20	1
Exam 9	169	655	25	3	7.84%	10	0
Exam 10	214	1577	32	48	4.97%	58	0
Exam 11	934	16439	26	40	2.62%	170	15
Exam 12	78	1653	12	50	18.45%	9	7

Table 1 ITC 2007 Dataset Information

Table 1 lists the main characteristics for each of the examination datasets provided by the organizers of ITC2007. The conflict density is a measure of the number of examinations that are in conflict due to student enrolment, defining how tightly the problem is constrained by student module choice. It is initially observed that the conflict density for most of the datasets is quite low, which is reflective of the amount of choice available to students within a modern curriculum, with a large variation in course or subject choices between each student. The measure of problem size, based on the number of exams and students, varies across the datasets. The largest exam dataset could be argued to be either Exam 3/Exam 11 or Exam 7 and the smallest to be either Exam 9 or Exam 12. The amount of periods and rooms available will also have a measurable effect on the difficulty of constructing a feasible solution. Exam 3 and Exam 11 are almost identical, however Exam 11 has a much smaller set of period resources available. The differences between Exam 3 and Exam 11 reflect a "real-world" situation where an examination session has been shortened to minimize space and staff costs, while keeping all other existing constraints where possible.

Recent attempts to solve the examination timetabling problem continue to involve a variety of different techniques. Genetic Algorithms [13] are modelled on Darwins theory of evolution. Once an initial population has been constructed, it is refined over a series of iterations, with an evaluation function calculating the fitness of each individual within the population. Late Acceptance Hyper-heuristics were introduced by Burke and Bykov [14]. Traditionally, the approach in hyper-heuristics was to compare the current solution with the solution immediately preceding within the neighbourhood search process. In late acceptance, the current solution is compared with what was the current solution a number of iterations previously. Late acceptance techniques are able to produce competitive results in a short timeframe. Reinforcement Learning [15] techniques are used to influence heuristic selection for hyper-heuristics. A memory log of heuristic actions is kept during execution, with successful actions being rewarded and unsuccessful actions being punished. With this log, successful actions are chosen more often and unsuccessful actions are chosen less often across the search space. Both long term [15] and short term [16] memories have been explored in this technique. Tabu Search [17], [18] is a local search based technique. Unlike other such techniques, it maintains a list

of solutions that have recently been visited, which is used to prevent the optimizer from repeatedly considering similar neighbourhoods, helping to avoid local optima. Hill Climbing is the simplest local search algorithm, introduced by Appleby in 1961 [19]. In Hill Climbing a candidate solution is only accepted if it has a better or equivalent fitness to the current one. Hill Climbing aims to converge quickly, but often has a final solution of relatively poor quality as it tends to get trapped in local optima. Simulated Annealing was introduced as a general optimization technique by Kirkpatrick, et al in 1983 [20]. Simulated Annealing is broadly similar to hill-climbing, however the technique is able to accept worse solutions through the use of a probability function and decreasing temperature parameter. Great Deluge was introduced by Dueck [21] in 1990 as a faster alternative to Simulated annealing. Great Deluge uses a boundary condition, rather than a probability function for the acceptance of worse solutions. In Great Deluge the boundary is initially set slightly higher than the initial solution, and is reduced gradually throughout the improvement process. The Extended Great Deluge was introduced by McMullan [22] for Course Timetabling, and later for Examination Timetabling [23]. The Extended Great Deluge algorithm adds a reheat mechanic similar to that employed in Simulated Annealing, where after a period of non-improvement the Great Deluge algorithm would self-terminate, the Extended Great Deluge employs a reheat function to widen the boundary condition to allow for the further acceptance of worse solutions in an attempt to escape local optima. Traditional problem solvers have primarily been implemented as single threaded applications. Modern desktop and server hardware are highly optimized for parallel workloads, and previously implemented solvers no longer take full advantage of the available hardware when executed on these machines. Ant Algorithms, introduced by Dorigo [10] and implemented for the examination timetabling problem by Eley [24], were among the first parallel implementations to solve the problem. Each ant works concurrently and independently to build a complete, or partial solution starting from an initial state defined by problem dependent criteria. The Scatter Search meta-heuristic has recently been implemented to execute in a parallel and distributed manner [25] over a series of independent servers.

3 Directed Examination Selection

Directed Selection, introduced in[26], is extended here to encompass a three phase process, building upon elements used in the Extended Great Deluge (EGD) algorithm introduced by McMullan [22]. The first phase is a Squeaky Wheel (adaptive) constructor, which is used to construct a series of initial timetables. Once construction has completed the best timetable and the weighted list used during construction is passed into the Directed Selection Optimization (DSO) phase. DSO utilizes the weighted list to influence the selection of the examination for optimization. After a number of non-improving reheat actions, the current best timetable is passed to the Highest Soft Constraint

Cost Optimization phase. When complete, the timetable is returned to the DSO phase while there is remaining execution time.

```
Start timer;
Read examination file;
Build clash information;
Squeaky Wheel Construction;
while time remaining do
    Directed Selection Optimization;
    Highest Soft Constraint Optimization;
end
Output results;
```

Algorithm 1: Sequence of Execution

3.1 Squeaky Wheel Construction

Squeaky Wheel (adaptive) construction [27] is an iterative construction process, building an initial schedule by placing one exam at a time, in the order determined by a weighted sequence. There are a number of different methods for determining the initial order of the weighted list, the technique presented here calculates the initial ordering based on examination size and the number of conflicts. Each exam is assigned to the first available time and room combination, where possible, ensuring that a feasible solution is maintained while minimizing soft constraint violations. If an exam cannot be scheduled in the current iteration, it is left unscheduled and the constructor moves onto the next exam. When an exam is scheduled a weighting based upon its current penalty, as defined by the various soft constraint violations, is added to the stored weighting in the weighted list. If an exam cannot be scheduled a suitably large weighting is used instead. Once an attempt has been made to schedule all exams, the weighted list is re-sorted and subsequently those exams with the highest weighted value (or most difficult exams) are first to be scheduled on the next iteration or "run" of the constructor. The weightings held in the list evolve over the duration of the entire construction process.

```
Read in the problem file into memory and build conflict and suitability matrices;
Calculate an initial weighting based on pre-defined criteria;
while stopping criteria not met do
    foreach exam ei in weightedList do
        for all suitable timeslots ti of ei do
            if CanSchedule(ei, ti) then
                best penalty and store best (bestti);
            end
        end
        if multipleBest found then
            Schedule(ei, randomBestti) and store associated weighting in
            weightedList;
        end
        else if bestTi found then
            Schedule (ei, bestti) and store associated weighting in weightedList;
        end
        else
            Leave exam unscheduled and add large weighting in weightedList;
        end
    end
    Sort(weightedList);
end
```

Algorithm 2: Squeaky-wheel (Adaptive) construction

3.2 Directed Selection Optimization

Directed Selection Optimization (DSO) is a new technique introduced by Hamilton-Bryce, McMullan and McCollum [26]. It is based on the premise that there exists a link where useful information can be passed from an adaptive based construction phase to an EGD based optimization phase [7]. It was shown in [26] that information which is traditionally discarded during the construction phase can be fed in to the optimization phase to influence and direct the selection of examinations for the neighbourhood search process. In Directed Examination Selection, the traditionally random selection of examinations is augmented with a portion of the weighted list generated during construction. After an initial learning period, reinforcement learning is used to influence examination selection to the portion(s) of the weighted list which show the greatest amount of improvement.

For example, the weighted list can be split into quarters. After the initial learning period the reinforcement learning list has the values (1, 4, 10, 7) for the first, second, third, and fourth quarters respectively, where 0 represents no improvement, and higher values represent greater amounts of improvement. While the highest value represents the potential for the greatest amount of improvement, it should not be used exclusively. Correspondingly while the lowest value represents the least potential for improvement, it should not be excluded from use. Therefore an element of chance should be introduced by simulating dice rolls, with the highest improving portion having approximately a 50% chance of being used, with each next portion of the weighted list having

a lesser chance. Finally if no portion of the weighted list has been selected then a number of random examinations equal to the portion size of the weighted list are selected. Through experimentation it was found that sorting the reinforcement learning list approximately every ten seconds provided good performance while also ensuring that the values in the list do not become 'stale' as the improvement process continues. As in [26], for performance reasons the weighted list is sorted during a reheat action. It is possible to sort the reinforcement learning list more often than the weighted list, due to the relative size differences between the lists. During initial experimentation it was found that it is possible to over-influence the selection of examinations to the detriment of the optimization process. This is prevented by applying a simulated 'die roll' to determine whether to use specific portions of the weighted list, or default to selecting examinations at random.

```
Sort Reinforcement Learning List (rlList);
if rnd.Next (1,6) >= 3 then
    Use best portion of weightedList;
else if rnd.Next (1, 6) >= 4 then
    Use next best portion of weightedList;
else if rnd.Next (1,6) ¿= 5 then
    Use next best portion of weightedList;
else if rnd.Next (1,6) == 6 then
    Use next best portion of weightedList;
else
    Select random exams for optimization;
end
```

Algorithm 3: Influencing Examination Selection

Once examination selection has occurred, the remainder of the optimization process is similar to the EGD algorithm. On each iteration, one of two neighbourhood heuristics is selected; either move or swap, and an attempt is made to apply the chosen heuristic to the selected examination list. In the new technique, boundary acceptance has been replaced with Late Acceptance Criteria. Late Acceptance Criteria was introduced by Burke and Bykov [28] [14], wherein a candidate solution is compared for acceptance against the current solution a number of iterations previously. This can be implemented as a simple queue structure of a predefined size, wherein the current solution is compared against the head value. At each iteration the head value is removed, if the candidate solution is accepted its cost is inserted onto the end of the queue, and if the candidate solution is rejected the last accepted cost is inserted onto the end of the queue. At the beginning of the optimization process the entire queue is initialised to the value of the initial cost function of the timetable undergoing optimization. The reheat mechanic from the EGD algorithm has been retained, to allow the algorithm an attempt to escape from local optimum conditions. Finally as with the EGD algorithm, the process will self-terminate

when a lack of improvement has been observed for a specified number of reheats of the late acceptance list.

```
Set the initial solution s using a construction heuristic;
Calculate initial cost function f(s); Initialize Late Acceptance list (laList);
while stopping criteria not met do
    Select portion of weightedList (optList) to use based on Reinforcement Learning
    criteria or a random exam;
    Select neighbourhood Heuristic S*;
    for all exams in optList do
        Apply S* on exam;
        Calculate f(s*);
        if f(s*) <= f(s) or f(s*) <= laList.FirstItem then
            Accept s = s*;
            Add new f(s) to laList;
            Update Reinforcement Learning Criteria with success;
        else
            Add existing f(s) to laList;
            Update Reinforcement Learning Criteria with fail;
    end
    if no improvement in given time T then
        Increase all values in laList by 10%;
end
```

Algorithm 4: Directed Selection Optimization (DSO)

3.3 Highest Soft Constraint Optimization

Highest Soft Constraint Optimization (HSCO) is a new optimization heuristic introduced here influenced by the Highest Cost construction heuristic introduced by Pillay and Banzhaf [29] [13]. The Highest Cost construction heuristic calculates the soft constraint cost of scheduling an examination given the current state of the timetable and the examination with the highest cost is scheduled first. In HSCO, the soft constraint penalty for each examination in the timetable is calculated. An attempt is then made to optimize the timetable in order from the examination with the highest soft constraint cost. As with the previous optimization phase, HSCO has been implemented with Late Acceptance criteria for boundary acceptance. During initial experimentation it was found that the neighbourhood swap heuristic used during the DSO phase resulted in degraded performance when used in the HSCO phase as it resulted in an examination being revisited multiple times during the search process. As such the HSCO phase contains only a neighbourhood move heuristic.

```
Set the initial solution s using a construction heuristic;
Improve the initial solution s using DSO;
Calculate cost function f(s);
Initialize Late Acceptance list (laList);
while stopping criteria not met do
    for all exams in timetable do
        Calculate soft constraint penalty on individual exam basis;
        Store penalty in scList;
    end
    Sort scList by penalty;
    for all exams in scList do
        Apply neighborhood move on exam;
        Calculate f(s*);
        if f(s*) <= f(s) or f(s*) <= laList.FirstItem then
            Accept s = s*;
            Add new f(s) to laList;
        else
            Restore last best s;
            Add existing f(s) to laList;
        end
    end
end
```

Algorithm 5: Highest Soft Constraint Optimization (HSCO)

4 Experimental Environment

The algorithm was implemented and tested on a PC with an Intel Xeon E5-1603 2.8GHz processor, 8GB RAM and Windows 7. The program was coded in C# targeting the .NET Framework 4.5. For each problem set, the program was executed for ten iterations, with a 240 second time limit per iteration determined by a benchmarking application released by the competition organizers. During initial experimentation it was found that allowing adaptive construction to execute for approximately 10% of the total execution time provided the best results with the new code.

5 Results and Analysis

The random seed used for generation of the results below has been recorded to ensure repeatability of the experiments. Initial experimentation identified that splitting the weighted list into sixths provided a greater improvement than any larger split, as well as good performance overall for the new optimization phases.

As with the EGD, the chosen neighbourhood search heuristics have been kept deliberately simple. While more complex heuristics can identify the optimal move, under previous experimentation these were found to have the effect of directing the search too intensively, resulting in more frequent local optimum situations [7]. The use of relatively simple search heuristics ensures that

the process is not protracted by time consuming explorations of the search space. For reference, SD is the Standard Deviation.

	Exam 1		Exam 2		Exam 3		Exam 4	
	EGD	**DS**	EGD	**DS**	EGD	**DS**	**EGD**	DS
Worst	5865	5405	495	430	10909	10813	24405	22464
Best	5377	5186	435	405	10236	9399	19171	19031
Average	5598.5	5302.1	454.6	418.1	10444.4	10036.6	20241	20531.3
SD	198.9630	71.5766	17.9580	8.9747	201.5403	496.6016	1516.1996	1241.7174

Table 2a Results for Exams 1 to 4 using ITC 2007 time limit

	Exam 5		Exam 6		Exam 7		Exam 8	
	EGD	DS	**EGD**	DS	EGD	**DS**	EGD	**DS**
Worst	3349	3337	26465	26575	4688	4219	8669	7704
Best	3090	3117	25940	26055	4475	3997	8050	7303
Average	3199.7	3236.8	26240	26253.5	4567.8	4115.7	8353.5	7555.1
SD	75.1059	75.6333	157.9557	166.9340	60.1956	69.6867	177.7734	135.4035

Table 2b Results for Exams 5 to 8 using ITC 2007 time limit

	Exam 9		Exam 10		Exam 11		Exam 12	
	EGD	**DS**	EGD	**DS**	EGD	**DS**	EGD	**DS**
Worst	1185	1124	15805	15940	132483	34773	5871	5564
Best	1049	1048	14636	14789	31080	30311	5311	5369
Average	1113.6	1089.8	15332.2	15167.9	68217.8	31415.1	5596.9	5464.4
SD	41.6445	24.0361	344.1898	413.5965	39829.9124	1339.0058	151.0875	63.9083

Table 2c Results for Exams 9 to 12 using ITC 2007 time limit

Tables 2a, 2b and 2c compare the new Directed Selection technique against the Extended Great Deluge algorithm. The new technique is able to produce better results than the EGD algorithm in nine of the twelve datasets. For Exams 1, 2, 8, 9, 11 and 12 the new Directed Selection (DS) technique is also able to produce more consistent results, as measured by the standard deviation, than those generated with the EGD algorithm.

Exams 4, 5 and 6 are the only instances where the original EGD algorithm outperforms DS. These specific instances have the lowest number of room and time combinations, and as such there is relatively small freedom of movement during the optimization process. This affects both the DSO and HSCO phases due to the nature of the neighbourhood heuristics involved. Due to the smaller freedom of movement, and the more focused examination of the search space, fewer improvements are identified. EGD is not affected as much due to its

exclusive use of stochastic selection, resulting in a greater examination of the whole search space.

	Directed Selection		Other Techniques			
	Best	Average	Müller ITC 2007 [30]	Adaptive Linear Combination [31]	Graph Colouring [9]	Multistage Algorithmic [32]
Exam 1	5186	5302.1	**4370**	5231	6234	5814
Exam 2	405	418.1	400	433	**395**	1062
Exam 3	9399	10036.6	10049	**9265**	13002	14179
Exam 4	19031	20531.3	18141	**17787**	17940	20207
Exam 5	3117	3236.8	**2988**	3083	3900	3986
Exam 6	**26055**	26253.5	26585	26060	27000	27755
Exam 7	**3997**	4115.7	4213	10712	6214	6885
Exam 8	**7303**	7555.1	7742	12713	8552	10449
Exam 9	1048	1089.8	**1030**	1111	N/A	N/A
Exam 10	**14789**	15167.9	16682	14825	N/A	N/A
Exam 11	30311	31415.1	34129	**28891**	N/A	N/A
Exam 12	**5369**	5464.4	5535	6181	N/A	N/A

Table 3 Comparison of best results and other techniques that keep competition time limits

Table 3 compares the new Directed Selection technique against those of Müller and other recently published research that utilizes the competition rules for time limits. Due to the prior unavailability of the hidden competition datasets, comparison results are not widely available for research that has been published post competition. While Müller's four phased technique continues to show its strength by producing the lowest penalties of all the approaches listed in four of the twelve datasets, Directed Selection is able to produce lower penalties for five of the twelve datasets. When compared to other post competition techniques, Directed Selection is able to produce significantly lower penalties for six of the eight public datasets.

	Directed Selection		Other Techniques			
	Best	Average	Extended Great Deluge [23]	Grammatical Evolution Hyper-heuristic [33]	Pursuit of Better Results Using Grid Resources [34]	Distributed Scatter Search [25]
Exam 1	5186	5302.1	4633	4362	4699	**4128**
Exam 2	405	418.1	405	**380**	385	**380**
Exam 3	9399	10036.6	9064	8991	8500	**7769**
Exam 4	19031	20531.3	15663	15094	14879	**13103**
Exam 5	3117	3236.8	3042	2912	2795	**2513**
Exam 6	26055	26253.5	25880	25735	25410	**25330**
Exam 7	3997	4115.7	4037	4025	3884	**3537**
Exam 8	7303	7555.1	7461	7452	7440	**7087**
Exam 9	1048	1089.8	1071	N/A	N/A	**913**
Exam 10	14789	15167.9	14374	N/A	N/A	**13053**
Exam 11	30311	31415.1	29180	N/A	N/A	**24369**
Exam 12	5369	5464.4	5693	N/A	N/A	**5095**

Table 4 Comparison of best results and other techniques that do not use ITC 2007 time limits

Table 4 compares the new Directed Selection technique against those of recently published research that do not utilize the competition rules for time limits. While the technique understandably is out preformed across all of the

data sets due to the significantly shorter time limit used, the results produced remain competitive. It is worth noting that the difference between the best recorded penalties and the technique presented here is small when considering the difference in execution time; 4 hours for Distributed Scatter Search and 240 seconds for Directed Selection.

6 Conclusion

This paper presents a new optimization technique which can successfully utilize information gathered during adaptive construction to direct and influence the selection of examinations used in the neighbourhood search process, augmenting the traditional stochastic selection method, as well as a new optimization heuristic inspired by the Highest Cost construction heuristic. These techniques have successfully been used for solving the Examination Timetabling Problem as described in the second International Timetabling Competition, ITC 2007. The combined technique presented has not been tailored specifically for solving this problem, and could be adapted for solving other problem areas. In addition to testing against other benchmark datasets, the effectiveness of the technique to solve other timetabling and scheduling problems will be investigated in future work.

Traditional scheduling techniques have primarily been implemented as single threaded applications. Over the past five years, there has been a significant increase in the availability of multi-core processors, and it is now common for modern desktop and laptop computers to contain processors which have multiple physical cores and are highly optimized for parallel processing. Due to this hardware shift, traditional schedulers no longer take full advantage of the underlying hardware. Future work will look into exploiting parallelism inherent in modern computing. While the Directed Selection technique presented here does not extend itself easily to traditional parallelism, this capability of modern processors can be exploited in other ways. Work is currently involved in identifying how multiple threads or processes working independently on a single problem, while also sharing useful information about the nature of the underlying problem, can be exploited to further improve upon the optimization process.

References

1. B. McCollum, "A perspective on bridging the gap between theory and practice in university timetabling," *Practice and Theory of Automated Timetabling VI*, vol. 3867, pp. 3–23, 2007.
2. R. Qu, E. K. Burke, B. McCollum, L. T. G. Merlot, and S. Y. Lee, "A survey of search methodologies and automated system development for examination timetabling," *Journal of Scheduling*, vol. 12, no. 1, pp. 55–89, Oct. 2008.
3. S. Kristiansen and T. R. Stidsen, "A Comprehensive Study of Educational Timetabling - a Survey," *Department of Management Engineering, Technical University of Denmark*, no. November, 2013.

4. B. McCollum, A. Schaerf, B. Paechter, P. McMullan, R. Lewis, A. J. Parkes, L. D. Gaspero, R. Qu, and E. K. Burke, "Setting the Research Agenda in Automated Timetabling: The Second International Timetabling Competition," *INFORMS Journal on Computing*, vol. 22, no. 1, pp. 120–130, May 2009.
5. M. W. Carter, G. Laporte, and S. Y. Lee, "Examination Timetabling : Algorithmic Strategies and Applications," *The Journal of the Operational Research Society*, vol. 47, no. 3, pp. 373–383, 1996.
6. B. McCollum, P. Mcmullan, T. Müller, and A. J. Parkes, "Next Steps for the Examination Timetabling Format and Competition," *Proceedings of PATAT 2012*, pp. 418–420, 2012.
7. E. K. Burke, G. Kendall, B. McCollum, and P. Mcmullan, "Constructive versus improvement heuristics: an investigation of examination timetabling," *3rd Multidisciplinary International Scheduling Conference: Theory and Applications*, pp. 28–31, 2007.
8. E. Burke and J. Newall, "Solving Examination Timetabling Problems through Adaption of Heuristic Orderings," *Annals of Operations Research*, vol. 129, no. 1-4, pp. 107–134, Jul. 2004.
9. N. R. Sabar, M. Ayob, R. Qu, and G. Kendall, "A graph coloring constructive hyper-heuristic for examination timetabling problems," *Applied Intelligence*, vol. 37, no. 1, pp. 1–11, Aug. 2011.
10. M. Dorigo, G. D. Caro, and L. M. Gambardella, "Ant Algorithms for Discrete Optimization," *Artificial Life*, vol. 5, no. 2, pp. 137–172, Apr. 1999.
11. J. H. Obit, D. Ouelhadj, D. Landa-Silva, and R. Alfred, "An Evolutionary Non-Linear Great Deluge Approach for Solving Course Timetabling Problems," *IJCSI International Journal of Computer Science Issues*, vol. 9, no. 4, pp. 1–13, 2012.
12. T. B. Cooper and J. H. Kingston, "The Complexity of Timetable Construction Problems," *Lecture Notes in Computer Science*, vol. 1153, pp. 281–295, 1996.
13. N. Pillay and W. Banzhaf, "An informed genetic algorithm for the examination timetabling problem," *Applied Soft Computing*, vol. 10, no. 2, pp. 457–467, Mar. 2010.
14. E. K. Burke and Y. Bykov, "A late acceptance strategy in hill-climbing for exam timetabling problems," *PATAT 2008 Conference, Montreal, Canada*, 2008.
15. E. Özcan, M. Misir, G. Ochoa, and E. K. Burke, "A Reinforcement Learning - Great-Deluge Hyper-Heuristic for Examination Timetabling," *International Journal of Applied Metaheuristic Computing*, vol. 1, no. 1, pp. 39–59, Jan. 2010.
16. R. Bai, J. Blazewicz, E. K. Burke, G. Kendall, and B. McCollum, "A simulated annealing hyper-heuristic methodology for flexible decision support," *4OR*, vol. 10, no. 1, pp. 43–66, Nov. 2011.
17. E. Ikonomovska, I. Chorbev, D. Gjorgjevik, and D. Mihajlov, "The Adaptive Tabu Search and Its Application to the Quadratic Assignment Problem," in *9th International Multiconference Information Society 2006*, M. Bohanec, M. Gams, V. Rajkovič, T. Urbančič, M. Bernik, D. Mladenić, M. Grobelnik, M. Heričko, U. Kordeš, O. Markič, J. Musek, M. Osredkar, I. Kononenko, and B. Novak Škarja, Eds., 2006, pp. 26–29.
18. S. Abdullah and H. Turabieh, "On the use of multi neighbourhood structures within a Tabu-based memetic approach to university timetabling problems," *Information Sciences*, vol. 191, pp. 146–168, May 2012.
19. J. S. Appleby, D. V. Blake, and E. A. Newman, "Techniques for producing school timetables on a computer and their application to other scheduling problems," *The Computer Journal*, 1961.
20. S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing." *Science (New York, N.Y.)*, vol. 220, no. 4598, pp. 671–80, May 1983.
21. G. Dueck and T. Scheuer, "Threshold accepting: A general purpose optimization algorithm appearing superior to simulated annealing," *Journal of Computational Physics*, vol. 90, no. 1, pp. 161–175, Sep. 1990.
22. P. Mcmullan, "An extended implementation of the great deluge algorithm for course timetabling," *Computational ScienceICCS 2007*, vol. 4487, pp. 538–545, 2007.
23. B. McCollum, P. McMullan, A. J. Parkes, E. K. Burke, and S. Abdullah, "An extended great deluge approach to the examination timetabling problem," *Proceedings of the 4th Multidisciplinary International Conference on Scheduling: Theory and Applications*, no. August, pp. 424–434, 2009.

24. M. Eley, "Ant algorithms for the exam timetabling problem," *Practice and Theory of Automated Timetabling VI*, vol. 3867, pp. 167–180, 2006.
25. C. Gogos, G. Goulas, P. Alefragis, V. Kolonias, and E. Housos, "Distributed scatter search for the examination timetabling problem," in *Proceedings of PATAT 2010*, 2010, pp. 211–223.
26. R. Hamilton-Bryce, P. McMullan, and B. McCollum, "Directing selection within an extended great deluge optimization algorithm," *Multidisciplinary International Conference on Scheduling : Theory and Applications (MISTA 2013)*, 2013.
27. D. P. Clements and D. E. Joslin, "Squeaky Wheel Optimization," *Journal Of Artificial Intelligence Research*, vol. 10, pp. 353–373, May 1999.
28. E. Ozcan, Y. Bykov, M. Birben, and E. K. Burke, "Examination timetabling using late acceptance hyper-heuristics," in *2009 IEEE Congress on Evolutionary Computation*. IEEE, May 2009, pp. 997–1004.
29. 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, no. 2, pp. 482–491, Sep. 2009.
30. T. Müller, "ITC2007 solver description: a hybrid approach," *Annals of Operations Research*, vol. 172, no. 1, pp. 429–446, Oct. 2009.
31. S. Abdul Rahman, A. Bargiela, E. K. Burke, E. Özcan, B. McCollum, and P. McMullan, "Adaptive linear combination of heuristic orderings in constructing examination timetables," *European Journal of Operational Research*, vol. 232, no. 2, pp. 287–297, Jan. 2014.
32. C. Gogos, P. Alefragis, and E. Housos, "An improved multi-staged algorithmic process for the solution of the examination timetabling problem," *Annals of Operations Research*, vol. 194, no. 1, pp. 203–221, Feb. 2010.
33. N. R. Sabar, M. Ayob, G. Kendall, and R. Qu, "Grammatical Evolution Hyper-heuristic for Combinatorial Optimization problems," *IEEE Transactions on Evolutionary Computation*, no. October, pp. 1–22, 2013.
34. C. Gogos, G. Goulas, P. Alefragis, and E. Housos, "Pursuit of better results for the examination timetabling problem using grid resources," in *2009 IEEE Symposium on Computational Intelligence in Scheduling*. IEEE, Mar. 2009, pp. 48–53.