A Sequence-based Selection Hyper-heuristic Utilising a Hidden Markov Model

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

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

ABSTRACT

Selection hyper-heuristics are optimisation methods that operate at the level above traditional (meta-)heuristics. Their task is to evaluate low level heuristics and determine which of these to apply at a given point in the optimisation process. Traditionally this has been accomplished through the evaluation of individual or paired heuristics. In this work, we propose a hidden Markov model based method to analyse the performance of, and construct, longer sequences of low level heuristics to solve difficult problems. The proposed method is tested on the well known hyper-heuristic benchmark problems within the CHeSC 2011 competition and compared with a large number of algorithms in this domain. The empirical results show that the proposed hyper-heuristic is able to outperform the current best-in-class hyper-heuristic on these problems with minimal parameter tuning and so points the way to a new field of sequence-based selection hyper-heuristics.

Categories and Subject Descriptors

I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search—*Heuristic methods*

Keywords

Hyper-heuristic, Cross-domain, Computational Design

1. INTRODUCTION

A *hyper-heuristic* performs a search over the space of heuristics which operate directly on the space of solutions, for solving computationally hard problems [4]. It conducts exploration of the search space of heuristics complying with the limitations exposed by the *domain barrier*, which does not allow any problem dependent information to pass through to the high-level where the hyper-heuristic components reside. This feature enables the reuse of their components and themselves on other problem domains without any change. Even though the term hyper-heuristic was coined in the early 21st century [7], the idea of combining the different existing heuristics to exploit their strengths dates back to 1960s [8, 12]. Two categories of hyper-heuristics can be defined, namely, *selection* hyper-heuristics and *generation* hyper-heuristics. Selection hyper-heuristics function by selecting and applying a heuristic from a set of low level heuristics followed by a move acceptance criterion to decide whether to accept or reject the new solution (Figure 1), while generation hyper-heuristics aim to generate new heuristics by understanding the characteristics of the input heuristics. Hyper-heuristics can be further distinguished by their feedback mechanisms and they can incorporate online learning, offline learning or no learning at all. The online learning hyper-heuristic learns from feedback during the search process, whereas the offline learning hyper-heuristic learns before the actual search starts [4]. This work focuses on the development of a novel online learning selection hyper-heuristic utilising a hidden Markov model to analyse sequences of heuristics. The performance of the hyper-heuristic is assessed through HyFlex [19], a software tool for facilitating the testing of hyper-heuristics. HyFlex provides implementations of six problem domains each with its own problem instances and relevant problem-specific information such as low level heuristics, providing scientists with a means to benchmark their algorithms and therefore solely focus on the hyper-heuristic development. The cross domain heuristic search over the six HyFlex problem domains was the CHeSC competition in 2011[1].

Traditional selection hyper-heuristics consider single heuristics (e.g. simple random and random permutation hyper-heuristics [7]) or heuristic pair performance (e.g. choice function hyper-heuristic [7]) when determining the heuristic to select (and execute) next. Several previously proposed hyper-heuristics have attempted to produce sequences of heuristics. The sequences of heuristics in these studies are usually predetermined in an offline manner such as in iterated local search [15] which applies a sequence of mutational heuristics (diversification) followed by hill climbers (intensification). Before CHeSC 2011 started, it is reported in [3] that the best performing hyper-heuristic on HyFlex was an approach based on an iterated local search which applies a set of low level heuristics in a predefined sequence. The current state-of-the-art hyper-heuristic and the winner

GECCO '15, July 11–15, 2015, Madrid, Spain

 ISBN 978-1-4503-3472-3/15/07…$15.00
DOI: http://dx.doi.org/10.1145/2739480.2754766

[1] http://www.asap.cs.nott.ac.uk/chesc2011/

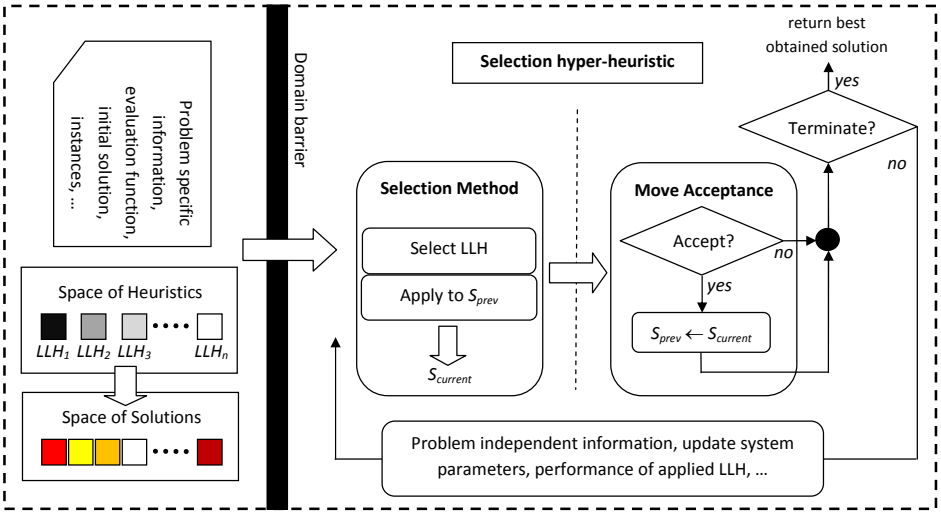

Figure 1: Selection hyper-heuristic framework

of CHeSC 2011, known as AdapHH [17], uses a technique to discover relay hybridised heuristics during part of the search process. The number of combined heuristics is fixed and maintained in a first-in-first-out manner. Other top performing approaches in CHeSC 2011 include the rank 2 algorithm VNS-TW [13], which is based on variable neighbourhood search and applies a predefined sequence of shaking heuristics then hill-climber heuristics; rank 3 was ML [14] which is based on a self-adaptive meta-heuristic using multiple cooperating agents to determine adequate sequences of local search heuristics; rank 4 was PHUNTER [6] which attempts to balance the *diversification* and *intensification* sequence of heuristics; and rank 5 was EPH [16] which evolves population of prefixed sequence of heuristics by employing *diversification* and *intensification* scheme. Soon after the competition, CHeSC 2011 became a benchmark for evaluating the performance and generality level of a selection hyper-heuristic for subsequent algorithms. These include [11], where an improved choice function hyper-heuristic is proposed, which uses heuristic pair performance. The results revealed the success of the approach when compared to the traditional choice function [7]; and [5, 1], where a set of successful iterated local search algorithms for cross-domain search are proposed.

In contrast to the above approaches, this work introduces a new method known as sequence-based selection hyper-heuristic (SSHH), inspired by the hidden Markov model (HMM) [2], at which each low level heuristic, analogous to a state in the HMM, has a transition probability matrix to move to another state and emission probability matrices to determine the parameters for each low level heuristic and the acceptance strategy. The HMM approach is able to both analyse the sequence of heuristics and generate sequences of any size for use in optimisation. The automatic generation of probability matrices linking low level heuristics, acceptance strategy and heuristic parameters ensures that many fewer parameters have to be set to operate the technique. The analysis of higher order (e.g. greater than 2) heuristic sequences in this way using a hidden Markov model is also unprecedented in the literature. Although a very recent technique using an HMM exists [21], it represents solutions as states rather than low level heuristics as proposed here. The proposed approach records the search process as a sequence of pairs of heuristic application in an online fashion, and processes this information to inform the selection of the next heuristic to apply in the optimisation. It does not explicitly enforce the *diversification* and *intensification* phases, but rather discovers and learns these sequences automatically. The HyFlex framework is used to evaluate this new hyper-heuristic and to compare with many of the state-of-the-art approaches shown above. In later experimentation we show the HMM-sequence based technique improves upon all the entries to this contest, demonstrating that it can improve on the state-of-the-art in this field, with many fewer parameters to optimise than competing techniques.

The paper is structured as follows. Section 2 provides briefly the background of HyFlex and CHeSC 2011. Section 3 describes the novelty and all algorithmic components of the proposed hyper-heuristic. Section 4 presents the results of the proposed method on HyFlex problems and discusses the generality of the approach. Section 5 provides the conclusions of the study.

2. HYFLEX

HyFlex (**Hy**per-heuristics **Flex**ible framework) [19] is an object oriented software framework written in Java providing an implementation of six problem domains and encapsulating the problem specific information such as solution representations, solution constructions, low level heuristics and evaluation functions. HyFlex currently provides an implementation of boolean satisfiability (SAT), 1D bin packing (BP), personnel scheduling (PS), permutation flowshop (PFS), travelling salesman problem (TSP), and vehicle routing problem (VRP). In each HyFlex problem, four types of perturbative low level heuristics are defined: (i) *mutational* which perturbs a solution randomly, (ii) *ruin and re-create* which destroys a given solution partially and then rebuilds the deleted parts, (iii) *hill climbing* that incorporates an iterative improvement process and returns a non-worsening solution, and (iv) *crossover* which creates a new solution by exchanging and recombining parts from two solutions. Each low level heuristic is associated with a problem and

heuristic dependent parameter, controlled by the user and taking a value in the range [0,1]. HyFlex was used in the Cross-domain Heuristic Search Challenge (CHeSC) in 2011, to determine the hyper-heuristic that works well across the different HyFlex problem domains. Competitors are invited to submit their hyper-heuristics to the HyFlex framework for evaluation. The organisers conducted 31 runs for each competing algorithm for five selected instances from each problem domain. Each run is limited to 600 seconds based on the organisers' machines. The methodology followed to rank the algorithms was inspired by the Formula One system. In each instance, the median values of the 31 runs of each algorithm are calculated, and the top eight algorithms based on the median values take the score 10, 8, 6, 5, 4, 3, 2 and 1 points, respectively. These points are summed across the 30 instances for each algorithm. The approach that scores the maximum is deemed the winner. Table 1 summarises the total number of low level heuristics and the indexes of each low level heuristic for each heuristic category. Twenty competitors submitted their algorithms to CHeSC 2011. The description of the twenty competing algorithms and the results are provided in the CHeSC 2011 competition website.

Table 1: The total number of low level heuristics and the distribution of heuristic types

problem domains	SAT	BP	PS	PFS	TSP	VRP
no. of heuristics	11	8	12	15	13	10
mutational	0-5	0,3,5	11	0-4	0-4	0,1,7
ruin & re-create	6	1,2	5-7	5,6	5	2,3
hill climbing	7,8	4,6	0-4	7-10	6-8	4,8,9
crossover	9,10	7	8-10	11-14	9-12	5,6

3. METHODOLOGY

The traditional iterative selection hyper-heuristic passes a solution through a heuristic selection process to decide on a heuristic to apply from a fixed set of low level heuristics and then a move acceptance process to accept or reject the newly created solution at each step. The selection process has traditionally been accomplished through the evaluation of individual or paired heuristics. Empirical evidence [20] shows that the traditional selection hyper-heuristic framework may lead to the mutational heuristics being ignored. Hence, several hyper-heuristic frameworks have been proposed to combine a set of mutational and hill climber heuristics by invoking a mutational heuristic first followed by a hill climber [20]. These frameworks are explicitly encouraging a diversifying (exploring of the search space) phase followed by an intensifying (exploiting of the accumulated search experience) phase. Here we propose a sequence-based selection hyper-heuristic utilising a hidden Markov model which is relatively simple to implement and achieves robustness by automatically adjusting itself to the problem domain. The method is also able to use its probability matrices to adapt to changing search characteristics over time. For example, a problem might be best solved in the initial stages by a certain sequence of heuristics selected from the heuristics pool, but as the nature of the optimisation problem changes over time, the sequence will also need to change, necessitating the online approach. The developed method differs markedly from standard selection hyper-heuristic approaches in that it is able to *observe* recent sequences of operations and so is able to learn sequences of heuristics that generate good solutions. The algorithm 1 provides the pseudocode of the proposed hyper-heuristic.

In this method, the sequences of low level heuristics are generated and analysed using hidden Markov model (HMM) [2]. The hidden states of an HMM correspond to the low level heuristics. Initially, each low level heuristic (state) has an initial probability to be selected (Algorithm 1, Line 1). Each low level heuristic has three probability matrices associated with it: An acceptance strategy emission probability matrix to associate heuristics with acceptance strategy, a heuristic parameter emission probability matrix to determine the parameters for each low level heuristic and a transition probability matrix to determine the probability of moving from one heuristic to another during the optimisation (Algorithm 1, Lines 2-4). Initially, all probabilities are distributed equally (Algorithm 1, Line 8). At the first step, an initial low level heuristic will be selected (Algorithm 1, Line 12).

The next low level heuristic, acceptance strategy (AS) and heuristic parameter (p) are chosen based on probabilities provided in the matrices (Algorithm 1, Lines 15-17), using a roulette wheel selection strategy. They are added to a record for later processing (Algorithm 1, Line 18). Recall that each low level heuristic in HyFlex is associated with a problem and heuristic dependent parameter (denoted as p), controlled by the user and taking a value in the range [0,1]. We discretised the choices for the control parameters provided in HyFlex into 11 different parameters: {0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0}, whereas for the acceptance strategy there are only two options: {1, 2}. The selected heuristic with the selected parameter will be applied to the candidate solution to produce a new solution (Algorithm 1, Line 19). If the acceptance strategy is 1, then the move acceptance method will be invoked and the emission and transition probability matrices are updated (Algorithm 1, Lines 20-28). Otherwise, the move acceptance method will not be applied meaning that the next low level heuristic will be applied on the new solution regardless of the quality of the new solution. The latter strategy helps the search to discover new regions in the search space. The move acceptance method uses the previous solution before applying the recorded recent sequence of heuristics (i.e. the solution which was returned the last time the move acceptance method was called) and the current (candidate) solution and then returns one of these two solutions (Algorithm 1, Line 21). To maximise the likelihood of generating best solutions, emission and transition probability matrices are updated only if the candidate solution is better than the best recorded solution (Algorithm 1, Lines 23-26). To update the matrices (Algorithm 1, Line 25), the scores of the three matrices corresponding to the recorded previous and next heuristics, acceptance strategies and heuristic parameters are increased by 1. The probability of selection in any of the matrices is $score_p / \sum_{\forall k}(score_k)$. The above methods are iteratively applied until a set of termination criteria is satisfied (Algorithm 1, Line 29).

Figure 2 illustrates an example of how the developed method would work on 3 low level heuristics. For simplicity, we consider only three possible heuristic parameters: {0.0, 0.1,

Algorithm 1: Sequence-based Selection Hyper-heuristic

```
Let I: initial state distribution;
Let A: state transition probabilities;
Let B: acceptance strategy emission probabilities;
Let C: heuristic parameter emission probabilities;
Let S_prev: previous solution;
Let S_curr: current solution;
Let S_best: best obtained solution;
I, A, B, C ← Initialise();
S_prev ← S_initial;
S_curr ← S_prev;
S_best ← S_prev;
llh_curr ← Select(I);
repeat
    llh_prev ← llh_curr;
    llh_curr ← Select(A, llh_prev);
    AS ← Select(B, llh_curr);
    p ← Select(C, llh_curr);
    record.Add(llh_prev, llh_curr, AS, p);
    S_curr ← Apply(S_curr, llh_curr, p);
    if AS == 1 then
        S_prev ← MoveAcceptance(S_prev, S_curr);
        S_curr ← S_prev;
        if S_curr isBetterThan S_best then
            S_best ← S_curr;
            UpdateScores(A, B, C, record);
        end
        record.Clear();
    end
until TerminationCriteriaSatisfied();
```

0.2}. The initial matrices and the updated matrices after 5 iterations are shown in the figure. At the first iteration, LLH2 is selected with selection parameter $p = 0.2$ and acceptance strategy $AS = 1$ and applied on an initial solution ($S_{initial}$). The newly generated solution (S_1) is rejected by the move acceptance method and therefore the next low level heuristic will be applied on $S_{initial}$. The matrices are not updated because the best obtained solution has not been improved. LLH1 with $p = 0.1$ and $AS = 1$ are selected next and applied on $S_{initial}$. The new solution (S_2) is accepted by the move acceptance method and its quality is better than the best obtained solution, and therefore, the scores of moving from LLH2 to LLH1, the heuristic parameter $p = 0.1$ and the acceptance strategy $AS = 1$ of LLH1 are increased by 1. LLH3 with $p = 0.0$ and $AS = 2$ are selected next and applied on S_2 generating new solution S_3. Nothing is updated and the move acceptance method is not invoked because $AS = 2$. We move to the next iteration where LLH1 with $p = 0.1$ and $AS = 1$ are selected and applied on S_3 generating new solution S_4. The move acceptance method compares S_4 with S_2 to decide whether to accept or reject S_4. In this example, S_4 is accepted and because its quality is better than the best obtained solution (which was S_2) the scores of moving from LLH1 to LLH3, the heuristic parameter $p = 0.0$ and the acceptance strategy $AS = 2$ of LLH3 are increased by 1; and also the scores of moving from LLH3 to LLH1, the heuristic parameter $p = 0.1$ and the acceptance strategy $AS = 1$ of LLH1 are increased by 1. LLH1 with $p = 0.2$ and $AS = 1$ are selected next and applied on S_4. The new solution (S_5) is accepted by the move acceptance method, but its quality is not better than the best obtained solution, hence, the matrices are not updated.

When trained for a sufficient number of iterations on the problem, those sequences of operations that are more likely to produce the best solutions will automatically be learned. The two emissions and transition probability matrices create a model of the optimisation process conducted thus far and the likelihood for of the optimisation being in state (low level heuristic) i at time n given the recent application of heuristics, heuristic parameters and acceptance strategies can be calculated thus:

$$L(i, n) = L(i, n-1) \times a_{llh_{n-1}.llh_n} \times e^1(p) \times e^2(AS) \quad (1)$$

where $L(i, n-1)$ is the likelihood of the previous state (low level heuristic) in the automaton, a is the probability of transition from one low level heuristic to another, e^1 is the probability of the emission of heuristic parameter p for llh_n, and e^2 is the probability of the emission of acceptance strategy AS for llh_n. Given the example in Figure 2 and according to the roulette wheel selection strategy, one can say that LLH3 with $AS = 2$ and $p = 0.0$ has the greatest probability to be selected next, followed by LLH1 with $AS = 1$ and $p = 0.1$.

4. EMPIRICAL RESULTS

In this section, the results of the proposed sequence-based selection hyper-heuristic approach, denoted as SSHH, are presented. There is still a debate going on the usefulness of crossover in the evolutionary algorithm community [10, 18]. Considering that we have proposed a single point based search framework, crossover operators provided in HyFlex are ignored by our method during the experiments, since crossover requires two solutions as input. Investigating how to best utilise all low level heuristics, including crossover operators within sequence-based selection hyper-heuristic would be of interest as future work. The experiments are performed on an i7-4770K CPU at 3.50GHz with 16GB RAM. A benchmarking software tool developed by the organisers of CHeSC is used to report the time our machine should take which is equivalent to 600 seconds of the organisers' machine. This program reported that a single run of our machine should terminate after 346 seconds which was implemented as the convergence criterion. The move acceptance criterion (only called when $AS = 1$) employed in this work accepts all improving moves by default; non-improving moves are accepted if the objective value of the candidate solution is less than a threshold. The threshold equals the objective function value of the best recorded solution plus a value (T). The value of T is set to 30, and is decreased by 2 in case no improvements in the current best obtained solution are found for 15 consecutive seconds. Note that if the integer part of the objective value is already 0, then T is added to the decimal part. For example, if the objective value is 0.0325 and $T = 30$ then the threshold would be 0.0355. The parameter setting for T of 30 has been set after a small number of preliminary experiments.

To compare the performance of the SSHH against the CHeSC 2011 competitors, 31 trials across five CHeSC instances from boolean satisfiability (SAT), one dimensional bin packing (BP), personnel scheduling (PS), permutation flow shop (PFS), travelling salesman problem (TSP) and vehicle routing problem (VRP) problem domains are performed. Table 2 provides the scores of SSHH and CHeSC 2011 competing algorithms based on the Formula One scoring system with respect to the median values across all CHeSC 2011 instances. The proposed SSHH ranks top with

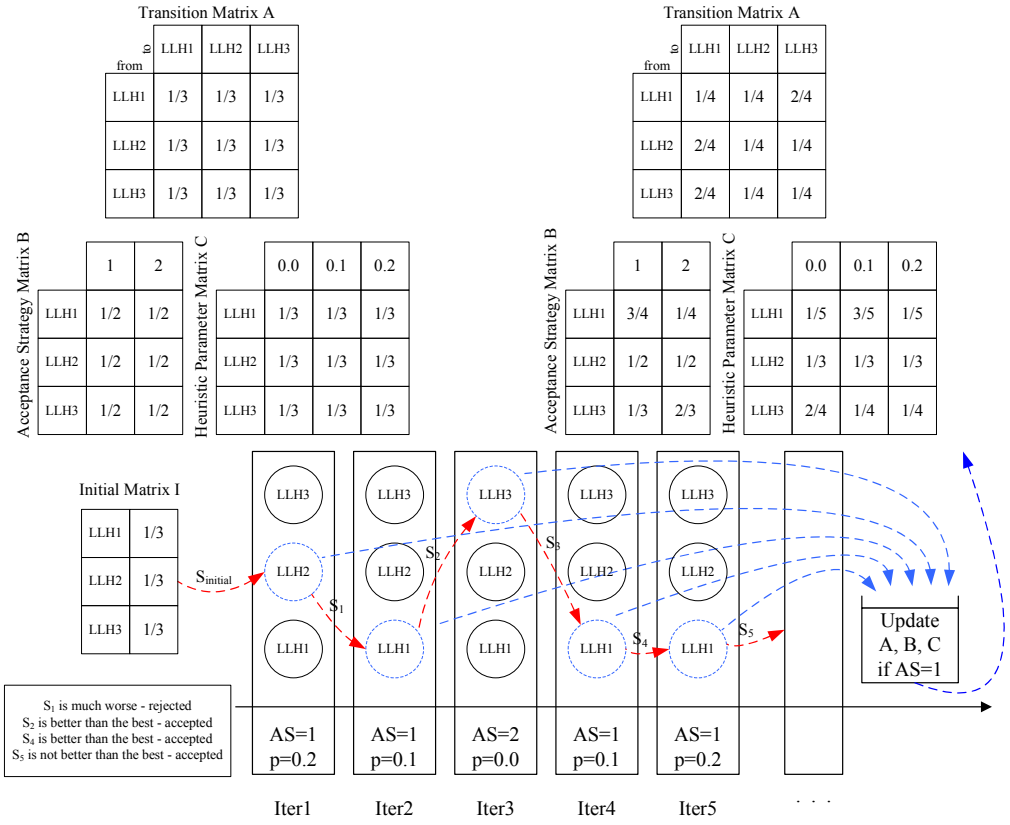

Figure 2: Sequence-based Selection Hyper-heuristic

a total score of 165.10, beating all other entries to the competition including the current best performer, AdapHH. SSHH is the best in SAT, BP and TSP problem domains. In the PS and VRP problem domains, it still performs well compared to the CHeSC approaches, but in the PFS problem domain, its performance is not as good as on the other problem domains. Interestingly, by examining the competing methods, the proposed SSHH performs better than the Markov chain hyper-heuristic (MCHH-S), that uses the Markov property, on all HyFlex problem domains. Further analysis of the behaviour of SSHH can be seen in Figure 4.

Figure 3 presents the boxplots of the normalised median objective function values of each competing method and SSHH for each instance for SAT, BP, PS, PFS, TSP and VRP problem domains. The median objective function values are normalised to a value in the range [0,1] as suggested in [9]. The normalisation is used to unify the scales of the objective function values. The following formula is employed for the normalisation:

$$n = \frac{m(i) - m_b(i)}{m_w(i) - m_b(i)} \quad (2)$$

where $m(i)$, for a given method, is the median objective function value of 31 runs on instance i, $m_b(i)$ is the best median objective function value of 31 runs obtained by any method on instance i, and $m_w(i)$ is similar but for the worst.

Due to space restrictions, we only provide analysis for three problem domains: BP, PS and TSP. Figure 4 provides the average utilisation rate of each heuristic considering only moves which improve on the best-of-run solution from 10 runs while solving an arbitrary instance from each problem domain. The figure also provides the probabilities of the emissions and transitions for each low level heuristic. Some of the low level heuristics do not seem to contribute to the best-of-run solutions. For example, LLH11 in PS does not contribute to the improvement of the best solutions. Fur-

Table 2: Scores of SSHH and CHeSC 2011 competing algorithms across six domains

Method	SAT	BP	PS	PFS	TSP	VRP	Overall
SSHH	39.10	45.00	22.50	3.50	41.00	14.00	165.10
AdapHH	30.43	38.00	8.00	37.00	34.75	14.00	162.18
VNS-TW	30.93	2.00	35.50	33.50	12.75	5.00	119.68
ML	10.00	8.00	29.50	39.00	10.00	21.00	117.50
PHUNTER	7.00	2.00	11.50	8.00	22.75	32.00	83.25
EPH	0.00	6.00	9.00	21.00	29.75	11.00	76.75
HAHA	26.43	0.00	23.00	3.50	0.00	13.00	65.93
NAHH	10.50	15.00	1.00	22.00	10.00	6.00	64.50
KSATS-HH	20.35	9.00	7.00	0.00	0.00	21.00	57.35
ISEA	3.50	23.00	14.50	3.50	7.00	3.00	54.50
HAEA	0.00	1.00	1.00	8.00	8.00	25.00	43.00
ACO-HH	0.00	16.00	0.00	9.00	7.00	1.00	33.00
GenHive	0.00	10.00	6.50	7.00	2.00	6.00	31.50
SA-ILS	0.25	0.00	16.00	0.00	0.00	4.00	20.25
XCJ	3.50	11.00	0.00	0.00	0.00	5.00	19.50
AVEG-Nep	9.50	0.00	0.00	0.00	0.00	8.00	17.50
DynILS	0.00	9.00	0.00	0.00	8.00	0.00	17.00
GISS	0.25	0.00	8.00	0.00	0.00	6.00	14.25
SelfSearch	0.00	0.00	2.00	0.00	2.00	0.00	4.00
MCHH-S	3.25	0.00	0.00	0.00	0.00	0.00	3.25
Ant-Q	0.00	0.00	0.00	0.00	0.00	0.00	0.00

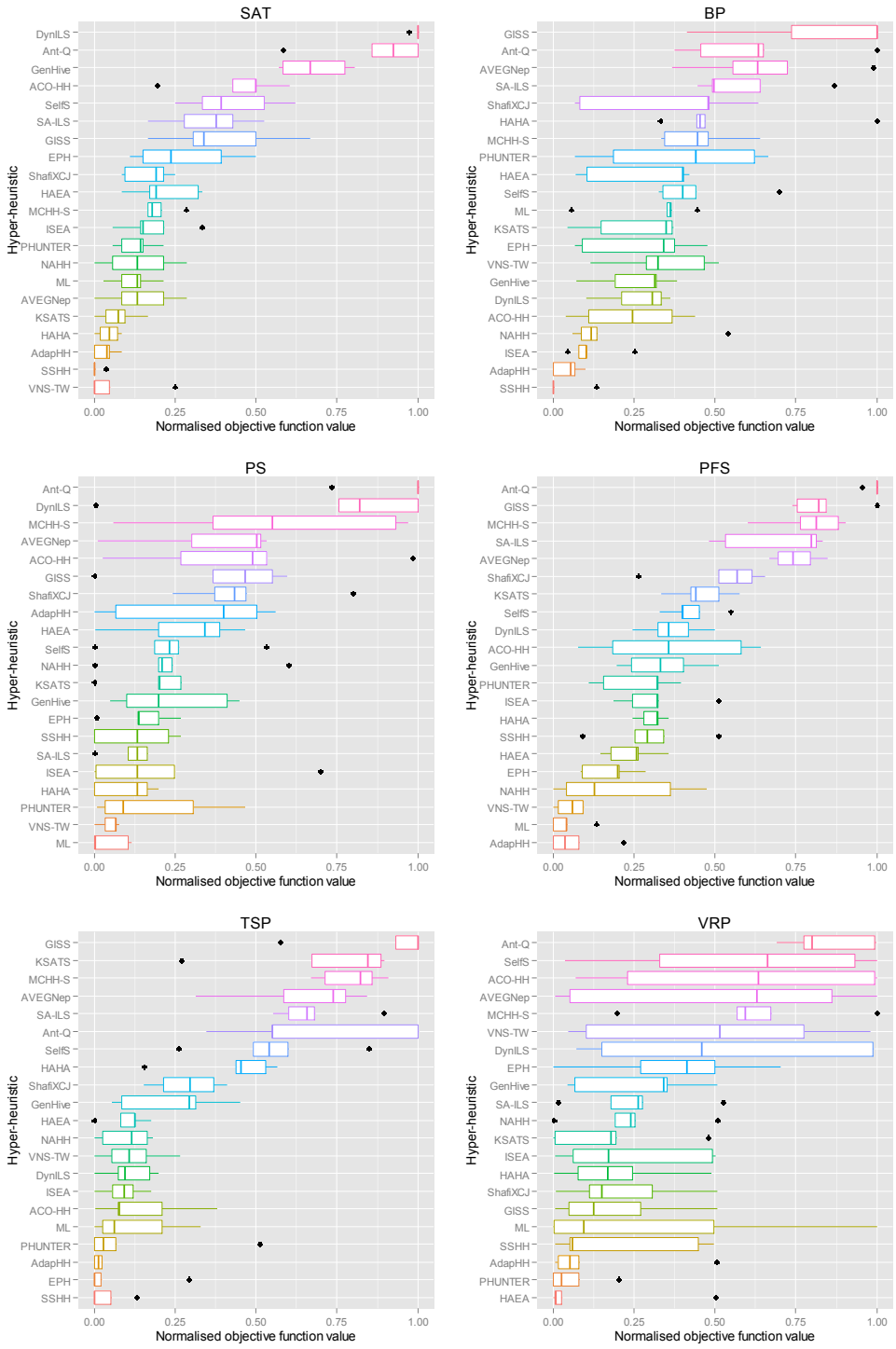

Figure 3: Boxplots of the normalised median objective function values of SSHH and CHeSC 2011 hyper-heuristics per each HyFlex problem domain over five CHeSC instances

thermore, a number of the low level heuristics are only useful when combined with other sequence of heuristics. For example, LLH0, LLH1 and LLH5 in TSP do not make improvement in their own, but the majority of the improvements in TSP are due to combining these heuristics (mutational and ruin & re-create heuristics) with LLH8 (hill climber). Interestingly, and by examining the transition matrix, LLH8 does not improve the best-of-run solutions unless combined with these heuristics. These types of sequences, which are automatically discovered by the proposed approach, have been previously suggested in the literature [15] by applying perturbation first (e.g. mutational heuristics) to explore the search space followed by the intensification phase (e.g. hill climbers) to exploit the accumulated search experience, but

until now, this has been done manually. In BP, the sequence LLH4-LLH6-LLH2-LLH6 seems to generate most of the improvements, an interesting finding. However, in the case of the PS problem domain, such heuristic sequences were rarely identified. This is likely to be due to the fact that the low level heuristics (in particular the hill-climbers) associated with the PS problem domain are slow, and therefore there was not enough time to learn and detect the good sequences. In general, the probabilities of the heuristic parameters are distributed equally. However, in BP large values are slightly preferable. Most of the low level heuristics in BP use acceptance strategy 2 meaning that the search is largely exploratory for this problem. For the TSP problem domain, it is observed that the most successful low level heuristic in this problem (LLH8 - hill climber) favours the first acceptance strategy and acts as an intensification process, while the other heuristics (particularly LLH0, LLH1 and LLH5) prefer the second acceptance strategy for the purposes of diversification.

5. CONCLUSION

In this paper a new sequence-based selection hyper-heuristic inspired by a hidden Markov model has been proposed. The method has been shown to successfully optimise a number of combinatorial optimisation problems. The proposed approach is the best general purpose hyper-heuristic for heuristic search across a number of different HyFlex problem domains, achieving improved performance over established state-of-the-art hyper-heuristics such as AdapHH. A key further point is that the SSHH approach requires only one parameter (the threshold in the move acceptance method) to be set as opposed to AdapHH that has over 45 parameters[2]. The adaptive iteration limited list-based threshold move accepting method used in AdapHH allows diversification only if the intensification phase does not yield any improvements for a predefined number of iterations. Our method discovers and learns automatically when to move from intensification to diversification and vice versa. The additional power of this approach comes from the fact that it is capable of discovering sequences of heuristics and automatically identifying the relationship between the heuristic selected, its heuristic parameter and acceptance strategy in addition to its relationship with other low level heuristics on the problem. One further advantage of the method is that the probabilities learned by the model are easily accessible and can be analysed to determine what has been learned about the search space and relationships between low level heuristics, heuristic parameters and acceptance strategy. Our initial observations showed that the performance of selection hyper-heuristics may vary depending on the choice of the low level heuristics, and that not all the heuristics contribute to the improvement of a candidate solution during part of the search process unless they are applied in combination with sequences of heuristics. The proposed approach begins from a naïve state where the nature of the low level heuristics is not known, yet its adaptive characteristics lead automatically to the discovery of useful sequences of mutational heuristics followed by hill climbers. These findings are supported by results from other studies [15]. Further work on this analysis will investigate how these probabilities change over time and their interaction with known features in the problem landscape.

[2]http://code.google.com/p/generic-intelligent-hyper-heuristic/

6. ACKNOWLEDGMENTS

This work was supported by EPSRC grant EP/K000519/1.

7. REFERENCES

[1] S. Adriaensen, T. Brys, and A. Nowé. Fair-share ILS: a simple state-of-the-art iterated local search hyperheuristic. In D. V. Arnold, editor, *Proceedings of the 2014 Conference on Genetic and Evolutionary Computation (GECCO '14)*, pages 1303–1310, New York, NY, USA, 2014. ACM.

[2] L. E. Baum and T. Petrie. Statistical inference for probabilistic functions of finite state Markov chains. *The Annals of Mathematical Statistics*, 37(6):1554–1563, 12 1966.

[3] E. K. Burke, T. Curtois, M. R. Hyde, G. Kendall, G. Ochoa, S. Petrovic, J. A. V. Rodríguez, and M. Gendreau. Iterated local search vs. hyper-heuristics: towards general-purpose search algorithms. In *IEEE Congress on Evolutionary Computation (CEC '10)*, pages 1–8, 2010.

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

[5] E. K. Burke, M. Gendreau, G. Ochoa, and J. D. Walker. Adaptive iterated local search for cross-domain optimisation. In *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation (GECCO '11)*, pages 1987–1994, New York, NY, USA, 2011. ACM.

[6] C. Y. Chan, F. Xue, W. H. Ip, and C. F. Cheung. A hyper-heuristic inspired by pearl hunting. In Y. Hamadi and M. Schoenauer, editors, *Learning and Intelligent Optimization*, Lecture Notes in Computer Science, pages 349–353. Springer Berlin Heidelberg, 2012.

[7] P. Cowling, G. Kendall, and E. Soubeiga. A hyperheuristic approach to scheduling a sales summit. In E. Burke and W. Erben, editors, *Practice and Theory of Automated Timetabling III*, volume 2079 of *Lecture Notes in Computer Science*, pages 176–190. Springer Berlin Heidelberg, 2001.

[8] W. B. Crowston, F. Glover, G. L. Thompson, and J. D. Trawick. *Probabilistic and parametric learning combinations of local job shop scheduling rules*. 117. Defense Technical Information Center, 1963.

[9] L. Di Gaspero and T. Urli. Evaluation of a family of reinforcement learning cross-domain optimization heuristics. In Y. Hamadi and M. Schoenauer, editors, *Learning and Intelligent Optimization*, Lecture Notes in Computer Science, pages 384–389. Springer Berlin Heidelberg, 2012.

[10] B. Doerr, E. Happ, and C. Klein. Crossover can provably be useful in evolutionary computation. In *Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation (GECCO '08)*, pages 539–546, New York, NY, USA, 2008. ACM.

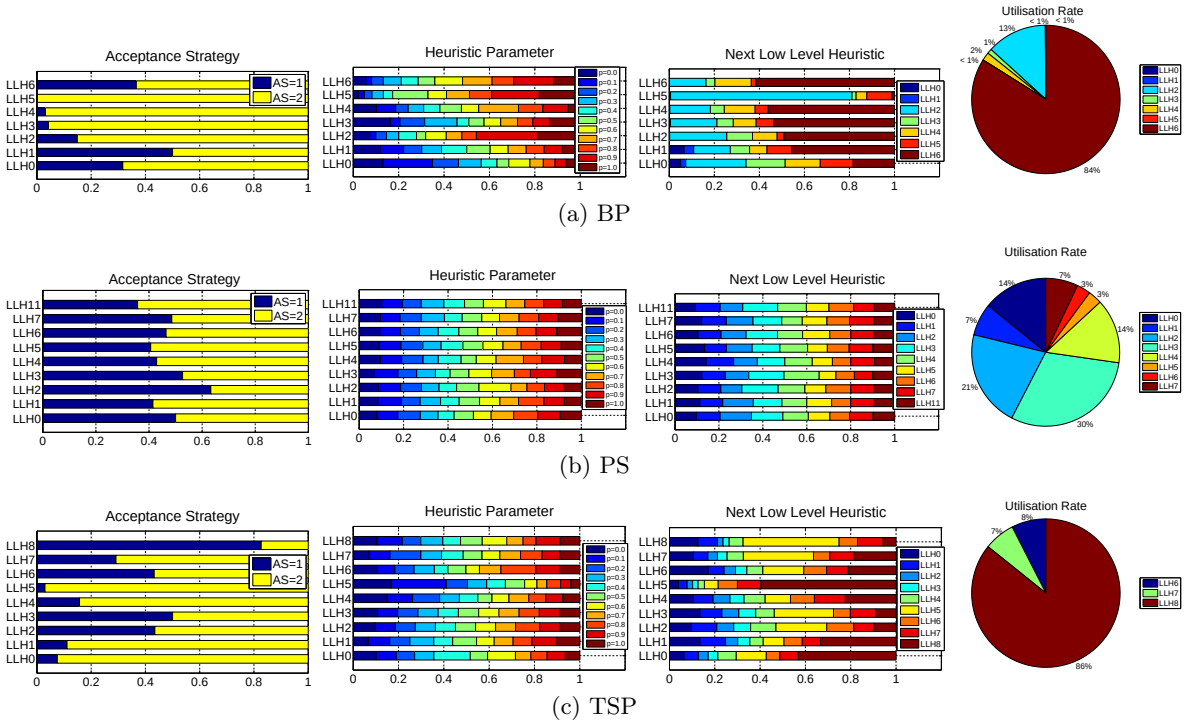

Figure 4: Mean emission and transition probabilities and utilisation rate of each low level heuristic considering only moves which improve on the best-of-run solution from 10 runs while solving an arbitrary instance from each problem domain

[11] J. H. Drake, E. Özcan, and E. K. Burke. An improved choice function heuristic selection for cross domain heuristic search. In C. A. C. Coello, V. Cutello, K. Deb, S. Forrest, G. Nicosia, and M. Pavone, editors, *Parallel Problem Solving From Nature (PPSN XII)*, volume 7492 of *Lecture Notes in Computer Science*, pages 307–316. Springer Berlin Heidelberg, 2012.

[12] H. Fisher and G. L. Thompson. Probabilistic learning combinations of local job-shop scheduling rules. In J. F. Muth and G. L. Thompson, editors, *Industrial Scheduling*, pages 225–251, New Jersey, 1963. Prentice-Hall, Inc.

[13] P.-C. Hsiao, T.-C. Chiang, and L.-C. Fu. A VNS-based hyper-heuristic with adaptive computational budget of local search. In *IEEE Congress on Evolutionary Computation (CEC '12)*, pages 1–8, 2012.

[14] M. Larose. A hyper-heuristic for the CHeSC 2011. In *The 53rd Annual Conference of the UK Operational Research Society (OR53)*, 2011.

[15] H. R. Lourenço, O. C. Martin, and T. Stützle. Iterated local search: framework and applications. In M. Gendreau and J.-Y. Potvin, editors, *Handbook of Metaheuristics*, volume 146 of *International Series in Operations Research and Management Science*, pages 363–397. Springer US, 2010.

[16] D. Meignan. An evolutionary programming hyper-heuristic with co-evolution for CHeSĆ11. In *The 53rd Annual Conference of the UK Operational Research Society (OR53)*, 2011.

[17] M. Misir, K. Verbeeck, P. De Causmaecker, and G. Vanden Berghe. A new hyper-heuristic implementation in HyFlex: a study on generality. In J. Fowler, G. Kendall, and B. McCollum, editors, *Proceedings of the 5th Multidisciplinary International Scheduling Conference: Theory and Application (MISTA2011)*, pages 374–393, 2011.

[18] M. Mitchell, J. H. Holland, and S. Forrest. When will a genetic algorithm outperform hill climbing. In J. D. Cowan, G. Tesauro, and J. Alspector, editors, *Advances in Neural Information Processing Systems 6*, pages 51–58. Morgan Kaufmann, 1994.

[19] G. Ochoa, M. Hyde, T. Curtois, J. A. Vazquez-Rodriguez, J. Walker, M. Gendreau, G. Kendall, B. McCollum, A. J. Parkes, S. Petrovic, and E. K. Burke. HyFlex: a benchmark framework for cross-domain heuristic search. In J.-K. Hao and M. Middendorf, editors, *Evolutionary Computation in Combinatorial Optimization*, volume 7245 of *Lecture Notes in Computer Science*, pages 136–147. Springer Berlin Heidelberg, 2012.

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

[21] W. Van Onsem, B. Demoen, and P. De Causmaecker. HHaaHMM: a hyper-heuristic as a hidden Markov model. http://www.hyflex.org/chesc2014/, 2014.