A Hidden Markov Model Approach to the Problem of Heuristic Selection in Hyper-Heuristics with a Case Study in High School Timetabling Problems

Ahmed Kheiri* a.kheiri@exeter.ac.uk
University of Exeter, College of Engineering, Mathematics and Physical Sciences, Streatham Campus, Harrison Building, Exeter EX4 4QF, United Kingdom

Ed Keedwell e.c.keedwell@exeter.ac.uk
University of Exeter, College of Engineering, Mathematics and Physical Sciences, Streatham Campus, Harrison Building, Exeter EX4 4QF, United Kingdom

doi:10.1162/EVCO_a_00186

Abstract

Operations research is a well-established field that uses computational systems to support decisions in business and public life. Good solutions to operations research problems can make a large difference to the efficient running of businesses and organisations and so the field often searches for new methods to improve these solutions. The high school timetabling problem is an example of an operations research problem and is a challenging task which requires assigning events and resources to time slots subject to a set of constraints. In this article, a new sequence-based selection hyper-heuristic is presented that produces excellent results on a suite of high school timetabling problems. In this study, we present an easy-to-implement, easy-to-maintain, and effective sequence-based selection hyper-heuristic to solve high school timetabling problems using a benchmark of unified real-world instances collected from different countries. We show that with sequence-based methods, it is possible to discover new best known solutions for a number of the problems in the timetabling domain. Through this investigation, the usefulness of sequence-based selection hyper-heuristics has been demonstrated and the capability of these methods has been shown to exceed the state of the art.

Keywords

Hyper-heuristic, educational timetabling, computational design, combinatorial optimisation, hidden Markov model.

1 Introduction

The field of search and optimisation in operations research has a long and varied history. Many methods have been developed that solve specific problems well and rely on problem-specific knowledge to function. Although highly computationally efficient, many of these methods are necessarily specific to each problem and so cannot easily be applied to other problems without significant modification. The research community has therefore looked to metaheuristics as generic problem-solving methods, for

*Ahmed Kheiri (KheiriA@cardiff.ac.uk) is currently working at Cardiff University.

Manuscript received: October 30, 2015; revised: March 15, 2016 and May 17, 2016; accepted: May 27, 2016.

example, genetic algorithms (Holland, 1975), tabu search methods (Glover, 1986), and other nature-inspired methods such as simulated annealing (Kirkpatrick et al., 1983) are widely used. However, these methods usually require significant expertise to implement and tune for specific problems, and in their standard versions at least, are unable to adapt to changing search spaces. Despite the efforts of the scientific community in developing a variety of search methodologies, there is no known efficient method that offers the best performance for solving different problems and for all possible situations, and in fact it has been proved that this is unachievable (Wolpert and Macready, 1997). Despite this, there has been scientific progress in designing general methods which are valid for a wide range of, rather than all, problem domains. Such systems are applicable to different instances from not only the same domain but also other problem domains. Moreover, they are easy to build and maintain and so are less costly in terms of the investment in time required to apply them to new problems. Hyper-heuristics have emerged as such general purpose, high-level search methodologies which are motivated by the goal of *selecting* or *generating* heuristics automatically to solve a wide range of difficult optimisation problems (Burke et al., 2013). This work focuses on the selection type of hyper-heuristics.

Traditional selection hyper-heuristics have focused on the improvement that single or paired low-level heuristics can bring to an optimisation. The task for these methods is to select the most appropriate and best performing heuristic for a given point in the optimisation. A wide number of methods have been proposed for this task including the simple random hyper-heuristic, choice function hyper-heuristic, and other methods presented in Burke et al. (2013). However, in scientific fields where sequences are known to be prevalent, for example in bioinformatics and language processing, a key feature is that context is found to be important and that the "meaning" of a token in the sequence cannot be determined unless the context in which it finds itself is also considered. Here it is proposed that it is possible to extend this context principle to the selection of heuristics within the search domain, noting that the effectiveness of search operations is determined to a certain extent by those that have been executed before it. In this article, we extend this principle to investigate the potential for the analysis of sequences of operations in search and optimisation problems to construct building blocks of good heuristic combinations.

This work studies and analyses the performance of two sequence-based methods, a simple fixed parametrised method and a hidden Markov model (HMM) approach (Kheiri and Keedwell, 2015b; Kheiri et al., 2015). A hyper-heuristic with a fixed parametrised sequence size is implemented to allow for experimentation on the sequence lengths and to discover information regarding the information that can be gained from each such sequence. In addition, the HMM method is shown to learn online the optimum sequence lengths automatically and is able to adapt the probability of heuristic application and sequence-based acceptance strategy to the search landscape. Experimentation with these methods is conducted on difficult problems from timetabling, a key problem area in operational research.

The high school timetabling problem is a hard combinatorial optimisation problem (Even et al., 1976). A solution requires the scheduling of events and resources in time slots subject to a set of *hard* and *soft* constraints. A solution is expected to satisfy all the hard constraints and as many of the soft constraints as possible. The importance of the high school timetabling problem stems from its difficulty due to the number of constraints involved, the NP nature of the problem (Even et al., 1976), and the need to perform this hard task by educational institutions everywhere; thus it became necessary to search for methods that help with automating this process.

The article is structured as follows. Section 2 overviews selection hyper-heuristics and high school timetabling problems. Section 3 describes the high school timetabling benchmark used for the experimentation. Section 4 describes the developed method including the algorithmic components and low-level heuristics. In Section 5, the performance of the developed methods is analysed and compared against the state-of-the-art approaches. Section 6 provides conclusions and areas for further work.

2 Background

2.1 Hyper-Heuristics

The term "hyper-heuristic" was used by Cowling et al. (2001) to describe a high-level problem-independent method that provides a solution methodology to solve a wide range of optimisation algorithms. Hyper-heuristics can be broadly classified into *selection* hyper-heuristics to select from a set of predefined heuristics, and *generation* hyper-heuristics which differ from selection methods in that they generate heuristics (Burke et al., 2013). This work focuses on the former class of hyper-heuristics.

Selection hyper-heuristics are motivated by the understanding that each heuristic performs differently on different problem instances and an approach that combines them could yield better overall performance (Burke et al., 2013). An iterated selection hyper-heuristic method (Özcan et al., 2008) aims to improve the current solution by *selecting* and applying a low-level heuristic from a set of predefined low-level heuristics (e.g., mutation operators and local search heuristics), leading to a new solution; then the *move acceptance* method decides whether to accept or reject the modified solution. This cycle of applying selection and move acceptance methods is repeated until a termination criterion is satisfied. The number of studies on selection hyper-heuristics is growing massively (Burke et al., 2013) and as such, space prevents us reviewing all such approaches. However, we will overview the methods that are relevant to those techniques proposed in this study.

The heuristic selection method is described in Section 4 but this must be coupled with a move acceptance method to determine whether to select or reject the generated solutions. A set of well-known metaheuristic-inspired move acceptance methods, including hill climbing (only improving) (HC), simulated annealing (SA), great deluge (GD), record-to-record travel (RR), and late acceptance (LA) methods are used as move acceptance criteria in this study.

A deterministic move acceptance method which accepts only improved solutions is described by Cowling et al. (2001). We refer to this method as the hill climbing move acceptance method.

Simulated Annealing (SA) (Abramson et al., 1999) is a probabilistic metaheuristic method, motivated by an analogy to the process of annealing in solids. At each step a new solution is generated. The new solution is accepted if it improved the previous solution. To prevent premature convergence on a local optimum, nonimproving solutions are accepted with a probability of $p = e^{-\frac{\Delta}{T}}$, where Δ is the quality (cost) change, and T is the method parameter, known as temperature, which regulates the probability to accept solutions with higher cost. Generally speaking, the method starts with a high temperature, then according to the cooling schedule, the temperature decreases gradually towards the end of the search process. One way of reducing the temperature is to apply the geometric cooling schedule: $T_{i+1} = T_i.\beta$, where β can be empirically tuned for a particular problem domain (Hajek, 1988). Simulated annealing has been used as a move acceptance method within the selection hyper-heuristics in Bilgin et al.

(2007), at which the nonimproving solutions are accepted with a probability given by the following equation:

$$p = e^{-\frac{\Delta}{F(1-\frac{t_{current}}{t_{limit}})}}, \tag{1}$$

where Δ is the change in the cost at time $t_{current}$, t_{limit} is the time limit, and F is the expected maximum change in the cost. Note that Δ is positive for nonimproving moves when the objective is minimising rather than maximising; and F is empirically determined in this work.

The Great Deluge (GD) algorithm was first introduced by Dueck (1993). GD is based on a stochastic framework which allows improving moves by default. Nonimproving moves are accepted if the cost of the candidate solution is better than an expected cost, named as the water level at each step. The water level gets updated according to the "rain speed" parameter. Dueck (1993) argued that if the rain speed value is chosen to be very small, then the algorithm requires high computational time to produce a high-quality solution. The cost value of the first generated candidate solution can be used as the initial level in GD. Great deluge is utilised as a move acceptance method within selection hyper-heuristics in Kendall and Mohamad (2004), at which the threshold level (τ) at time $t_{current}$ is updated with the following equation:

$$\tau = f + F \times \left(1 - \frac{t_{current}}{t_{limit}}\right), \tag{2}$$

where t_{limit} is the time limit, F is the expected maximum change in the cost, and f is the expected final cost value. In this work F and f are empirically determined.

Dueck (1993) proposed another variant of GD named the record-to-record travel (RR) method. The idea of RR is based on the simple notion that any new solution, which is not much worse than the best solution recorded, is accepted.

The late acceptance method (Özcan et al., 2009) is a variant of the hill climbing method. A candidate solution in the late acceptance method is accepted if its quality is better than a solution which was obtained L steps before. The method requires an implementation of a circular queue of size L which maintains the cost values of L previously visited solutions.

2.2 High School Timetabling Problems

Due to the extreme difficulty of high school timetabling problems, metaheuristics are preferred in most of the studies. Abramson (1991) described the high school timetabling problem and then proposed a simulated annealing solution method. The author presented a parallel algorithm and proved experimentally that the developed method works faster than the equivalent sequential algorithm. The tests were made on data from an Australian school. The THOR school timetabling tool was developed by Melício et al. (2006) for Portuguese schools. An initial solution is first created using a heuristic construction algorithm, which is improved further by applying fast simulated annealing. The tool was used by more than 100 Portuguese schools and was applied with great success. Zhang et al. (2010) used a simulated annealing–based algorithm with a newly designed neighbourhood structure to solve the high school timetabling problem.

One of the earliest studies that applied the tabu search to solve the high school timetabling problem was suggested by Wright (1996). The conducted study produced a timetable for a large comprehensive school in England, using a solution method that involved heuristic search and a form of tabu search. Alvarez-Valdés et al. (2002)

developed an algorithm based on tabu search for teacher assignment in Spanish secondary schools. The algorithm was implemented in three phases. In the first phase a parallel heuristic algorithm is used to build an initial solution, and in the second phase the solution is improved using the tabu search to obtain feasible solutions. In the third phase the solution is improved further. The algorithm was applied to 12 Spanish school instances and proved to provide better allocations than the manual ones. Jacobsen et al. (2006) developed an approach that generates an initial solution using a construction heuristic with a graph colouring algorithm. The solution is then improved using tabu search. The algorithm was tested on data from German high schools. Bello et al. (2008) treated the high school timetabling problem as a graph colouring problem, and used it in association with the tabu search algorithm. The system was applied to instances from Brazilian high schools.

Colorni et al. (1992) used several metaheuristics based on genetic algorithms (GA), simulated annealing (SA), and tabu search (TS) and compared them using instances of Italian high school data. The research results showed that a hybrid of a genetic algorithm with local search could give promising performance. Calderia and Ross (1997) evaluated the use of genetic algorithms to solve instances of school timetabling that are randomly generated. An initial population of feasible timetables are produced in an initial procedure and the GA is used to improve the quality of the generated population. A highly constrained school timetabling problem extracted from the requirements of a German high school was investigated by Bufé et al. (2001) using a hybrid approach. An evolutionary algorithm combined with local search that uses specific mutation operators to optimise the given timetables was used to find feasible solutions. Filho et al. (2001) used a new representation for the high school timetabling problem by forming clusters from pairs of teachers and classes. The authors applied a constructive genetic algorithm to solve instances of two Brazilian high schools. Wilke et al. (2002) presented a genetic algorithm for solving the German high school timetabling problem. A hybrid genetic approach is applied using multiple genetic operators which proved to perform better than the traditional genetic algorithm. Beligiannis et al. (2008) solved the high school timetabling problem using an adaptive evolutionary algorithm. The algorithm did not employ a crossover operator, and the results showed the success of the approach when applied on the Greek high school timetabling problem. Raghavjee and Pillay (2008) applied a genetic algorithm to the school timetabling problem. The algorithm is based on creating an initial population of timetables and then applying a mutation operator to refine this population. It was tested with five high school timetabling problems and proved to generate better results than all the methods tested with the same set. Raghavjee and Pillay (2012) compared the performance of a genetic algorithm and genetic programming using the Abramson (1991) dataset of five high school timetabling problems. The genetic programming approach proved to give a better performance in the five problems over other approaches including genetic algorithm, neural networks, tabu search, and greedy search, especially in the large problems in the set.

Other approaches used in the high school timetabling problem include adaptive large neighbourhood search (Sørensen et al., 2012), integer programming (Birbas et al., 2009), particle swarm optimisation (Tassopoulos and Beligiannis, 2012), tiling algorithms (Kingston, 2005), bee algorithms (Lara et al., 2008), Hopfield neural networks (Smith et al., 2003), walk down jump up algorithm (Wilke and Killer, 2010), greedy randomised adaptive search procedure (Moura and Scaraficci, 2010), and constraint programming approach (Valouxis and Housos, 2003). For the survey on the high school timetabling problem, the reader is directed to Pillay (2013).

3 High School Timetabling Benchmark

Due to the different education systems and constraints imposed by different educational institutions, a group of researchers (Post et al., 2012) has proposed a unified high school timetabling format that fits with the different education systems across the world. To encourage scientists and practitioners to provide solution methods for the high school timetabling problem, an international timetabling competition (ITC 2011)[1] (Post et al., 2013) using the unified benchmark instances collected from over ten countries was organised.

The competition consisted of three rounds. In the first and third rounds, competitors were expected to submit solutions to a set of instances without any restriction on the resources or techniques used to generate these solutions. In the second round, solvers were submitted to the organisers and tested on a set of instances in a specified time limit of 1000 seconds.

The problem instance consists of *times* which are the intervals of time in which events run; *resources* which are the entities that attend events; and *events* which specify the coordination of resources. Solutions to the timetabling problem consist of the allocation of resources to events. The resources that must be allocated are the group of students (known as a class), the teacher, and the room in which the event will take place. Each resource has a set of constraints associated with it (e.g., limitations on the number of classes students are required to take in one day, teacher workload, and room capacities). Events can be single lessons or a set of lessons (an event group) and each event has a number of properties that influence the allocation of resources to them through constraints or other optimality criteria. These are the event duration, pre-assigned resources (e.g., some events must take place in certain rooms), the contribution of the event to workload, and pre-assigned time slots. Fifteen hard/soft *constraints* are expected to be satisfied (Post et al., 2013):

- **C01 Assign resource:** Assign resource to event.
- **C02 Assign time:** Assign time to event.
- **C03 Split events:** Split event into subevents under specific constraints.
- **C04 Distribute split events:** Split event into subevents under constrained durations.
- **C05 Prefer resources:** Assign specific resource(s) to event.
- **C06 Prefer times:** Assign specific time(s) to event.
- **C07 Avoid split assignments:** Assign the same resource to a set of events.
- **C08 Spread events:** Spread events evenly through the cycle.
- **C09 Link events:** Assign the same time to a set of events.
- **C10 Avoid clashes:** Assign resources without having clashes.
- **C11 Avoid unavailable times:** Avoid assigning resources at unavailable times.

[1] Available at http://www.utwente.nl/ctit/hstt/itc2011/

- **C12 Limit idle times:** Avoid having idle times for resources.
- **C13 Cluster busy times:** For a number of days, resources must be busy.
- **C14 Limit busy times:** For a number of times, resources must be busy.
- **C15 Limit workload:** Schedule the workload without exceeding a limit.

The quality of a solution is evaluated in terms of (*hardViolationScore*,*softViolation Score*). A solution with a cost of (25,78) indicates an infeasibility value of 25 (sum of weighted hard constraints violations) and objective value of 78 (sum of weighted soft constraints violations). The weight value and whether the constraint is hard or soft per each constraint type are presented in the instance.

The instances used during ITC 2011 are still available online in the competition website, but deprecated. The developers of the benchmark project suggested the focus on solving a set of instances, referred to as XHSTT-2014 instances, which contains a carefully selected set of worldwide instances in their most up-to-date form. However, in this work, we used the ITC 2011 instances of round 2 and compared the performance of our approach against the ITC 2011 solvers using the same rules imposed for the second round of the competition. We then applied the developed method on XHSTT-2014 instances. Table 1 summarises the main characteristics of the ITC 2011 and XHSTT-2014 instances obtained from 12 countries.

Due to space restrictions, we provide only a brief description of the studied problem. For a full description of the problem, the reader is directed to Post et al. (2012, 2013).

Four competitors (GOAL, HySST, Lectio, and HFT) submitted solvers to the second round of the ITC 2011 competition (Post et al., 2013; Kheiri, 2014). GOAL combined iterated local search with simulated annealing (Fonseca et al., 2014). HySST applied a stochastic local search hyper-heuristic (Kheiri et al., 2016). Lectio employed an approach based on adaptive large neighbourhood search (Sørensen et al., 2012). HFT developed an evolutionary algorithm as a solution method (Domrös and Homberger, 2012). The ranking method was based on the average rank of ten independent trials with ten seeds chosen at random over eighteen selected instances per team with each run for 1,000 seconds. The GOAL team obtained the lowest mean rank and were therefore deemed the winner of the second round of ITC 2011. Soon after the competition, the results of the GOAL team were improved using a late acceptance hill-climbing method as reported in Fonseca et al. (2015).

4 Overall Approach

Hyper-heuristic methods operate above the level of heuristics and so do not deal directly with the problem representation; the low-level heuristics provided are designed to work at this level. As such, the proposed hyper-heuristic algorithm works simply by invoking the KHE platform, an open source software tool written by Jeff Kingston (2014)[2] which constructs initial solutions using the concept of hierarchical timetabling utilising a tree structure to represent resource allocations to events. The exact implementation of the solution is not of concern from a hyper-heuristic perspective, but the interested reader is directed to Kingston (2014). However, an initial produced solution of a given problem instance generally violates many of the problem constraints (see Kingston, 2014). The

[2]Available at http://sydney.edu.au/engineering/it/~jeff/khe/

Table 1: Characteristics of ITC 2011 and XHSTT-2014 problem instances.

Country-Instance	Times	Teachers	Rooms	Classes	Students	Events	Duration
Round 2 of ITC 2011 set							
Brazil-Instance3	25	16		8		69	200
Finland-ElementarySchool	35	22	21	60		291	445
Finland-SecondarySchool2	40	22	21	36		469	566
Greece-Aigio1stHighSchool2010	35	37		208		283	532
Netherlands-Kottenpark2008	40	81	11	34		1047	1118
Greece-WesternUniversityInstance3	35	19		6		210	210
Greece-WesternUniversityInstance5	35	18		6		184	184
Common to both sets							
Brazil-Instance2	25	14		6		63	150
Brazil-Instance4	25	23		12		127	300
Brazil-Instance6	25	30		14		140	350
Spain-School	35	66	4	21		225	439
Greece-WesternUniversityInstance4	35	19		12		262	262
Italy-Instance4	36	61		38		748	1101
Kosova-Instance1	62	101		63		809	1912
Netherlands-Kottenpark2003	38	75	41	18	453	1156	1203
Netherlands-Kottenpark2005	37	78	42	26	498	1235	1272
Netherlands-Kottenpark2009	38	93	53	48		1148	1274
South Africa-Woodlands2009	42	40		30		278	1353
XHSTT-2014 set							
Australia-BGHS98	40	56	45	30		387	1564
Australia-SAHS96	60	43	36	20		296	1876
Australia-TES99	30	37	26	13		308	806
Denmark-Falkonergaardens Gymnasium2012	50	90	69		279	1077	1077
Denmark-HasserisGymnasium2012	50	100	71		523	1235	1235
Denmark-VejenGymnasium2009	60	46	53		163	918	918
Finland-College	40	46	34	31		387	854
Finland-HighSchool	35	18	13	10		172	297
Finland-SecondarySchool	35	25	25	14		280	306
Greece-HighSchool1	35	29		66		372	372
Greece-ThirdHighSchoolPatras2010	35	29		84		178	340
England-StPaul	25	68	67	67		1227	1227
USA-Westside2009	100	134	108			628	6354
South Africa-Lewitt2009	148	19	2	16		185	838

proposed hyper-heuristic is, therefore, used to fix as many of these violated constraints as possible.

Recall that hyper-heuristics are composed of two components: the selection component and the move acceptance component. In the selection component, a low-level heuristic will be selected and applied to a current solution ($S_{current}$) and that will

generate a new solution (S_{new}). Now, given $S_{current}$ and S_{new}, the move acceptance will decide whether to accept or reject S_{new}. In this work, a sequence-based selection (SS) method described in Section 4.1 is used to select and apply sequences of heuristics. Five different reusable metaheuristics inspired methods are used as move acceptance including hill climbing local search (HC), simulated annealing (SA), great deluge (GD), record-to-record travel (RR), and late acceptance (LA) methods. For example, if we used HC as a move acceptance, then S_{new} will be accepted only if its quality is better than $S_{current}$; otherwise, it will be rejected. If we used SA, then S_{new} will be accepted if its quality is better than $S_{current}$ or it will be accepted with a given probability.

Most of the parameters involved with the move acceptance methods are set to values which were suggested in previous studies. In the case of solutions with hard constraint violations, the F value in both simulated annealing and great deluge criteria is assigned to 0.01% of the cost of the best recorded solution during the search process; otherwise, the value is set to 1% of the cost of the best recorded solution in hand. Similarly, the value of f in GD is set to 0.001% of the cost of the best solution in hand, and to 0.1% if the best solution violates only soft constraints. These settings are suggested in Kalender et al. (2013) and Ahmed et al. (2015). The memory size L of the LA acceptance method is set to 500 as suggested in Özcan et al. (2009). The RR accepts the nonimproving solution if its cost is not worse than the cost of the best recorded solution in hand plus 0,5. This parameter is chosen after light experimentation.

The sequence-based selection hyper-heuristic approach in this study manages a set of 15 low-level heuristics to improve the quality of a single solution:

- **LLH0:** swap the time slots of two randomly selected events. As an example, assume that a *Geography* class event is assigned to the second time slot on Monday and the *History* class event is assigned to the third time slot on Friday. LLH0 will assign *History* class to the second time slot on Monday, and *Geography* class to the third time slot on Friday.

- **LLH1:** select two random events and swap their time slots in case they have the same duration or not adjacent; otherwise, the swap occurs but the first event is moved to follow the last time slot occupied by the second event. As an example, assume that a *Geography* class event with a duration of one is assigned to the first time slot on Monday and the *History* class event with a duration of two is assigned to the second time slot on Monday. LLH1 will assign *Geography* class to the third time slot on Monday (not to the second time slot on Monday), and *History* class to the first time slot on Monday. However, if both events have the same duration, then LLH0 will be invoked.

- **LLH2:** randomly select an event and reschedule to a random time slot. As an example, assume that a *Mathematics* class event is assigned to the first time slot on Monday. LLH2 will select a new random time slot, for example, the last time slot on Tuesday, and then it will unassign *Mathematics* from the first time slot on Monday and then assign it to the last time slot on Tuesday.

- **LLH3:** randomly select an unassigned event and then assign it to a random time slot. This low-level heuristic works similarly to LLH2; however, the event is expected to be unassigned to begin with. As an example, assume that a *Mathematics* class event is unassigned to any time slot. LLH3 will select a

random time slot, for example, last time slot on Tuesday, and then it will assign *Mathematics* to the last time slot on Tuesday.

- **LLH4:** randomly select an assigned event and then unassign it. This low-level heuristic is the opposite of LLH3.
- **LLH5:** randomly assign and unassign several events. This is a ruin and recreate low-level heuristic which has a parameter that takes a value between 1 and 10. This parameter will be selected randomly each time this heuristic is invoked. The parameter represents the number of events to be assigned or unassigned. If, for example, the parameter has a value of 5, then LLH5 will select five events at random and then at each selected event the heuristic either applies LLH2, LLH3, or LLH4 (selected randomly with an equal distribution).
- **LLH6:** shuffle the assignment of several events. This heuristic has a parameter that takes a value between 1 and 10. This parameter will be selected randomly each time this heuristic is invoked. The parameter represents the number of events to be shuffled. If, for example, the parameter has a value of 5, then LLH6 will select five events at random and then shuffle their assignments.
- **LLH7:** split a randomly selected event into two events. This heuristic divides a randomly chosen event if it has an assignment of a time block of at least two consecutive time slots into two events such that their durations should sum to the duration of the original event. As an example, assume that a *Geography* class event with a duration of three is assigned to the first time slot on Monday; LLH7 will divide the teaching of *Geography* into two separate (still consecutive) time slots with one event having a duration of one and the other event having a duration of two. This low-level heuristic will allow for future moves to operate on those two events separately.
- **LLH8:** merge two randomly selected events adjacent in time and sharing the same events. This heuristic is the opposite of LLH7.
- **LLH9:** swap two random resources. As an example, assume that a *Geography* class event is assigned to *Class Room A* and the *History* class event is assigned to *Class Room B*. LLH9 will assign a *History* class to *Class Room A*, and a *Geography* class to *Class Room B*.
- **LLH10:** reschedule a resource element of an event. As an example, assume that *Class Room A* is assigned to a *Geography* class event. LLH10 could re-assign *Class Room A* to a *History* class.
- **LLH11:** randomly select an unassigned resource and then assign it at random. This low-level heuristic works similarly to LLH10; however, the resource is expected to be unassigned to begin with. As an example, assume that *Teacher A* is unassigned to any class. LLH11 will select a random class event and then it will assign *Teacher A* to the selected class.
- **LLH12:** randomly select an assigned resource and then unassign it. This low-level heuristic is the opposite of LLH11. As an example, assume that *Teacher A* is assigned to a given class. LLH12 will unassign *Teacher A* to the given class.

- **LLH13:** randomly assign and unassign several resources. This is a ruin and recreate low-level heuristic which has a parameter that takes a value between 1 and 10. This parameter will be selected randomly each time this heuristic is invoked. The parameter represents the number of resources to be assigned or unassigned. If, for example, the parameter has a value of 5, then LLH13 will select five resources at random and then at each selected resource the heuristic either applies LLH10, LLH11, or LLH12 (selected randomly with an equal distribution).
- **LLH14:** shuffle the assignment of several resources. This heuristic has a parameter that takes a value between 1 and 10. This parameter will be selected randomly each time this heuristic is invoked. The parameter represents the number of resources to be shuffled. If, for example, the parameter has a value of 5, then LLH14 will select five resources at random and then shuffle their assignments.

4.1 Sequence-Based Selection Hyper-Heuristic

The hyper-heuristic implemented in this study aims to select and apply sequences of heuristics instead of selecting and applying a single heuristic. This section introduces a modified framework for selection hyper-heuristics enabling the operation of selecting sequences of heuristics. Figure 1 illustrates how a generic sequence-based selection hyper-heuristic framework operates.

This study applies a hidden Markov model (Baum and Petrie, 1966) approach for the sequence-based selection method (see Figure 2). The goal is to learn and generate sequences of low-level heuristics where low-level heuristics represent hidden states of the model. Each state (low-level heuristic) has a transition probability to move to another state (or itself) and a sequence-based acceptance strategy probability to decide whether a sequence of heuristics is constructed. Therefore we define two matrices: one to store the scores, hence the probabilities, to move from a heuristic to another; and the other to store the scores of the acceptance strategy for each low-level heuristic. We refer to the first matrix as *TransitionScore* and the second as $ASScore$. Note that the Markov chain model employed in the work in Kheiri and Keedwell (2015a) is a simpler Markov model where the state is directly visible to the observer and therefore the state (low-level heuristic) transition probabilities are the only parameters.

Given the current low-level heuristic (llh_c), the hyper-heuristic uses a roulette wheel selection strategy to select the next low-level heuristic (llh_n) with a probability given by:

$$\frac{TransitionScore_{llh_c,llh_n}}{\sum_{\forall j}(TransitionScore_{llh_c,llh_j})}. \tag{3}$$

The hyper-heuristic will then select the sequence-based acceptance strategy l for the selected heuristic llh_n with a probability given by:

$$\frac{ASScore_{llh_n,l}}{\sum_{\forall j}(ASScore_{llh_n,j})}. \tag{4}$$

The sequence-based acceptance strategy (AS) has two options. If the first option is selected (i.e., $l = 1$), this means the sequence of heuristics is now completed and the heuristics in the sequence will be applied to the candidate solution to generate a new solution. The move acceptance of the hyper-heuristic will be applied to decide whether to accept or reject the new solution. The relevant transition and sequence-based acceptance

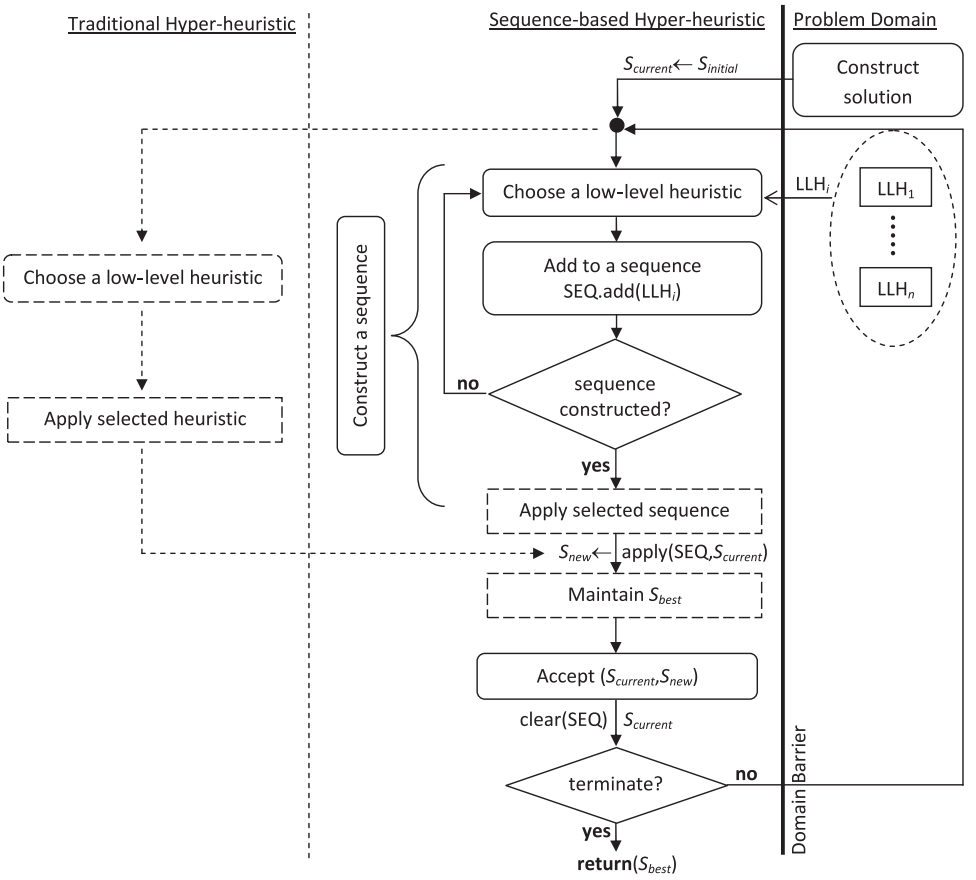


Figure 1: A sequence-based selection hyper-heuristic framework.

strategy scores will be updated where the new solution has a quality better than the quality of the best solution in hand. If the second option is selected (i.e., $l = 2$), then the selected heuristic will be added to the sequence of heuristics and no evaluation is conducted. Initially, the hyper-heuristic mechanism assigns an equal probability to move from any heuristic to another and the probability of selecting the associated sequence-based acceptance strategy in order to allow all the low-level heuristics to process the given solutions. In other words, $TransitionScore_{llh_i,llh_j} = 1$ for all i, j; and $ASScore_{llh_i,l} = 1$ for all i, l. These scores are updated as long as the best solution recorded in hand is improved. Consequently, after a number of steps the hyper-heuristic learns and detects a list of sequences of low-level heuristics that perform well. The hyper-heuristic assigns a higher probability of calling and applying these sequences and hence a lower probability of the use of heuristics that generate worsening results. In Kheiri and Keedwell (2015b), score values are simply increased by 1 as a reward mechanism as long as the best recorded solution in hand is improved.

We provide an example of how the developed method would work on five low-level heuristics. Figure 3 shows the initial score values of the two HMM matrices. For example, if llh_2 is initially selected as the current low-level heuristic and considering the scores in the transition matrix, the probability to move from llh_2 to any other heuristic is

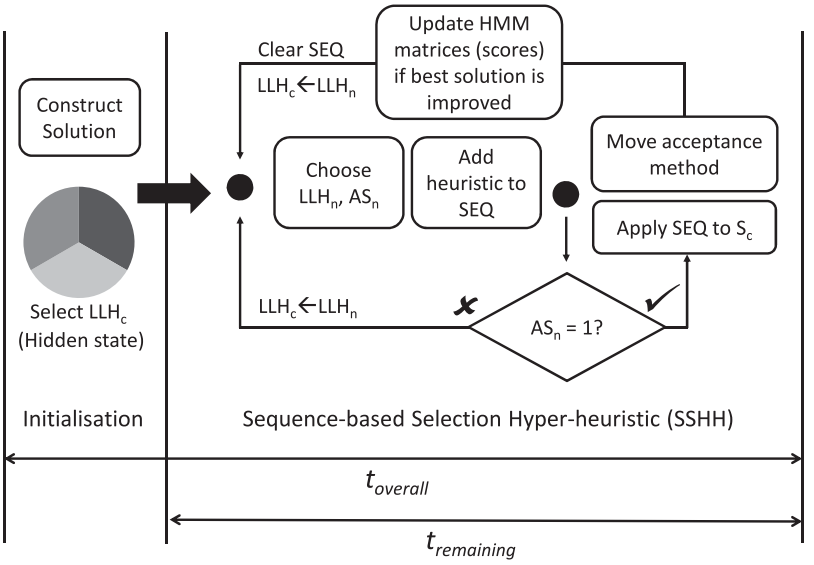


Figure 2: A sequence-based selection hyper-heuristic utilising HMM; where LLH_c and LLH_n are the current and next selected low-level heuristics, respectively, and AS_n is the next selected sequence-based acceptance strategy.

	llh_0	llh_1	llh_2	llh_3	llh_4
llh_0	1	1	1	1	1
llh_1	1	1	1	1	1
llh_2	1	1	1	1	1
llh_3	1	1	1	1	1
llh_4	1	1	1	1	1

TransitionScore

	l=1	l=2
llh_0	1	1
llh_1	1	1
llh_2	1	1
llh_3	1	1
llh_4	1	1

ASScore

Figure 3: Initial score values of the HMM matrices for five low-level heuristics.

1/5. The hyper-heuristic applies the roulette wheel selection method to select the next low-level heuristic. We assume that llh_1 is selected next. The hyper-heuristic will then select the sequence-based acceptance strategy of llh_1. The probability at this point of selecting any of the two acceptance strategies is 1/2. We assume that the hyper-heuristic selects the acceptance strategy $l = 2$, meaning that llh_1 will be added to the sequence but an evaluation is not conducted. We move on to the next step and that we are now on llh_1. Again, we select the next low-level heuristic using the scores in the transition matrix to move from llh_1 and in this example, llh_4 is selected. The probability of selecting any of the two acceptance strategies for llh_4 is 1/2. Assume that the hyper-heuristic uses the roulette wheel selection strategy to select the acceptance strategy $l = 1$. This means that the sequence of heuristics is now constructed and it will be applied to the current solution. The constructed sequence is llh_1, llh_4 meaning that a sequence of heuristics of

	llh_0	llh_1	llh_2	llh_3	llh_4
llh_0	1	1	1	1	1
llh_1	1	1	1	1	2
llh_2	1	2	1	1	1
llh_3	1	1	1	1	1
llh_4	1	1	1	1	1

TransitionScore

	l=1	l=2
llh_0	1	1
llh_1	1	2
llh_2	1	1
llh_3	1	1
llh_4	2	1

ASScore

Figure 4: Updated score values of the HMM matrices.

size 2 is generated. This sequence will be applied sequentially to the current solution, returning a new solution. Assume that the new solution is better than the current best obtained solution during the optimisation. In this case the relevant scores will be updated and increased by 1. The scores of moving from llh_2 to llh_1 and from llh_1 to llh_4 will be updated. Also the sequence-based acceptance strategy $l = 2$ of llh_1 and the sequence-based acceptance strategy $l = 1$ of llh_4 will be updated as illustrated in Figure 4. Note that if the new solution does not improve the quality of the best solution in hand, then the scores in the HMM matrices will not be updated. The next component of the hyper-heuristic is the move acceptance method. If, for example, we are using HC move acceptance, then the new solution will be accepted if its quality is better than the candidate solution; otherwise, the new solution will be rejected. We are now at llh_4, and we continue with the same strategy to construct and apply the next sequence of heuristics using the new scores and so on.

In this work we investigate the performance of three further variants of this method. They all adopt a similar selection strategy but differ in the reward mechanism.

- **Sequence-Based Selection Hyper-Heuristic with Linear Update (SSHH-L):** Corresponding score values are increased linearly by t as a reward mechanism as long as the best recorded solution in hand is improved; where t is the time elapsed in seconds.
- **Sequence-Based Selection Hyper-Heuristic with Nonlinear Update (SSHH-N):** Corresponding score values are increased nonlinearly by $e^{t/c}$ as a reward mechanism as long as the best recorded solution in hand is improved; where t is the time elapsed in seconds and c is a constant assigned arbitrarily to 30 in this work.
- **Sequence-Based Selection Hyper-Heuristic with Delta Update (SSHH-D):** Instead of a predefined rewarding mechanism, the difference in objective value between the newly generated solution and the best recorded solution after the application of the selected sequence of heuristics is used as a score value.

5 Results

5.1 Experimental Setup

The experimentation is conducted on an i7-4770K CPU at 3.50 GHz with a memory of 16.00 GB. The Mann–Whitney–Wilcoxon test (Fagerland and Sandvik, 2009; Kruskal,

1957) is performed with a 95% confidence level in order to compare pairwise performance variations of two given algorithms statistically. The following notations are used: Given algorithm A versus algorithm B, (i) $+$ $(-)$ denotes that A (B) is better than B (A) and this performance variance is statistically significant, (ii) A $\simeq$ B indicates that there is no statistical significant between A and B. In the first set of experiments, eight selected instances from the ITC 2011 set (one instance per each country) are used. The termination criterion is set to 200 seconds for the preliminary experiments.

5.2 Comparison of the Different Move Acceptance Methods

Table 2 summarises the performance of the sequence-based selection (SS) method with the different move acceptance methods, where the SS prefix refers to the sequence-based selection approach and the suffixes refer to the various move acceptance methods. As expected, the results confirm that the move acceptance method does not influence the overall performance greatly. Based on the Mann–Whitney–Wilcoxon test with respect to the averages over ten runs on the selected instances, there are no statistically significant performance differences between the different methods on almost all instances. The only exception is on the South Africa-Woodlands2009 instance, at which simulated annealing seems to perform better than the others.

Based on the ranking strategy employed during the second round of ITC 2011, the sequence-based selection method, when combined with the record-to-record travel move acceptance method, seems to perform slightly better than the other hyper-heuristic methods, as shown in Figure 5. Hence, SS-RR is taken under consideration from this point onwards for further performance analysis, and we will refer to SS-RR as SSHH.

5.3 Comparison of SSHH to the Parametrised Sequence Approaches

In this section, we describe a set of exhaustive experiments to determine the extent to which sequences of heuristics are useful in comparison to single heuristics and compare this with SSHH's ability to discover these online. We implemented a simple hyper-heuristic to create and evaluate sequences of heuristics rather than applying single heuristics. Given n low-level heuristics $\{LLH_0, \ldots, LLH_{n-1}\}$, we form other sequences of heuristics of size 2 and size 3 and then invoke them successively. The total number of sequences of low-level heuristics is $n + n^2 + n^3$ in the overall: $\{LLH_0, \ldots, LLH_{n-1}, LLH_0 + LLH_0, LLH_0 + LLH_1, \ldots, LLH_{n-1} + LLH_{n-1}, LLH_0 + LLH_0 + LLH_0, \ldots, LLH_{n-1} + LLH_{n-1} + LLH_{n-1}\}$, where $LLH_i + LLH_j + LLH_k$ denotes the sequence of applying LLH_i followed by LLH_j and followed by LLH_k.

We are able to present results for a sequence size of up to 3 where 3,615 permutations are investigated. There are 5,4240 permutations of four heuristics and an analysis of this space is too computationally complex to execute here. The proposed SSHH does not take the size of sequences as a parameter, but rather it learns the optimum size during the optimisation (in an online manner).

One thousand solutions are randomly generated and applied to each sequence to these generated solutions over eight ITC 2011 instances (a representative instance from each country). The utilisation rate for each sequence is equal but that does not mean that applying all these sequences would improve the quality of the input solutions. We therefore computed the utilisation rate considering only improving moves per sequence and for each instance. Table 3 provides the top five sequences that generate the highest utilisation rate of improving moves. From the table, we observe that sequences of size 3 are the dominants. This is perhaps to be expected as a greater movement in

Table 2: Pairwise performance comparison of SS-HC, SS-SA, SS-GD, SS-RR, and SS-LA (row vs column) using the Mann–Whitney–Wilcoxon test based on the average over ten runs for eight instances.

Brazil-Instance2 (BR) and Finland-ElementarySchool (FI) instances

BR	SS-HC	SS-SA	SS-GD	SS-RR	SS-LA	FI	SS-HC	SS-SA	SS-GD	SS-RR	SS-LA
SS-HC		≃	≃	≃	≃	SS-HC		≃	≃	≃	≃
SS-SA	≃		≃	≃	≃	SS-SA	≃		≃	≃	≃
SS-GD	≃	≃		≃	≃	SS-GD	≃	≃		≃	≃
SS-RR	≃	≃	≃		≃	SS-RR	≃	≃	≃		≃
SS-LA	≃	≃	≃	≃		SS-LA	≃	≃	≃	≃	

Greece-Aigio1stHighSchool2010 (GR) and Italy-Instance4 (IT) instances

GR	SS-HC	SS-SA	SS-GD	SS-RR	SS-LA	IT	SS-HC	SS-SA	SS-GD	SS-RR	SS-LA
SS-HC		≃	≃	≃	≃	SS-HC		≃	≃	≃	≃
SS-SA	≃		≃	≃	≃	SS-SA	≃		≃	≃	≃
SS-GD	≃	≃		≃	≃	SS-GD	≃	≃		≃	≃
SS-RR	≃	≃	≃		≃	SS-RR	≃	≃	≃		≃
SS-LA	≃	≃	≃	≃		SS-LA	≃	≃	≃	≃	

Kosova-Instance1 (KS) and Netherlands-Kottenpark2009 (NL) instances

KS	SS-HC	SS-SA	SS-GD	SS-RR	SS-LA	NL	SS-HC	SS-SA	SS-GD	SS-RR	SS-LA
SS-HC		≃	≃	≃	≃	SS-HC		≃	≃	≃	≃
SS-SA	≃		≃	≃	≃	SS-SA	≃		≃	≃	≃
SS-GD	≃	≃		≃	≃	SS-GD	≃	≃		≃	≃
SS-RR	≃	≃	≃		≃	SS-RR	≃	≃	≃		≃
SS-LA	≃	≃	≃	≃		SS-LA	≃	≃	≃	≃	

South Africa-Woodlands2009 (ZA) and Spain-School (ES) instances

ZA	SS-HC	SS-SA	SS-GD	SS-RR	SS-LA	ES	SS-HC	SS-SA	SS-GD	SS-RR	SS-LA
SS-HC		−	−	−	≃	SS-HC		≃	≃	≃	≃
SS-SA	+		+	+	+	SS-SA	≃		≃	≃	≃
SS-GD	+	−		≃	+	SS-GD	≃	≃		≃	≃
SS-RR	+	−	≃		+	SS-RR	≃	≃	≃		≃
SS-LA	≃	−	−	−		SS-LA	≃	≃	≃	≃	

search space can be accomplished with the application of three well-selected heuristics, although poorer performance could also perhaps be expected if the heuristics are not well matched. It is clear from Table 3 that heuristic 5 is particularly important for generating new and better solutions in these problems, and the 3-fold combination of LLH5 appears top for 4/8 instances. However, it should also be noted that for the other half, LLH5 is best combined with other heuristics to deliver optimal performance. This study of heuristic sequence behaviour is important in determining heuristic performance in context as part of the overall search process, rather than as single application events.

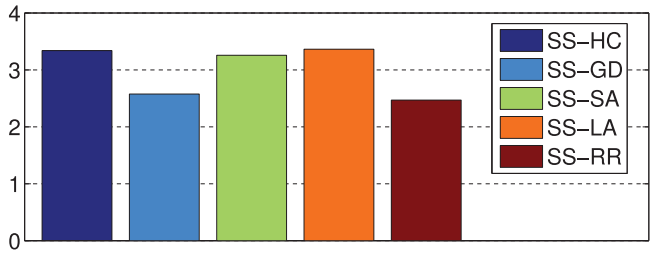


Figure 5: Comparisons of the different move acceptance methods when combined with the sequence-based selection method. The vertical axis shows the score values using ITC 2011 ranking strategy over ten runs from eight instances.

Table 3: The performance of applying sequences of low-level heuristics of sizes one, two, and three and applying them on randomly generated solutions. LLHi-j-k denotes the sequence $LLH_i + LLH_j + LLH_k$.

Instance	First	Second	Third	Fourth	Fifth
Brazil-Instance2	LLH5-5-5	LLH1-5-5	LLH5-1-5	LLH5-0-5	LLH1-5-4
Finland-ElementarySchool	LLH5-5-5	LLH5-5-1	LLH1-5-5	LLH5-0-5	LLH5-1-5
Greece-Aigio1stHighSchool2010	LLH5-5-5	LLH6-5-5	LLH5-5-6	LLH5-6-5	LLH5-4-5
Italy-Instance4	LLH1-5-0	LLH5-0-1	LLH5-0-0	LLH4-5-5	LLH5-13-0
Kosova-Instance1	LLH4-5-1	LLH5-4-4	LLH0-4-6	LLH5-4-0	LLH4-5-4
Netherlands-Kottenpark2009	LLH1-5-10	LLH4-5-5	LLH5-0-5	LLH5-3-5	LLH0-0-5
South Africa-Woodlands2009	LLH5-5-5	LLH5-0-5	LLH0-5-5	LLH5-5-1	LLH1-5-5
Spain-School	LLH1-5-4	LLH5-4-4	LLH4-5-0	LLH4-4-5	LLH5-4-1

Now that the potential for applying sequences of heuristics has been established, the performance of two hyper-heuristic methods is compared; one method has a fixed parametrised sequence size and the other is SSHH, which learns the optimum sequence lengths automatically during the search process. The selection method of the fixed parametrised hyper-heuristic (FPHH) method selects at each decision point the size of the sequence l randomly (limited to three in this work), and then selects randomly a sequence of size l and applies it to the candidate solution. The same record-to-record travel move acceptance method employed for SSHH is used in FPHH. Table 4 provides the ranking score of SSHH and FPHH for eight ITC 2011 selected instances over ten runs using the ITC 2011 ranking method. The table shows that SSHH performs the best in nine instances including two draws. This provides evidence that the size of the sequences of heuristics should not be fixed, but rather that the hyper-heuristic method should take the responsibility of detecting the optimum size of sequences during the optimisation.

5.4 Comparison of the Different Variants of SSHH

The performance of the different variants of SSHH described in Section 4 are investigated over eight ITC 2011 instances (a representative instance from each country). Table 5 and Figure 6 summarise the performance of each variant based on ten runs for each selected instance using the ranking strategy employed in the second round

Table 4: Score of SSHH and FPHH for each selected instance over ten runs which is computed using the scoring scheme utilised in the second round of the ITC 2011 competition for ranking different approaches. The best score values are highlighted in bold.

Instance	SSHH	FPHH
Brazil-Instance2	**1.45**	1.55
Finland-ElementarySchool	**1.50**	**1.50**
Greece-Aigio1stHighSchool2010	**1.20**	1.80
Italy-Instance4	**1.40**	1.60
Kosova-Instance1	**1.20**	1.80
Netherlands-Kottenpark2009	**1.40**	1.60
South Africa-Woodlands2009	1.60	**1.40**
Spain-School	**1.50**	**1.50**

Table 5: Score of each SSHH variant for each selected instance over ten runs which is computed using the scoring scheme utilised in the second round of the ITC 2011 competition for ranking different approaches. The best score values are highlighted in bold.

Instance	SSHH	SSHH-L	SSHH-N	SSHH-D
Brazil-Instance2	2.55	**1.70**	2.95	2.80
Finland-ElementarySchool	2.70	**2.10**	2.70	2.50
Greece-Aigio1stHighSchool2010	3.10	2.15	2.80	**1.95**
Italy-Instance4	2.80	2.30	2.70	**2.20**
Kosova-Instance1	3.00	2.30	**2.00**	2.70
Netherlands-Kottenpark2009	2.20	**2.15**	**2.15**	3.50
South Africa-Woodlands2009	3.05	2.15	**1.95**	2.85
Spain-School	3.20	**1.80**	2.80	2.20

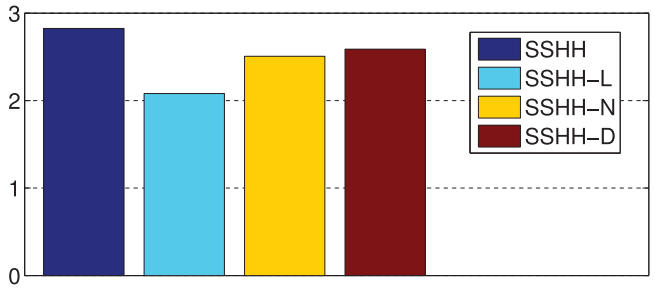
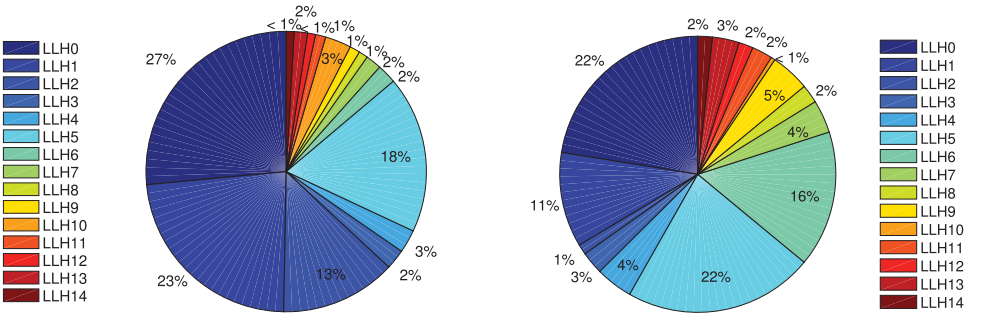


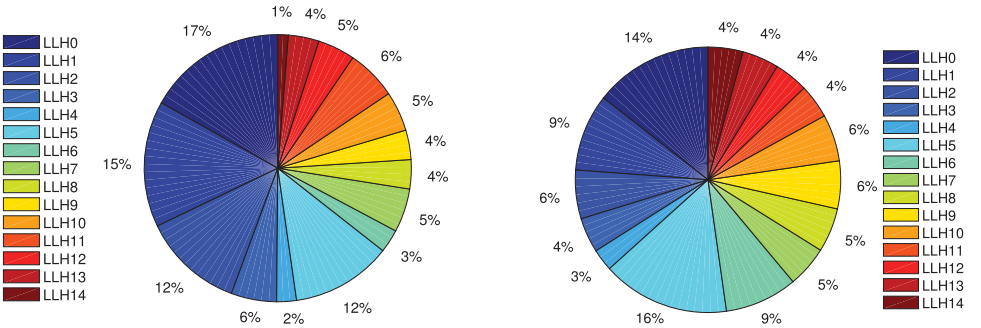
Figure 6: Overall scores of the different variants of SSHH using ITC 2011 ranking strategy over ten runs from eight instances.

of the ITC 2011 competition. The results show that SSHH-L with linear update performs overall better than SSHH, SSHH-N, and SSHH-D. SSHH-L generates the best score in four instances including one draw. SSHH-N obtains the best results in three instances including a draw. The ranking results put SSHH-D third with best score in two



(a) Spain-School (b) South Africa-Woodlands2009

Figure 7: Average utilisation rate of each low-level heuristic considering only invocations that generated improvement on the best-of-run solution and AS = 1 from ten runs using SSHH while solving Spain-School and South Africa-Woodlands2009 instances.



(a) Spain-School (b) South Africa-Woodlands2009

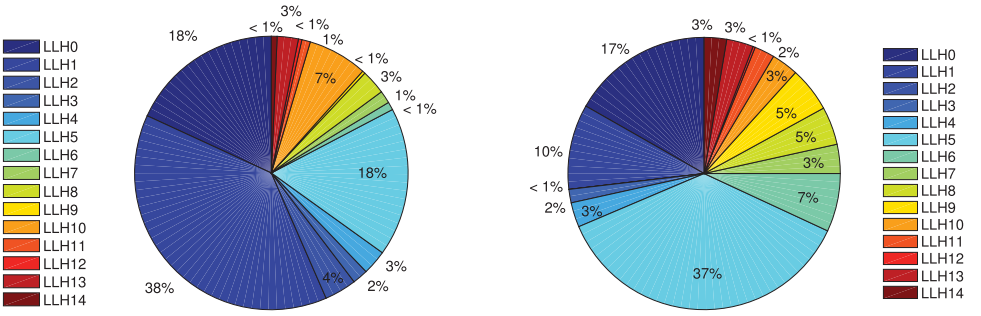
Figure 8: Average utilisation rate of each low-level heuristic considering only invocations that generated improvement on the best-of-run solution and AS = 1 and 2 from ten runs using SSHH while solving Spain-School and South Africa-Woodlands2009 instances.

instances. SSHH performs worse than the other algorithms considering that the number of instances for which it produces the best ranking score is zero.

The best and the worst performing methods (SSHH-L and SSHH) are taken forward for the performance comparison to previously proposed approaches.

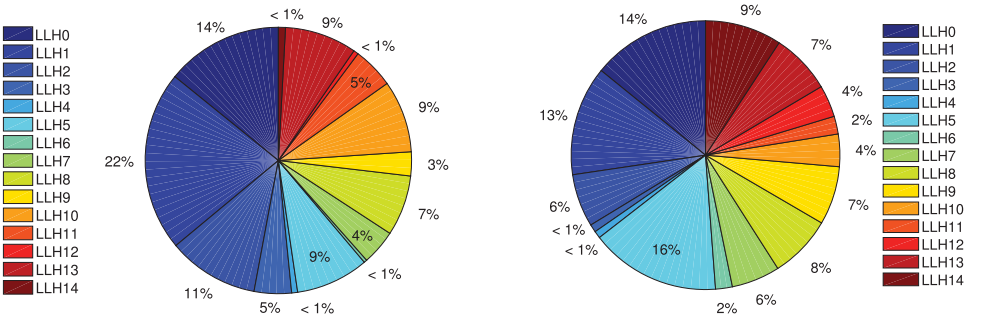
5.5 An Analysis of SSHH and SSHH-L Methods

Figure 7 depicts the average heuristic utilisation rate using SSHH over ten runs of each low-level heuristic considering only invocations that generated improvement on the best-of-run solution and AS = 1 while solving two selected sample instances of Spain-School and South Africa-Woodlands2009. Figure 8 shows the same but considering both AS = 1 and AS = 2. It can be observed in Figure 7 that LLH0, LLH1, and LLH5 are the most successful heuristics, generating the highest utilisation rate in both instances. LLH2 seems to perform well in the Spain-School instance, and the same applies to LLH6 in the South Africa-Woodlands2009 instance. The remaining



(a) Spain-School (b) South Africa-Woodlands2009

Figure 9: Average utilisation rate of each low-level heuristic considering only invocations that generated improvement on the best-of-run solution and AS = 1 from ten runs using SSHH-L while solving Spain-School and South Africa-Woodlands2009 instances.



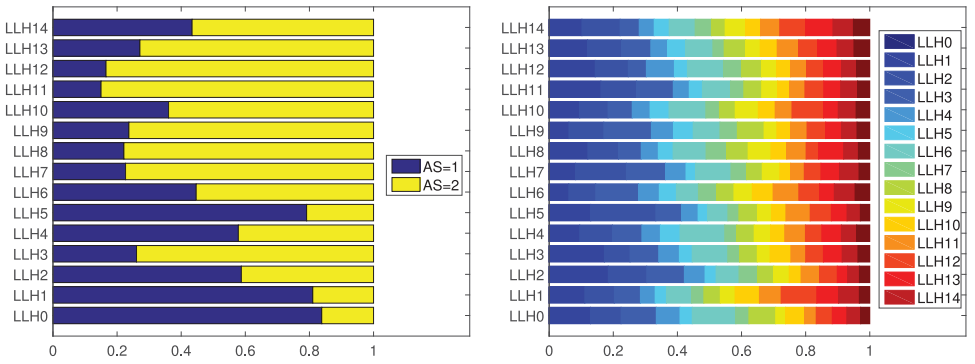
(a) Spain-School (b) South Africa-Woodlands2009

Figure 10: Average utilisation rate of each low-level heuristic considering only invocations that generated improvement on the best-of-run solution and AS = 1 and 2 from ten runs using SSHH-L while solving Spain-School and South Africa-Woodlands2009 instances.

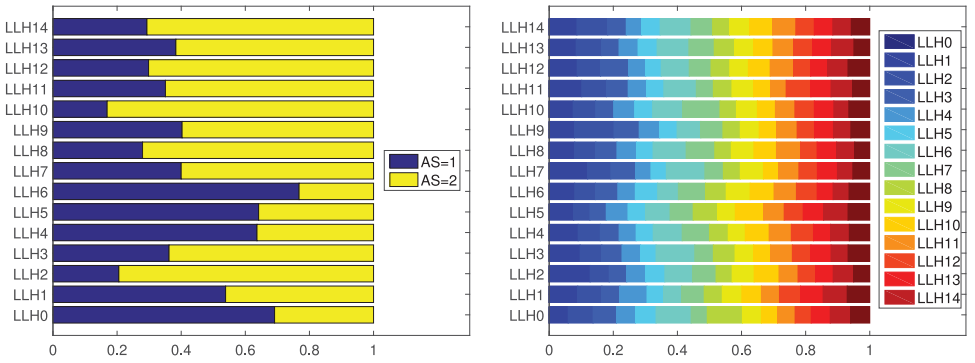
heuristics appear to provide poor performance. One may notice from Figure 7 that the most successful heuristics are event-oriented operators (LLH0-LLH8) rather than resource-oriented operators (LLH9-LLH14). Having said that and by examining Figure 8, the results show that resource-oriented operators are useful when combined and applied within a sequence of operations.

We repeat the same experiments for SSHH-L. Figures 9 and 10 provide the average utilisation rate when AS = 1 and AS = 1 and 2, respectively, while solving two selected sample instances of Spain-School and South Africa-Woodlands2009. Again and from Figure 9, LLH0, LLH1, and LLH5 are the most successful heuristics, generating the highest utilisation rate in both instances. We also observe from Figure 10 the involvement of resource-oriented operators showing that they are useful when combined and applied within a sequence of heuristics.

Figure 11 provides the average probabilities of the HMM matrices for each low-level heuristic over ten runs using SSHH while solving two selected sample problem



(a) Spain-School

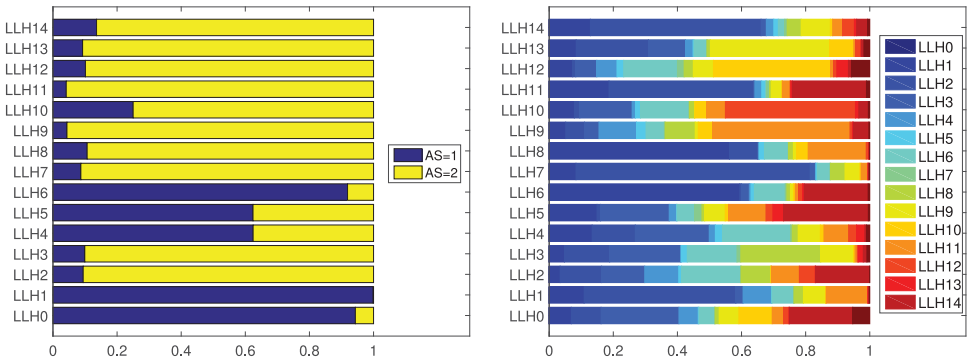


(b) South Africa-Woodlands2009

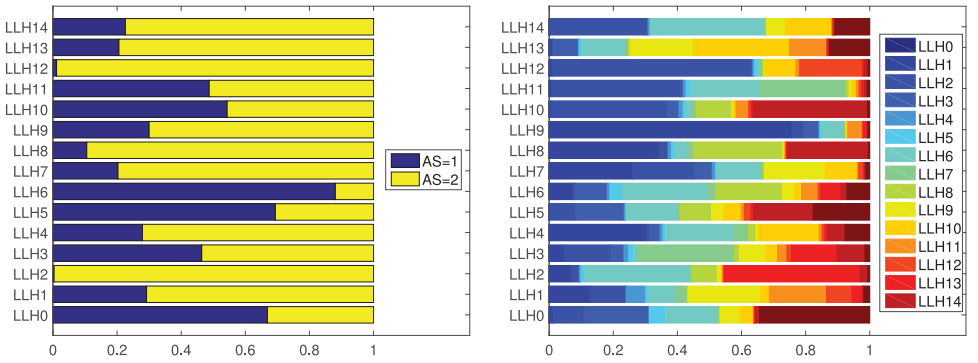
Figure 11: Average probabilities of the HMM matrices for each low-level heuristic from ten runs using SSHH while solving Spain-School and South Africa-Woodlands2009 instances.

instances of Spain-School and South Africa-Woodlands2009. By examining the transition figure for Spain-School instance, the following exploitative heuristics LLH0, LLH5, and LLH1 appear to be more successful than others in delivering best-of-run solutions as shown by the probability matrix in the South Africa-Woodlands2009 problem instance. In the latter, LLH9 is an exploration heuristic that needs to be combined with (preferably) LLH1.

Figure 12 shows the average probabilities of the HMM matrices for each low-level heuristic over ten runs using SSHH-L while solving the same two selected instances. In the Spain-School instance, the heuristic LLH1 seems to dominate the search process with the help of several exploration heuristics such as LLH2, LLH7, LLH11, and LLH13. Sets of likely sequences have been generated using a roulette wheel simulator (Algorithm 1) and the final HMM probability matrices as input for the Spain-School and South-Africa-Woodlands2009 problem instances. Table 6 summarises the results. Although single heuristics are frequently used, the approach clearly identifies sequences of size 2 and 3 as useful to the search. In the South Africa-Woodlands2009 instance, the sequences LLH14-LLH5 and LLH9-LLH0 are identified amongst the top ten generated sequences. An interesting finding is that the exploration heuristics LLH9 and LLH14 are resource-oriented operators whilst LLH0 and LLH5 are event-oriented operators. In the



(a) Spain-School



(b) South Africa-Woodlands2009

Figure 12: Average probabilities of the HMM matrices for each low-level heuristic from ten runs using SSHH-L while solving the Spain-School and South Africa-Woodlands2009 instances.

Spain-School instance, the sequence of size 3, LLH13-LLH9-LLH0, is also identified being generated 153 times. It should be noted here that this represents the set of sequences generated from the final learned matrix for each problem and the approach may also have identified alternative sequences during the search process.

5.6 Comparison of SSHH and SSHH-L to the Best-Known Methods

Although the ITC 2011 set of instances is deprecated, we would like to obtain an approximate idea of the relative performance of SSHH and SSHH-L with respect to ITC 2011 solvers. For the comparisons with ITC 2011 solvers, a trial terminates after the equivalent to $timeLimit = 1{,}000$ seconds is reached as the competition requires. A benchmarking software tool provided at the competition website is used to report the equivalent time value in the used machine. It reported that our machine should take 492 seconds per run. Using the results of the second round of ITC 2011 solvers provided at the competition website, Figure 13 summarises the performance comparison of SSHH and SSHH-L to the ITC 2011 methods (GOAL, HySST, Lectio, and HFT) over 18 ITC 2011 instances, each with ten runs. The results show that the SSHH-L method is the winner with a score of 2.22 in the overall, with SSHH slightly behind, indicating that

Algorithm 1: Simulator

Let $LLH = \{llh_1, llh_2, \ldots\}$ represent set of low-level heuristics;
Let $TransitionScore_{llh_c, llh_n}$ represent score of moving from llh_c to llh_n;
Let $ASScore_{llh_n, l}$ represent score of applying acceptance strategy l for llh_n;
Let S represent the number of sequences to be constructed;
$llh_n \leftarrow SelectRandom(LLH)$; $AS \leftarrow 2$;
for $i \leftarrow 1, 2, \ldots, S$ **do**
    $SEQ.Clear()$;
    while $AS == 2$ **do**
        $llh_c \leftarrow llh_n$;
        $llh_n \leftarrow RouletteWheel(TransitionScore, llh_c)$;
        $AS \leftarrow RouletteWheel(ASScore, llh_n)$;
        $SEQ.Add(llh_n)$;
    end
    $RECORD.Add(SEQ)$;
end
return $RECORD$;

Table 6: The top ten constructed sequences of low-level heuristics while solving Spain-School and South Africa-Woodlands2009 instances using the developed simulator.

Spain-School		South Africa-Woodlands2009	
Sequence	Count	Sequence	Count
LLH_1	2759	LLH_5	1021
LLH_0	1081	LLH_0	595
LLH_5	394	LLH_1	338
$LLH_8 + LLH_0$	333	LLH_{14}	326
LLH_{10}	227	$LLH_{14} + LLH_5$	263
$LLH_7 + LLH_1$	201	$LLH_2 + LLH_5$	259
$LLH_{13} + LLH_1$	198	LLH_6	229
$LLH_{13} + LLH_8 + LLH_0$	153	LLH_{10}	200
$LLH_2 + LLH_1$	145	LLH_{13}	189
$LLH_{11} + LLH_1$	134	$LLH_9 + LLH_0$	187

the sequence selection–based approach produces best-in-class results on this problem set.

Table 7 summarises the results on XHSTT-2014 benchmark over ten runs. The solvers (SSHH and SSHH-L) terminate as long as there is no improvement to the best solution in hand for 200 seconds. At the time of submission, our solvers managed to deliver 9 best-known solutions, and match the best-known solutions in 4 instances. The performance is superior on most of the large instances including those from Australia, Denmark, the Netherlands, and the USA. They also perform well on the problem instances from Greece and Kosova. The performance on the instances from Brazil, Spain, Finland, England, and Italy is average but still our solvers can deliver near

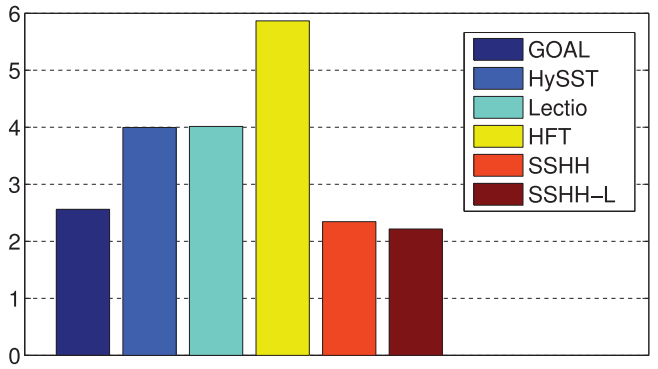


Figure 13: Comparisons of ITC solvers, SSHH and SSHH-L based on ITC 2011 ranking strategy over ten runs for each ITC 2011 instances.

best-known solutions. However, they cannot generate the best or near the best-known solutions on the problem instances from South Africa.

Considering the averages performance, SSHH-L is the winner in 18 instances including 3 ties; while SSHH wins in 10 instances including 3 ties. Overall, the linear nature of SSHH-L and its ability to weight improvements later in the search as more important appears to preserve diversity and improve on the standard SSHH implementation.

6 Conclusion

The goal in hyper-heuristic research is to raise the level of generality by offering search methodologies that are easier to use and cheaper to implement and maintain than knowledge-intensive methods and yet deliver better average-case performance across a range of problems.

A sequence-based selection hyper-heuristic framework is introduced in this study whose selection component aims to discover sequences of heuristics.

In this article it has been shown:

- That the selection strategy of a hyper-heuristic is more important than the move acceptance method for this range of problems.
- That sequences of heuristics are able to deliver improved performance over single applications of a heuristic and that some problem instances benefit from the use of multiple heuristic types working together in sequence.
- That sequence lengths are better to be optimised on a per instance basis rather than randomly chosen.
- That an approach that prioritises heuristics that are able to generate better solutions towards the end of the search performs better than an approach that does not. This stands to reason because, as the search progresses, it becomes increasingly difficult to find solutions that achieve better performance than the current solution and so those heuristics capable of discovering these later in the search should be rewarded accordingly. This method also provides a mechanism for the hyper-heuristic to modify its learned behaviour later in the

Table 7: The performance of SSHH and SSHH-L over ten trials showing the average and the best-obtained solution in terms of *hardViolationScore*, *softViolationScore* on XHSTT-2014 benchmark. The table also shows the current best-known quality of solution and the highest-known lower bound, as reported in the benchmark website (at time of submission, April 2015), per each instance. The best cost values are highlighted in bold.

Instance	SSHH avg	SSHH best	SSHH-L avg	SSHH-L best	Best known	Lower bound
Australia-BGHS98	5,548	0,520	5,542	**0,493**	1,386	0,0
Australia-SAHS96	2,15	**0,2**	2,15	**0,2**	0,24	0,0
Australia-TES99	2,99	**0,61**	2,89	0,65	0,125	0,0
Brazil-Instance2	0,48	0,10	0,35	0,10	**0,5**	0,5
Brazil-Instance4	7,116	2,117	6,114	2,117	**0,51**	0,51
Brazil-Instance6	0,163	0,101	0,129	0,101	**0,35**	0,35
Denmark-FalkonergaardensGymnasium2012	0,1807	**0,1522**	0,1806	**0,1522**	0,3310	0,285
Denmark-HasserisGymnasium2012	12,2863	**12,2628**	12,2916	12,2641	12,3124	0,7
Denmark-VejenGymnasium2009	2,2829	2,2731	2,2827	**2,2720**	2,4097	0,0
Spain-School	0,1025	0,517	0,940	0,517	**0,336**	0,334
Finland-College	3,49	0,8	2,22	0,8	**0,0**	0,0
Finland-HighSchool	0,43	0,7	0,27	0,7	**0,1**	0,0
Finland-SecondarySchool	0,103	0,89	0,105	0,89	**0,83**	0,77
Greece-HighSchool1	**0,0**	**0,0**	**0,0**	**0,0**	**0,0**	0,0
Greece-ThirdHighSchoolPatras2010	1,29	**0,0**	1,20	**0,0**	**0,0**	0,0
Greece-WesternUniversityInstance4	0,16	**0,4**	0,13	**0,4**	**0,4**	0,0
Italy-Instance4	0,169	0,38	0,62	0,38	**0,34**	0,27
Kosova-Instance1	0,18	**0,3**	0,18	**0,3**	**0,3**	0,0
Netherlands-Kottenpark2003	0,934	**0,466**	0,933	**0,466**	0,617	0,0
Netherlands-Kottenpark2005	4,1175	**0,811**	4,1161	**0,811**	0,1078	0,89
Netherlands-Kottenpark2009	12,14061	2,7495	13,8500	2,8505	**0,9180**	0,170
England-StPaul	29,1235	19,1294	31,1212	19,1306	**16,2258**	0,0
USA-Westside2009	0,582	**0,512**	0,584	**0,512**	0,697	0,0
South Africa-Lewitt2009	2,57	0,52	4,48	1,104	**0,0**	0,0
South Africa-Woodlands2009	14,0	9,0	15,0	10,0	**0,0**	0,0

search, something that is possible, but less prevalent in the standard SSHH method.

In conclusion, the empirical results indicate that the proposed method is powerful and an effective general search methodology performing better than the current state-of-the-art methods in solving high school timetabling problems.

Future work will focus on the automated adaptation of the single parameter of the method within the move acceptance component.

Acknowledgements

This work was supported by EPSRC grant EP/K000519/1. The authors would like to thank David Walker for developing the simulator presented in the results section.

References

Abramson, D. (1991). Constructing school timetables using simulated annealing: Sequential and parallel algorithms. *Management Science*, 37(1):98–113.

Abramson, D. A., Dang, H., and Krisnamoorthy, M. (1999). Simulated annealing cooling schedules for the school timetabling problem. *Asia-Pacific Journal of Operational Research*, 16(1):1–22.

Ahmed, L. N., Özcan, E., and Kheiri, A. (2015). Solving high school timetabling problems worldwide using selection hyper-heuristics. *Expert Systems with Applications*, 42(13):5463–5471.

Alvarez-Valdés, R., Parreño, F., and Tamarit, J. M. (2002). A tabu search algorithm for assigning teachers to courses. *TOP*, 10(2):239–259.

Baum, L. E., and Petrie, T. (1966). Statistical inference for probabilistic functions of finite state Markov chains. *The Annals of Mathematical Statistics*, 37(6):1554–1563.

Beligiannis, G. N., Moschopoulos, C. N., Kaperonis, G. P., and Likothanassis, S. D. (2008). Applying evolutionary computation to the school timetabling problem: The Greek case. *Computers and Operations Research*, 35(4):1265–1280.

Bello, G. S., Rangel, M. C., and Boeres, M. C. S. (2008). An approach for the class/teacher timetabling problem. In *Proceedings of the 7th International Conference on the Practice and Theory of Automated Timetabling*, pp. 1–6.

Bilgin, B., Özcan, E., and Korkmaz, E. E. (2007). An experimental study on hyper-heuristics and exam scheduling. In E. K. Burke and H. Rudová (Eds.), *Practice and Theory of Automated Timetabling VI*, volume 3867 of *Lecture Notes in Computer Science*, pp. 394–412.

Birbas, T., Daskalaki, S., and Housos, E. (2009). School timetabling for quality student and teacher schedules. *Journal of Scheduling*, 12(2):177–197.

Bufé, M., Fischer, T., Gubbels, H., Häcker, C., Hasprich, O., Scheibel, C., Weicker, K., Weicker, N., Wenig, M., and Wolfangel, C. (2001). Automated solution of a highly constrained school timetabling problem—Preliminary results. In E. J. W. Boers (Ed.), *Applications of Evolutionary Computing*, volume 2037 of *Lecture Notes in Computer Science*, pp. 431–440.

Burke, E. K., Gendreau, M., Hyde, M., Kendall, G., Ochoa, G., Özcan, E., and Qu, R. (2013). Hyper-heuristics: A survey of the state of the art. *Journal of the Operational Research Society*, 64(12):1695–1724.

Calderia, J. P., and Ross, A. C. (1997). School timetabling using genetic search. In *Proceedings of the International Conference on the Practice and Theory of Automated Timetabling*, pp. 115–122.

Colorni, A., Dorigo, M., and Maniezzo, V. (1992). A genetic algorithm to solve the timetable problem. Technical Report Technical Report No. 90-060, Politecnico di Milano, Italy.

Cowling, P., Kendall, G., and Soubeiga, E. (2001). A hyperheuristic approach to scheduling a sales summit. In E. Burke and W. Erben (Eds.), *Practice and Theory of Automated Timetabling III*, volume 2079 of *Lecture Notes in Computer Science*, pp. 176–190.

Domrös, J., and Homberger, J. (2012). An evolutionary algorithm for high school timetabling. In *Proceedings of the 9th International Conference on the Practice and Theory of Automated Timetabling*, pp. 485–488.

Dueck, G. (1993). New optimization heuristics: The great deluge algorithm and the record-to-record travel. *Journal of Computational Physics*, 104(1):86–92.

Even, S., Itai, A., and Shamir, A. (1976). On the complexity of timetable and multicommodity flow problems. *SIAM Journal on Computing*, 5(4):691–703.

Fagerland, M. W., and Sandvik, L. (2009). The Wilcoxon–Mann–Whitney test under scrutiny. *Statistics in Medicine*, 28(10):1487–1497.

Filho, G. R., Antonio, L., and Lorena, L. A. N. (2001). A constructive evolutionary approach to school timetabling. In E. J. W. Boers (Ed.), *Applications of Evolutionary Computing*, volume 2037 of *Lecture Notes in Computer Science*, pp. 130–139.

Fonseca, G. H. G., Santos, H. G., and Carrano, E. G. (2015). Late acceptance hill-climbing for high school timetabling. *Journal of Scheduling*, pp. 1–13.

Fonseca, G. H. G., Santos, H. G., Toffolo, T. Â. M., Brito, S. S., and Souza, M. J. F. (2014). GOAL solver: A hybrid local search based solver for high school timetabling. *Annals of Operations Research*, pp. 1–21.

Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. *Computers and Operations Research*, 13(5):533–549.

Hajek, B. (1988). Cooling schedules for optimal annealing. *Mathematics of Operations Research*, 13(2):311–329.

Holland, J. (1975). *Adaptation in natural and artificial systems*. Ann Arbor, MI: University of Michigan Press.

Jacobsen, F., Bortfeldt, A., and Gehring, H. (2006). Timetabling at German secondary schools: Tabu search versus constraint programming. In *Proceedings of the 6th International Conference on the Practice and Theory of Automated Timetabling*, pp. 439–442.

Kalender, M., Kheiri, A., Özcan, E., and Burke, E. K. (2013). A greedy gradient-simulated annealing selection hyper-heuristic. *Soft Computing*, 17(12):2279–2292.

Kendall, G., and Mohamad, M. (2004). Channel assignment optimisation using a hyper-heuristic. In *Proceedings of the 2004 IEEE Conference on Cybernetic and Intelligent Systems*, pp. 790–795.

Kheiri, A. (2014). *Multi-stage hyper-heuristics for optimisation problems*. PhD thesis, University of Nottingham, School of Computer Science.

Kheiri, A., and Keedwell, E. (2015a). Markov chain selection hyper-heuristic for the optimisation of constrained magic squares. *15th UK Workshop on Computational Intelligence*.

Kheiri, A., and Keedwell, E. (2015b). A sequence-based selection hyper-heuristic utilising a hidden Markov model. In *Proceedings of the 2015 on Genetic and Evolutionary Computation Conference, GECCO'15*, pp. 417–424.

Kheiri, A., Keedwell, E., Gibson, M. J., and Savic, D. (2015). Sequence analysis-based hyper-heuristics for water distribution network optimisation. *Procedia Engineering*, 119: 1269–1277.

Kheiri, A., Özcan, E., and Parkes, A. J. (2016). A stochastic local search algorithm with adaptive acceptance for high-school timetabling. *Annals of Operations Research*, 239(1):135–151.

Kingston, J. H. (2005). A tiling algorithm for high school timetabling. In E. Burke and M. Trick (Eds.), *Practice and Theory of Automated Timetabling V*, volume 3616 of *Lecture Notes in Computer Science*, pp. 208–225.

Kingston, J. H. (2014). KHE14: An algorithm for high school timetabling. In *Proceedings of the 10th International Conference on the Practice and Theory of Automated Timetabling*, pp. 269–291.

Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, 220(4598):671–680.

Kruskal, W. H. (1957). Historical notes on the Wilcoxon unpaired two-sample test. *Journal of the American Statistical Association*, 52(279):356–360.

Lara, C., Flores, J. J., and Calderon, F. (2008). Solving a school timetabling problem using a bee algorithm. In A. Gelbukh and E. F. Morales (Eds.), *MICAI 2008: Advances in Artificial Intelligence*, volume 5317 of *Lecture Notes in Computer Science*, pp. 664–674.

Melício, F., Calderia, J. P., and Rosa, A. (2006). THOR: A tool for school timetabling. In *Proceedings of the 6th International Conference on the Practice and Theory of Automated Timetabling*, pp. 532–535.

Moura, A. V., and Scaraficci, R. A. (2010). A GRASP strategy for a more constrained school timetabling problem. *International Journal of Operational Research*, 7(2):152–170.

Özcan, E., Bilgin, B., and Korkmaz, E. E. (2008). A comprehensive analysis of hyper-heuristics. *Intelligent Data Analysis*, 12(1):3–23.

Özcan, E., Bykov, Y., Birben, M., and Burke, E. K. (2009). Examination timetabling using late acceptance hyper-heuristics. In *IEEE Congress on Evolutionary Computation*, pp. 997–1004.

Pillay, N. (2013). A survey of school timetabling research. *Annals of Operations Research*, pp. 1–33.

Post, G., Ahmadi, S., Daskalaki, S., Kingston, J. H., Kyngas, J., Nurmi, C., and Ranson, D. (2012). An XML format for benchmarks in high school timetabling. *Annals of Operations Research*, 194(1):385–397.

Post, G., Di Gaspero, L., Kingston, J. H., McCollum, B., and Schaerf, A. (2013). The third international timetabling competition. *Annals of Operations Research*, pp. 1–7.

Raghavjee, R., and Pillay, N. (2008). An application of genetic algorithms to the school timetabling problem. In *Proceedings of the 2008 Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists on IT Research in Developing Countries: Riding the Wave of Technology*, pp. 193–199.

Raghavjee, R., and Pillay, N. (2012). A comparison of genetic algorithms and genetic programming in solving the school timetabling problem. In *Fourth World Congress on Nature and Biologically Inspired Computing*, pp. 98–103.

Smith, K. A., Abramson, D., and Duke, D. (2003). Hopfield neural networks for timetabling: Formulations, methods, and comparative results. *Computers and Industrial Engineering*, 44(2):283–305.

Sørensen, M., Kristiansen, S., and Stidsen, T. R. (2012). International timetabling competition 2011: An adaptive large neighborhood search algorithm. In *Proceedings of the 9th International Conference on the Practice and Theory of Automated Timetabling*, pp. 489–492.

Tassopoulos, I. X., and Beligiannis, G. N. (2012). A hybrid particle swarm optimization based algorithm for high school timetabling problems. *Applied Soft Computing*, 12(11):3472–3489.

Valouxis, C., and Housos, E. (2003). Constraint programming approach for school timetabling. *Computers and Operations Research*, 30(10):1555–1572.

Wilke, P., Gröbner, M., and Oster, N. (2002). A hybrid genetic algorithm for school timetabling. In B. McKay and J. Slaney (Eds.), *AI 2002: Advances in Artificial Intelligence*, volume 2557 of *Lecture Notes in Computer Science*, pp. 455–464.

Wilke, P., and Killer, H. (2010). Walk down jump up algorithm: A new hybrid algorithm for timetabling problems. In *Proceedings of the 8th International Conference on the Practice and Theory of Automated Timetabling*, pp. 440–446.

Wolpert, D. H., and Macready, W. G. (1997). No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1:67–82.

Wright, M. B. (1996). School timetabling using heuristic search. *Journal of the Operational Research Society*, 47(3):347–357.

Zhang, D., Liu, Y., M'Hallah, R., and Leung, S. C. H. (2010). A simulated annealing with a new neighborhood structure based algorithm for high school timetabling problems. *European Journal of Operational Research*, 203(3):550–558.