Adaptive Thompson Sampling for Hyper-heuristics

Fawaz Alanazi
School of Computer Science
University of Nottingham
Nottingham, United Kingdom
Email: psxfa1@nottingham.ac.uk

***Abstract*—There is an interest in search algorithms capable of learning and adapting their behaviour while solving a given problem. A hyper-heuristic operates on a set of predefined heuristics and applies a machine learning technique to predict which heuristic is the most effective to apply at a given point in time. Thompson Sampling is a machine learning mechanism interacting with the search environment to adapt its behaviour through trial-and-error. Despite the fact that it originated in the 1930s, the work on Thompson Sampling in the literature on search heuristics is limited. This paper is the first study investigating the Thompson Sampling approach in the field of hyper-heuristics. I propose an adaptive Thompson Sampling mechanism for hyper-heuristics and extensively evaluate its performance on a wide range of test models and combinatorial optimisation problems. The proposed algorithm is tested and compared with a large number of hyper-heuristics within a well-known competition for hyper-heuristics called *CHeSC 2011*. The results reveal that the proposed hyper-heuristic outperforms all the competing hyper-heuristics, including the state-of-the-art algorithm, on three combinatorial optimisation problems: (1) Personnel Scheduling; (2) Permutation Flow-shop, and (3) the Travelling Salesman problem.**

1. Introduction

Hyper-heuristics are general-purpose search algorithms for either selecting heuristics from a set of predefined heuristics or generating heuristics based on a set of pre-existing heuristic components to solve optimisation problems [1]. The former is termed *selection hyper-heuristics*, and the latter *generation hyper-heuristics*. The term 'hyper-heuristic' was recently introduced by Cowling et al. [2]. However, the idea of combining multiple heuristics online originated in the 1960*s* [3]. The primary goal of a class of hyper-heuristics is to provide solutions of acceptable qualities on various problems rather than a single one, by selecting or generating heuristics that, in turn, operate on the search space of a given problem. This work focuses on the selection type of hyper-heuristics. A generic selection hyper-heuristic is a single-point based search algorithm consists of two main strategies: (1) *heuristic selection* and (2) *move-acceptance* [4]. The heuristic selection strategy chooses a heuristic from a pool of heuristics and applies to the current solution. The move acceptance strategy decides whether to accept or discard the newly generated solution.

Online-learning selection hyper-heuristics operate on a set of heuristics, applying a machine learning technique to predict, while solving a given problem, which heuristic is the most effective to apply at a given point in time [1]. In general, heuristics perform differently between different regions of the search space of a given problem (see e.g. [4] [5]). A challenge for online-learning hyper-heuristics is to balance between exploring the effectiveness of the heuristics on the current search point, and exploiting the most promising heuristics based on previous observations. This is termed the 'exploration versus exploitation' (EvE) trade-off. Thompson Sampling was introduced by W. R. Thompson in 1933, and forms a learning mechanism for addressing the EvE trade-off on general models so-called *stationary* multi-armed bandit problems [6]. It is based on the concept of probability matching: each action is randomly chosen with probability consistent with its probability of being optimal. Despite the fact that it was absent from the artificial intelligence literature, there has recently been renewed interest in Thompson Sampling, and strong theoretical results have emerged that justify its use [7] [8] [9]. Kaufmann *et al* [10] proved that Thompson Sampling is asymptotically optimal when applied to the so-called Bernoulli multi-armed bandit problem. Its effectiveness on more complex reward models was demonstrated theoretically in [9]. The idea of Thompson Sampling has also been successfully applied to several real-world problems in industry (see e.g. [11] [12]).

To the best of the author's knowledge, this is the first study to investigate Thompson Sampling in the field of hyper-heuristics. I propose an adaptive Thompson Sampling mechanism for hyper-heuristics to solve difficult optimisation problems. The proposed hyper-heuristic is evaluated on a set of well-known combinatorial optimisation problems using HyFlex [13]. HyFlex is a multi-domain framework that provides implementations of six combinatorial optimisation problems each associated with a set of problem instances, including: (1) Boolean satisfiability; (2) bin packing; (3) personnel scheduling; (4) permutation flow-shop; (5) travelling salesman problem; and (6) the vehicle routing problem. The hyper-heuristic is compared to a large number of hyper-heuristics on the HyFlex problems within a well-known hyper-heuristic competition called Cross-domain Heuristic

Search Challenge (CHeSC) 2011 [13]. When it comes to the CHeSC 2011 competition's highest-ranking hyper-heuristics, the winning hyper-heuristic, known as *AdapHH*, was based on a dynamic relay technique, which hybridised the best-performing heuristics. The second ranking hyper-heuristic was the *VNS-HH* (i.e. variable neighbourhood search based hyper-heuristic), which applied a predefined sequence of mutation and local search heuristics. The *ML* hyper-heuristic was ranked third, and applied a mutation heuristic, followed by a set of local search heuristics (the description of the remaining competing hyper-heuristics can be found on the competition website[1]). Since the competition, a number of studies have employed the CHeSC 2011 as a benchmark to evaluate their algorithms. Burke et al. [14] assessed the performance of two adaptive iterated local search hyper-heuristics using HyFlex. Drake et al. [15] proposed a variant of the *choice function* hyper-heuristic [2], applying it to HyFlex problems. Kheiri and Keedwell [16] tested a hyper-heuristic utilising a hidden Markov chain model using the HyFlex framework.

This paper is organised as follows. Section 2 introduces the proposed Thompson Sampling hyper-heuristic. Section 3 presents the benchmark problems. Section 5 shows the experimental results of the proposed hyper-heuristic on both the HyFlex problems and a set of test models. Section 6 draws the conclusions from this work.

2. Adaptive Thompson Sampling Hyper-heuristic

A generic hyper-heuristic framework consists of a finite search space $\mathscr{X}$, an objective function $f : \mathscr{X} \to \mathbb{R}$, a set $H := \{h_1, \ldots, h_m\}$ of m low-level heuristics, and two high-level strategies (i.e. heuristic selection and move-acceptance). The proposed hyper-heuristic generates an initial solution $x \in \mathscr{X}$ which is evolved by selecting and applying heuristics from a prefixed set of low-level heuristics (variation operators). The newly generated solution is then evaluated, and a decision is made whether it should be accepted or rejected. A simple move-acceptance strategy is employed which accepts the new solution if (and only if) it is not worse than the current candidate solution.

The proposed hyper-heuristic applies a Thompson Sampling mechanism to identify which heuristic should be applied to a given search point. Thompson Sampling is a Bayesian selection strategy that chooses a heuristic with a probability matching its probability of being optimal based on all past observations [6]. Thus, a heuristic is considered to be optimal based on a Bayesian estimate. Typically, low-level heuristics perform differently as the optimiser moves between different regions of the search space of a given problem. The proposed Thompson Sampling strategy is therefore associated with a sliding-time-window, which only maintains the recent past results of low-level heuristics. This enables past, and potentially irrelevant, information about the performance of low-level heuristics to be discarded. That, in turn, signifies the importance of recent observations concerning the estimation of the effectiveness of low-level heuristics with respect to the current search point. A formal presentation of the proposed hyper-heuristic in the form of a pseudo-code is given in Algorithm 1.

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

Algorithm 1 Thompson Sampling Hyper-heuristic (TSHH)

```
1:  Input: a finite set $\mathscr{X}$, an objective function $f : \mathscr{X} \to \mathbb{R}$,
    an integer $w \in \mathbb{N}^+$ {size of the sliding-time-window},
2:  and a set $H := \{h_1, \ldots, h_m\}$ of $m$ mutational and local
    search low-level heuristics, where $\forall\ i \in \{1, \ldots, m\}$, $h_i : \mathscr{X} \to \mathscr{X}$.
3:  Let $MU \subset H$ be a subset of mutational low-level heuristics, and $LS \subset H$ be a subset of local search low-level heuristics.
4:  Set $\alpha_i^{(0)} := 1$, and $\beta_i^{(0)} := 1\ \forall\ i \in \{1, \ldots, m\}$.
5:  Choose an initial solution $x \in \mathscr{X}$.
6:  for $t = 0, 1, \ldots$ until the stopping condition is satisfied do
7:      Sample $U_i^{(t)} \sim Beta(\alpha_i^{(t)}, \beta_i^{(t)})\ \forall\ i \in \{1, \ldots, m\}$
8:      Pick $h_{mu} \in MU$ that has $U_{mu}^{(t)} = \arg\max_{i \in MU}\{U_i^{(t)}\}$
9:      $x' := h_{mu}(x)$ {apply the chosen heuristic}
10:     Pick $h_{ls} \in LS$ that has $U_{ls}^{(t)} = \arg\max_{j \in LS}\{U_j^{(t)}\}$
11:     $x'' := h_{ls}(x')$
12:     $\eta_i^{(t)} = \begin{cases} 1 & \text{if heuristic } i \text{ is chosen and } f(x) > f(x'') \\ 0 & \text{otherwise} \end{cases}$
13:     $\ell_i^{(t)} = \begin{cases} 1 & \text{if heuristic } i \text{ is chosen and } f(x) \leq f(x'') \\ 0 & \text{otherwise} \end{cases}$
14:     Set $\alpha_i^{(t+1)} := \alpha_i^{(t)} + \eta_i^{(t)}$
15:     Set $\beta_i^{(t+1)} := \beta_i^{(t)} + \ell_i^{(t)}$
16:     if $t \geq w$ then
17:         $\alpha_i^{(t+1)} := \alpha_i^{(t)} - \eta_i^{(t-w)}$
18:         $\beta_i^{(t+1)} := \beta_i^{(t)} - \ell_i^{(t-w)}$
19:     end if
20:     if $f(x) \geq f(x'')$ then
21:         $x := x''$
22:     end if
23: end for
```

Algorithm 1 considers each execution of low-level heuristics as a Bernoulli trial with output $\{0, 1\}$, where 1 denotes that the applied low-level heuristic generated a strictly better solution with respect to a given objective function and search point, and 0 otherwise. The *success probability* of a low-level heuristic donates the probability that, when applied, the heuristic generates a strictly better solution with respect to the current search point and a given objective function [5]. Hence, the success probability of a selected low-level heuristic is the parameter that specifies the output. Due to the fact that Beta distribution being a *conjugate* prior to the Bernoulli distribution, Algorithm 1 models the prior success distributions of low-level heuristics as Beta distributions. This scenario, where the Beta distribution is used to model Bernoulli distributions, is a commonly used setting for the Thompson Sampling and was first used by Thompson [6]. More formally, let $\alpha_i^{(t)}$ and $\beta_i^{(t)}$ be the number of successes and failures respectively, observed from low-level heuristic i until time t. The proposed Thompson Sampling strategy associates each low-level heuristic i with a utility score $U_i^{(t)}$ which is a random variable, at time t of algorithm 1, sampled randomly from a Beta distribution with parameters $\alpha_i^{(t)}$ and $\beta_i^{(t)}$ (i.e. $U_i^{(t)} \sim \text{Beta}(\alpha_i^{(t)}, \beta_i^{(t)})$). Thompson Sampling can be easily implemented in these settings; wherein only the number of successes and failures of low-level heuristics should be stored.

The proposed hyper-heuristic generates an initial solution $x \in \mathscr{X}$ which is evolved by (i) choosing a heuristic from a set of mutational heuristics and a heuristic from a set of local search heuristics; (ii) applying the chosen heuristics to the current candidate solution, thus generating a new solution $x'' \in \mathscr{X}$; (iii) evaluating the newly generated solution x'' and deciding whether it should be accepted or discarded. This scenario (i.e. where mutational and local-search heuristics are successively applied) is referred to as an *iterated local search* approach, which is considered to better explore and exploit the search space of a given problem (see e.g. [20]). In the t-th iteration of Algorithm 1, the utility score of low-level heuristic i, for all $i \in \{1, \ldots, m\}$, is sampled from a Beta $(\alpha_i^{(t)}, \beta_i^{(t)})$ distribution, and the heuristic with the maximum utility score, in each subset of the low-level heuristics, is chosen (see lines $8, 10$ of Algorithm 1). The parameters $\alpha_i^{(t)}$ and $\beta_i^{(t)}$ are initialised to one and updated during the search process based on the performance of low-level heuristics (see lines $12-15$ in Algorithm 1). The sliding-time-window is based on the 'first-in/first-out' principle, and contains the results of the last w iterations of Algorithm 1, where w is the size of the window (see lines $16-19$ of Algorithm 1). Each slot in the window contains the index of the selected heuristic, and whether it generated a better solution with respect to a given objective function. The size of the window w is a parameter of the algorithm. If w is too large, the algorithm may estimate the effectiveness of low-level heuristics based on irrelevant information about their current performance, and if w is too small, the algorithm may quickly forget the differences between the performances of low-level heuristics. Algorithm 1 accepts the newly generated solution if, and only if, it is not worse than the current candidate solution with respect to a given objective function (see lines $20-22$ in Algorithm 1).

3. HyFlex Problems

In the CHeSC 2011 competition, the same hyper-heuristic was evaluated on a range of well-known optimisation problems to assess its generality and applicability to different problems. The organisers of the competition designed the HyFlex framework to facilitate the comparisons between the competing hyper-heuristics. HyFlex (version 1.0) is a software platform providing implementations of several instances of six well-known combinatorial optimisation problems: Boolean satisfiability (SAT), bin packing (BP), personnel scheduling (PS), permutation flow-shop (PF), travelling salesman problem (TSP), and vehicle routing problem (VRP). HyFlex also provides a set of predefined low-level heuristics for each problem, which are classified into four categories: mutation, local search, ruin-recreate, and crossover low-level heuristics. Each low-level heuristic in HyFlex is a variation operator that can be defined as a function returning a complete solution, given one or more complete solutions. Mutation low-level heuristics stochastically perturb the input solution, thus their applications may lead to a worse solution; whereas, local search low-level heuristics search for a locally optimal solution and never return a deteriorating solution with respect to a given objective function. Ruin-recreate low-level heuristics deconstruct part of the given solution and then reconstruct and return a complete solution. Crossover low-level heuristics return a new solution generated by combining components of two input solutions. In HyFlex, each low-level heuristic (with the exception of crossover heuristics) is associated with a problem-dependent parameter that can be somewhat adjusted by the user. For example, the *depth of search* of local-search low-level heuristics is a user-controlled parameter that determines the number of hill-climbing steps. The parameter takes a value between $[0, 1]$: the smaller the value that the parameter takes, the smaller are the hill-climbing steps that the low-level heuristic performs [13].

4. Boolean Test Models

The performance of the proposed hyper-heuristic was assessed on a set of test models to gain insights into its behaviour in a controlled search environment. Such models are necessary for the improvement of the empirical understanding of hyper-heuristics. The proposed models are inspired by the Boolean artificial search scenario in [21], and can be described as follows. There are m low-level heuristics available to work on a given solution. Each low-level heuristic when applied returns a reward either 0 or 1 (i.e. failure 0 or success 1 in generating a better solution with respect to a given objective function). The difference between the low-level heuristics is their success probabilities of getting a reward 1. Let q_i, for $i \in \{1, \ldots, m\}$, be the success probability of the i-th low-level heuristic. Suppose that the success probability of one low-level heuristic is p and the success probabilities of the other low-level heuristics are $q < p$. We refer to the heuristic that has success probability p as the *best* low-level heuristic. As mentioned, the success probabilities of typical low-level heuristics vary between different regions of the search space of a given problem. Switching search models are thus proposed, in which the entire run period is divided into five sub-periods, which are referred to as *search phases*. All search phases contain identical run length (i.e. they have the same number of objective function evaluations). I investigated how the proposed hyper-heuristic adapts the selection probability of the best low-level heuristic on different settings of the distribution of success probabilities of the heuristics, and experimented with six models based on the following assumptions: (1) the number of low-level heuristics $m := 10$; (2) the best low-level heuristic varies from one search phase to another. In particular, the best low-level heuristic in search-phase $1,2,3,4$ and 5 is heuristic $1,2,3,4$ and 5 respectively; (3) the considered distributions of the success probability of low-level heuristics in each model are given in Table 1.

5. Empirical Evolutions

The proposed hyper-heuristic was compared to the CHeSC 2011 competition hyper-heuristics. International competitors submitted 20 hyper-heuristics to the CHeSC 2011 competition. The competing hyper-heuristics were evaluated on five instances per problem domain: SAT, BP, PS, PF, TSP, and VRP. In the CHeSC competition, 31 runs of the same hyper-heuristic on each problem instance (thirty instances in total) were conducted. The median of the objective function values for each problem instance was used to measure the performance of the hyper-heuristics. All the HyFlex problems were minimisation problems (i.e. hyper-heuristics are used to minimise the values of the given objective functions). A ranking method inspired by the Formula One ranking system was used [13]. In each instance, the eight best hyper-heuristics are given the ranks of $10,8,6,5,4,3,2$, and 1, respectively, while the remaining hyper-heuristics are given ranks of zero. In the case of ties (i.e. more than one hyper-heuristic achieving the same median of the objective function value), a tie method is used to assign appropriate ranks to the tied hyper-heuristics (further details are available on the competition website). The winner of the competition is the hyper-heuristic achieving the highest overall cumulative ranks on all problem domains.

The experiments in this current study were run on an Intel i7 CPU at 3.60 GHz with 16 GB RAM. The CHeSC 2011 competition website provides a benchmarking program computing the equivalent time to 600 seconds on the organisers' machines. The program reported that 415 seconds on our machine is equal to 600 seconds on the competition organisers machines. The TSHH was tested 31 times on each of the five CHeSC 2011 instances per problem domain: SAT, BP, PS, PF, TSP, and VRP. In the TSHH, the mutational subset of low-level heuristics consists of the mutational and ruin-recreate low-level heuristics and the local search subset contains the available local search low-level heuristics in HyFlex. The sliding-time-window size in the TSHH is set to 2×10^5 based on initial experiments. In HyFlex, each low-level heuristic (apart from crossover heuristics) consists of a user-controlled parameter that takes a value between $[0,1]$. The proposed hyper-heuristic employs the Thompson Sampling strategy to control these parameters and adapt their values while solving a given problem. Particularly, each parameter value i, for $i \in \{0.1, 0.2, \ldots, 0.9\}$, is associated with a utility score that, in the t-th iteration of the algorithm, sampled from a $Beta(\delta_i^{(t)}, \chi_i^{(t)})$ distribution where $\delta_i(t)$ is the number of times the hyper-heuristic generates a strictly better solution under the i-th parameter value, and $\chi_i(t)$ is the number of times the hyper-heuristic, under the i-th parameter value, fails to produce a strictly better solution with respect to a given objective function.

Table 2 shows the rank of the TSHH compared to the CHeSC competitors' hyper-heuristics on the six CHeSC problem domains. Overall, the proposed hyper-heuristic achieved the second best approach across all of the problem domains. In the PS, PF, and TSP problem domains, the TSHH outperformed all the other hyper-heuristics. However, it did not perform very well in either the SAT and VRP problems, thus raising an interest subject for future work. Figure 3 demonstrates the normalised median of the objective function values of the TSHH and all the competitors' hyper-heuristics per problem domains. The following formula was used to normalise the median of the objective function values of the hyper-heuristics as in [16]:

$$\frac{d_i - d_{\min}}{d_{\max} - d_{\min}} \tag{1}$$

where d_i is the median of objective function values of the hyper-heuristic, $d_{\max}$ is the maximum median value, and $d_{\min}$ is the minimum median value.

The proposed hyper-heuristic was also assessed on the Boolean test models. The size of the window w in the proposed hyper-heuristic was set to 4000 on the proposed test models based on preliminary experiments. The TSHH was compared with several hyper-heuristics. The simple random hyper-heuristic (SRHH) was employed as a baseline for these comparisons, as it selects low-level heuristics uniformly at random [2]. The reinforcement learning hyper-heuristic (RLHH), as proposed in [17], associates each low-level heuristic with a weight, which is increased by one if an improvement is found in the candidate solution, or decreased by one otherwise. User-defined bounds of the weights are used to maintain the weights of low-level heuristics within a certain interval. The RLHH selects the low-level heuristic with the maximum weight. The Markov chain hyper-heuristic (MCHH), proposed in [18], is based on a Markov chain model, in which the low-level heuristics form the state space. The model is fully connected. The weight of each edge represents the transition probability of moving from the current state (i.e. the current heuristic) to the destination state (i.e. any alternative heuristic, including the heuristic itself). All the weights of the edges are initialised to the same value. If the application of the chosen heuristic result in a measure of improvement in the candidate solution, then the weight of the edge leading to the selection of this heuristic is increased by one. The well-known *roulette wheel* selection strategy is employed to select the following heuristic, based on the weights of the edges of the current state. The proposed hyper-heuristic also compared with a popular *confidence interval* based algorithm, known as the *upper confidence bound* (UCB) algorithm proposed in [19]. The UCB hyper-heuristic (UCBHH) chooses a heuristic that maximises the following confidence bound:

$$\frac{\alpha_i^{(t)}}{\alpha_i^{(t)} + \beta_i^{(t)}} + c\sqrt{\frac{2\log(t)}{\alpha_i^{(t)} + \beta_i^{(t)}}} \tag{2}$$

TABLE 1. THE DISTRIBUTIONS OF THE SUCCESS PROBABILITY (SP) OF HEURISTICS IN THE PROPOSED TEST MODELS.

SP	Model 1	Model 2	Model 3	Model 4	Model 5	Model 6
p	0.400	0.200	0.180	0.100	0.100	0.010
q	0.300	0.100	0.150	0.050	0.010	0.005

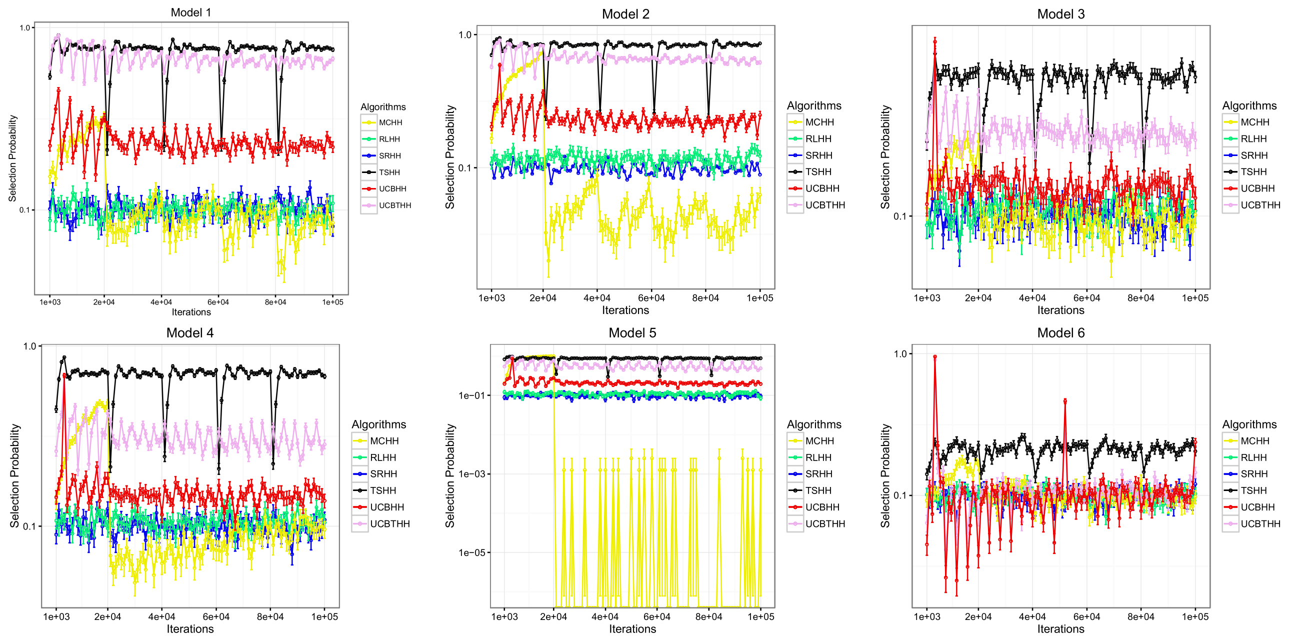

Figure 1. The selection probability of the best low-level heuristic, with error bars, with the Markov chain hyper-heuristic (MCHH), reinforcement learning hyper-heuristic (RLHH), simple random hyper-heuristic (SRHH), Thompson sampling hyper-heuristic (TSHH), UCB hype-heuristic (UCBHH), and UCB-tuned hyper-heuristic (UCBTHH) on the proposed test models estimated over 800 runs. The length of the run is 10^5 iterations.

where $\alpha_i^{(t)}, \beta_i^{(t)}$, and t are as defined in Algorithm 1. The scaling factor c is a user-defined parameter to balance between the exploitation quantity on the left-hand-side and the exploration quantity on the right-hand-side in (2). The scaling factor c in the UCBHH is set to 1 as in [19]. The UCB-Tuned (UCBT) algorithm is a variant of the UCB algorithm proposed in [19] which, at time t, chooses a heuristic that maximises the following quantity:

$$\frac{\alpha_i^{(t)}}{\alpha_i^{(t)}+\beta_i^{(t)}} + \sqrt{\frac{\log(t)}{\alpha_i^{(t)}+\beta_i^{(t)}} \min\left\{\frac{1}{4}, \sigma_i^2 + \sqrt{\frac{2\log(t)}{\alpha_i^{(t)}+\beta_i^{(t)}}}\right\}} \tag{3}$$

where $\alpha_i^{(t)}, \beta_i^{(t)}$, and t are as defined in Algorithm 1, and σ_i^2 is the sample variance. The upper confidence bounds of the *UCBHH* and *UCBTHH* are computed based on the entire history of the run. This implies that as the number of executions of low-level heuristics increases the new results of low-level heuristics are weighted as less important on the estimation of the prescribed upper confidence bounds in (2) and (3). Therefore, both *UCBHH* and *UCBTHH* were combined with a sliding-time-window to accommodate the dynamic change in the success probabilities of low-level heuristics. The size of the window is set to 4000.

Figure 1 shows the selection probability of the best heuristic with the TSHH and the considered algorithms on the proposed test models. In Model 1, the results revealed that the TSHH identified the best low-level heuristic in each search phase. Comparable results were also obtained when the UCBTHH was applied on the same test model. While the UCBHH chooses the best heuristic less frequently than both the TSHH and the UCBTHH, the estimated selection probability of the best heuristic with the UCBHH was larger than that with the MCHH, RLHH, and SRHH. Almost identical results were obtained in Model 2 as shown in the figure. The results indicated that MCHH is able to identify the best heuristic, but is too slow to learn the changes in the success probabilities of heuristics. In Model 3, the difference between the success probability of the best heuristic and sub-optimal heuristics is relatively small, i.e. 0.03. The results have shown that the ability of the TSHH to identify the best heuristic in each search phase as being more effective than with other hyper-heuristics. In Models 4 and 5, the results of the RLHH and SRHH are almost the same with respect to the selection probability of the best heuristic. In Model

6, the success probabilities of the heuristics are somewhat small. However, the success probabilities of typical low-level heuristics on real problems have the potential to be considerably smaller (see e.g. [5]). The TSHH is the only hyper-heuristic capable of distinguishing the differences between the success probabilities of the low-level heuristics. In the first search phase, the MCHH increases the selection probability of the best heuristic. However, it almost chooses heuristics uniformly at random in the other search phases. A high level of variance can be observed in the estimated selection probability of the best heuristic with the UCBHH. Overall, the results revealed that the TSHH is capable of identifying the most effective heuristic in all of the proposed models.

The effectiveness of the proposed hyper-heuristic with respect to the number of low-level heuristics was also assessed. Figure 2 shows the selection probability of the best heuristic with the TSHH and the considered hyper-heuristics at the end of each search phase on Model 2 with m heuristics, for ($m = 10, \dots, 210$).

6. Conclusion

This study has proposed a general-purpose adaptive Thompson Sampling hyper-heuristic and evaluated its performance on a wide range of test models and combinatorial optimisation problems. The results have revealed that the proposed hyper-heuristic effectively identifies the heuristics with the maximum empirical probabilities of improving the candidate solution on general test models. The generality of the proposed hyper-heuristic has been assessed on a broad range of combinatorial optimisation problems. The results have shown that the proposed hyper-heuristic produces competitive, often better, results compared to a large number of hyper-heuristics including the state-of-the-art algorithm. This study has empirical evidence that Thompson Sampling is an effective learning mechanism within a hyper-heuristic framework, and has also provided suggestions as to how it can be improved. In terms of directions for future research, further work could investigate the potential for improving the performance of the Thompson Sampling hyper-heuristic on the Boolean satisfiability and the vehicle routing problems.

7. Acknowledgement

I would like to thank Dr. Joshua Knowles for a fruitful discussion about Thompson Sampling.

References

[1] 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*, vol. 64, no. 12, pp. 1695–1724, 2013.

[2] P. Cowling, G. Kendall, and E. Soubeiga, "A hyperheuristic approach to scheduling a sales summit," in *Practice and Theory of Automated Timetabling III*. Springer, 2001, pp. 176–190.

[3] H. Fisher and G. L. Thompson, "Probabilistic learning combinations of local job-shop scheduling rules," *Industrial scheduling*, vol. 3, no. 2, pp. 225–251, 1963.

[4] E. Özcan, B. Bilgin, and E. E. Korkmaz, "A comprehensive analysis of hyper-heuristics," *Intelligent Data Analysis*, vol. 12, no. 1, pp. 3–23, 2008.

[5] F. Alanazi and P. K. Lehre, "Limits to learning in reinforcement learning hyper-heuristics," in *Evolutionary Computation in Combinatorial Optimization*. Springer, 2016, pp. 170–185.

[6] W. R. Thompson, "On the likelihood that one unknown probability exceeds another in view of the evidence of two samples," *Biometrika*, vol. 25, no. 3/4, pp. 285–294, 1933.

[7] S. Agrawal and N. Goyal, "Further optimal regret bounds for thompson sampling," *arXiv preprint arXiv:1209.3353*, 2012.

[8] O. Chapelle and L. Li, "An empirical evaluation of thompson sampling," in *Advances in neural information processing systems*, 2011, pp. 2249–2257.

[9] O.-C. Granmo, "Solving two-armed bernoulli bandit problems using a bayesian learning automaton," *International Journal of Intelligent Computing and Cybernetics*, vol. 3, pp. 207–234, 2010.

[10] E. Kaufmann, N. Korda, and R. Munos, "Thompson sampling: An asymptotically optimal finite-time analysis," in *Algorithmic Learning Theory*. Springer, 2012, pp. 199–213.

[11] T. Graepel, J. Q. Candela, T. Borchert, and R. Herbrich, "Web-scale bayesian click-through rate prediction for sponsored search advertising in microsoft's bing search engine," in *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, 2010, pp. 13–20.

[12] L. Tang, R. Rosales, A. Singh, and D. Agarwal, "Automatic ad format selection via contextual bandits," in *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*. ACM, 2013, pp. 1587–1594.

[13] E. K. Burke, M. Gendreau, M. Hyde, G. Kendall, B. McCollum, G. Ochoa, A. J. Parkes, and S. Petrovic, "The cross-domain heuristic search challenge–an international research competition," in *International Conference on Learning and Intelligent Optimization*. Springer, 2011, pp. 631–634.

[14] 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*. ACM, 2011, pp. 1987–1994.

[15] J. H. Drake, E. Özcan, and E. K. Burke, "An improved choice function heuristic selection for cross domain heuristic search," in *Parallel Problem Solving from Nature-PPSN XII*. Springer, 2012, pp. 307–316.

[16] A. Kheiri and E. Keedwell, "A sequence-based selection hyper-heuristic utilising a hidden markov model," in *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*. ACM, 2015, pp. 417–424.

[17] A. Nareyek, "Choosing search heuristics by non-stationary reinforcement learning," in *Metaheuristics: Computer decision-making*. Springer, 2004, pp. 523–544.

[18] A. Kheiri and E. Keedwell, "Markov chain selection hyper-heuristic for the optimisation of constrained magic squares," 2015.

[19] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem," *Machine learning*, vol. 47, no. 2-3, pp. 235–256, 2002.

[20] H. R. Lourenço, O. C. Martin, and T. Stützle, *Iterated local search*. Springer, 2003.

[21] Á. R. S. Fialho, "Adaptive operator selection for optimization," Ph.D. dissertation, Citeseer, 2011.

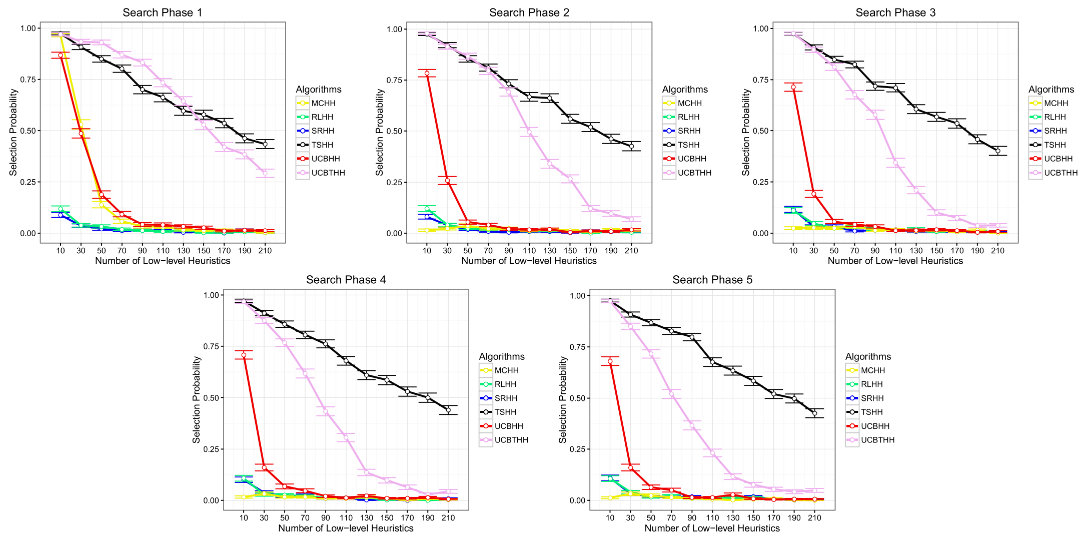

Figure 2. The selection probability of the best heuristic at the end of each search phase with the Markov chain hyper-heuristic (MCHH), reinforcement learning hyper-heuristic (RLHH), simple random hyper-heuristic (SRHH), Thompson sampling hyper-heuristic (TSHH), UCB hype-heuristic (UCBHH), and UCB-tuned hyper-heuristic (UCBTHH) on Model 2 estimated over 500 runs. The length of the run is 10^6 iterations.

TABLE 2. THE RANK OF THE PROPOSED THOMPSON SAMPLING HYPER-HEURISTIC (TSHH) ON HYFLEX COMPARING WITH ALL THE CHESC 2011 COMPETING HYPER-HEURISTICS

Method	SAT	BP	PS	PF	TSP	VRP	Overall
AdapHH	**34·75**	**45.00**	7·00	31·00	34·75	14·00	166·50
TSHH	0·00	24·00	**47.00**	**44.00**	**38.00**	7·00	160·00
VNS-TW	34·25	2·00	30·50	27·00	13·75	4·00	111·50
ML	14·50	8·00	25·50	32·50	10·00	21·00	111·50
PHunter	10·50	3·00	9·50	6·00	22·75	**33.00**	84·75
EPH	0·00	7·00	7·50	16·00	29·75	12·00	72·25
HAHA	32·75	0·00	21·50	0·83	0·00	14·00	69·08
NAHH	14·00	17·00	1·00	19·50	10·00	6·00	67·50
KSATS-HH	24·00	9·00	5·50	0·00	0·00	22·00	60·50
ISEA	6·00	27·00	11·50	1·50	8·00	4·00	58·00
HAEA	0·50	2·00	1·00	5·33	8·00	27·00	43·83
ACO_HH	0·00	19·00	0·00	6·33	7·00	1·00	33·33
GenHive	0·00	12·00	4·50	5·00	2·00	6·00	29·50
XCJ	5·50	11·00	0·00	0·00	0·00	5·00	21·50
AVEG-Nep	12·00	0·00	0·00	0·00	0·00	9·00	21·00
SA_ILS	0·75	0·00	14·00	0·00	0·00	4·00	18·75
DynILS	0·00	9·00	0·00	0·00	9·00	0·00	18·00
GISS	0·75	0·00	8·00	0·00	0·00	6·00	14·75
MCHH-S	4·75	0·00	0·00	0·00	0·00	0·00	4·75
SelfSearch	0·00	0·00	1·00	0·00	2·00	0·00	3·00
Ant-Q	0·00	0·00	0·00	0·00	0·00	0·00	0·00

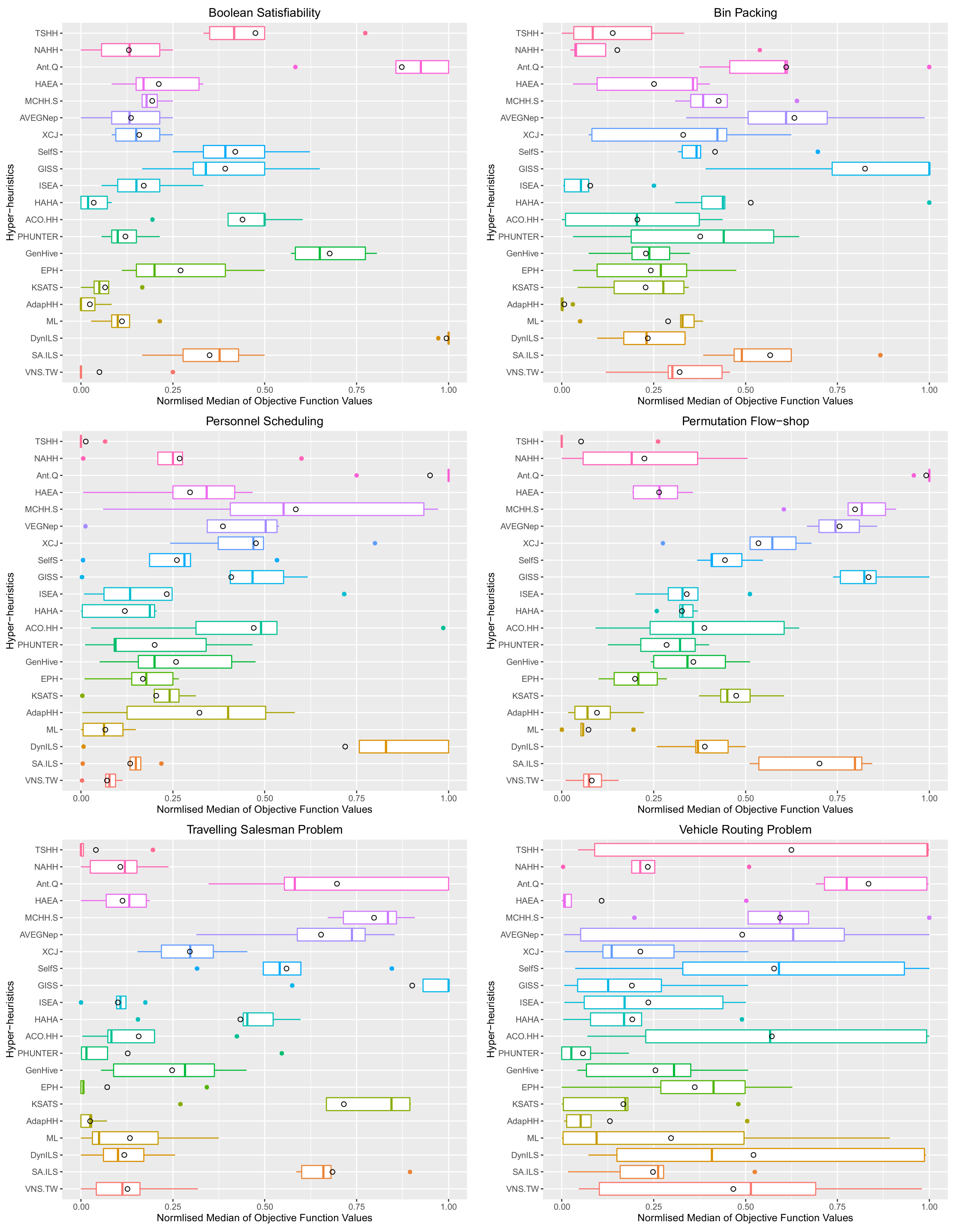

Figure 3. Box plots of the normalised median of the objective function values of the proposed Thompson Sampling hyper-heuristic (TSHH) and all the CHeSC competing hyper-heuristics on five instances per problem domain in HyFlex. The small circles indicate the means.