Tyasnurita, Raras and Özcan, Ender and Shahriar, Asta and John, Robert (2015) Improving performance of a hyper-heuristic using a multilayer perceptron for vehicle routing. In: 15th UK Workshop on Computational Intelligence (UKCI 2015), 7-9 Sep 2015, Exeter, UK.

Improving Performance of a Hyper-heuristic Using a Multilayer Perceptron for Vehicle Routing

Raras Tyasnurita[1], Ender Özcan[2], Shahriar Asta[3], and Robert John[4]
[1,2,3,4] ASAP Research Group, School of Computer Science, University of Nottingham, Nottingham, United Kingdom NG8 1BB
[1] Department of Information Systems, *Institut Teknologi Sepuluh Nopember (ITS)*, Surabaya, Indonesia 60111
Email: {rxt,exo,sba,rij}@cs.nott.ac.uk

***Abstract*—A hyper-heuristic is a heuristic optimisation method which generates or selects heuristics (move operators) based on a set of components while solving a computationally difficult problem. Apprenticeship learning arises while observing the behaviour of an expert in action. In this study, we use a multilayer perceptron (MLP) as an apprenticeship learning algorithm to improve upon the performance of a state-of-the-art selection hyper-heuristic used as an expert, which was the winner of a cross-domain heuristic search challenge (CHeSC 2011). We collect data based on the relevant actions of the expert while solving selected vehicle routing problem instances from CHeSC 2011. Then an MLP is trained using this data to build a selection hyper-heuristic consisting of a number classifiers for heuristic selection, parameter control and move acceptance. The generated selection hyper-heuristic is tested on the unseen vehicle routing problem instances. The empirical results indicate the success of MLP-based hyper-heuristic achieving a better performance than the expert and some previously proposed algorithms.**

I. Introduction

The design and development of effective heuristic optimisation methods to real-world problems is often time-consuming and becoming increasingly complex. Hence, there is a considerable interest in automating the design of heuristic search methods and obtaining more general approaches applicable to various instances with different characteristics or multiple domains [12], [22]. Hyper-heuristics have emerged as such methods [16]. A hyper-heuristic is a high level search method or a learning mechanism that *selects* or *generates* a set of low level heuristics for solving hard computational problems. A selection hyper-heuristic performs a single point based search using a single active solution and fixed set of low level heuristics. At each step, a heuristic is chosen and applied to a solution and the resultant solution is considered to replace the incumbent solution using a move acceptance method. In this study, each low level heuristic is *perturbative*, processing and returning a complete solution at all times. The hyper-heuristic research has the potential of bringing together promising ideas from the field of Machine Learning accumulated over the years into heuristic optimisation [10]. A diagram of hyper-heuristic framework in figure 1 illustrates the *domain barrier* between hyper-heuristic and problem domain layer. Any problem domain specific knowledge is not allowed to pass through that barrier. However, domain independent information, such as, quality of a solution, number of heuristics available can be accessed by a hyper-heuristic.

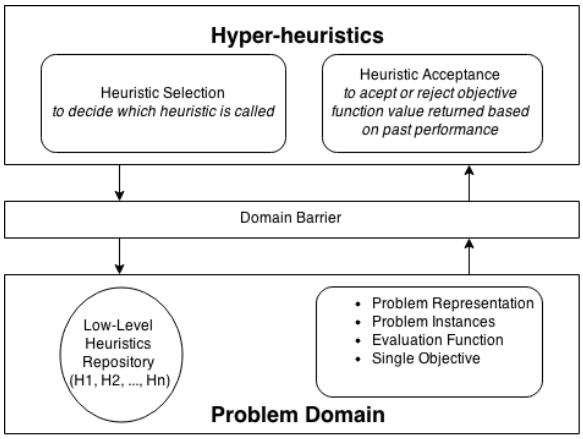


Fig. 1: Hyper-heuristics Conceptual Framework [8]

There are different classifications for hyper-heuristics based on different criteria [11], [13]. This study focuses on an *offline* approach based on apprenticeship learning [1] which generates a selection hyper-heuristic for solving a Vehicle Routing Problem (VRP) [17]. Asta and Özcan [5] applied apprenticeship learning using the VRP domain from a hyper-heuristic benchmark implemented as a part of Hyper-heuristic Flexible framework (HyFlex) [32]. They have used decision trees, C4.5 as the machine learning approach. However, in this study, we use a multilayer perceptron (MLP) and compare its performance to C4.5 and some other previously proposed approaches on the same problem domain. MLP learns from an expert selection hyper-heuristic through training (observation) how to perform heuristic selection, move acceptance and set relevant parameters using a set of sample instances and then mimics the behavior of the expert operating as a 'new' selection hyper-heuristic for solving unseen (test) instances.

This paper is organised as follows. Section II provides an overview of related work, Multilayer Perceptron, Vehicle Routing Problem, and Cross-Domain Heuristic Search Challenge (CHeSC), successively. Section III describes the proposed approach. Section IV presents the experimental results. Finally, section V summarises our findings and points out potential research directions.

II. Background

A. Related Work

The idea of automating the design of algorithms has been explored from many perspectives since the initial work of machine learning by Arthur Samuel in 1959 [31]. According to [34], the latest machine learning research can operate at a higher level of generality than hyper-heuristic. The concept of a heuristic in the optimisation community is similar to that of a classifier in the machine learning community. Since both communities share a common goal which is providing algorithms ability to generalise to new datasets, it motivates to closer integration on these two fields.

Apprenticeship learning is mostly applied in the field of robotics [1]. In a previous work [5], a machine learning approach based on apprenticeship learning with the C4.5 classifier was implemented to build a generalised selection hyper-heuristics. The approach was initially trained on small problem instances in the VRP domain consisting of two representative problem instance classes (Solomon and Gehring-Homberger [40]). It was capable of generalizing the extracted knowledge to unseen problem instances. The result outperforms the expert which used an Adaptive Dynamic Heuristics Set (ADHS) or AdapHH strategy [30]. It is worth mentioning that it also delivers outstanding performance compared to some other previously proposed selection hyper-heuristics in various occasions on HyFlex VRP instances. This work encourages us to extend the previous work in order to gain a better performance by implementing a Multilayer Perceptron (MLP). Other work on apprenticeship learning was implemented in Bin Packing, which can be found in [7].

This study is also inspired from work in [4], which found that between two classification algorithms of Waikato Environment for Knowledge Analysis (Weka) interface, MLP is a better algorithm compared with J48 (Java implementation of C4.5 algorithm) in most of the datasets available from The University of California Irvine (UCI) Machine Learning Repository. Weka is a popular open source data mining and machine learning platform implemented in Java [43]. Algorithms based on neural network have better learning capability if trained properly. Besides, in this case it only requires a small training times.

Actually, function approximation with MLP has been implemented by one of the contestants of the CHeSC 2011 competition [18]. MLP as learning agent became one component on the algorithm proposed based on Reinforcement Learning (RL). That is, in [18], the proposed algorithm is only responsible for heuristic selection and no move acceptance strategy is employed. One major difference between our approach and the one in [18] is that we also build move acceptance classifiers to improve the performance.

There are a number of related studies on machine learning in hyper-heuristic. One work observed off-line learning hyper-heuristic using Evolutionary Algorithms (EAs) in bin-packing problems [38]. EAs is excellent for searching very large spaces. Other work investigated on-line learning evolutionary hyper-heuristics which solves Dynamic VRP [20]. On-line learning is good if there is lack of availability instances and more suitable for dynamic problems.

B. Multilayer Perceptron

Multilayer Perceptron (MLP) is the most popular neural network architecture, which utilises back-propagation on error-correction learning rule, and process input by feeding it forward from one layer to the next layer. There are three layers in MLP: input, hidden, and output layers. Artificial Neural Network (ANN) is able to predict with high accuracy after it has been taught with historical data. MLP have been applied successfully to solve many real world applications. Some well-known application areas are airline marketing tactician, data compression, financial prediction, hand-written character recognition, autonomous driving, ECG noise filtering, protein secondary structure, psychiatric patient length of stay, and speech recognition [36].

MLPs pass the output of their layers through an activation function. MLP usually use sigmoid (logistic) activation function in the feed forward. By far this is the most common form of activation function.

Sigmoid (logistic) activation function is defined by [23] :

$$y_j = \frac{1}{1 + exp(-v_j)} \tag{1}$$

Differentiability is the only requirement that an activation function has to satisfy. This form is defined by :

$$y_j = \begin{cases} 0 & for \quad \sum_{j=1}^{n} w_j x_j \leq threshold \\ \frac{1}{1+exp(-v_j)} & for \quad \sum_{j=1}^{n} w_j x_j > threshold \end{cases} \tag{2}$$

where v_j is the weighted (w_j) sum of all synaptic inputs (x_j) of neuron j, and y_j is the output of the neuron.

Tuning the MLP topology design is a challenge itself because the task is complex and commonly addressed by simple trial and error procedures. There are no constraints on the number of hidden layers, however it has been demonstrated in [25] that an MLP with a single hidden layer can approximate any bounded continuous function with arbitrarily small error. Hence for a preliminary work, it is sensible to focus on one hidden layer and not complicate the MLP structure unnecessarily.

Moreover, parameters tuning is needed to obtain an optimal classifier. Four main parameters for learning MLP are the number of hidden nodes or neurons, learning rate, momentum, and number of training time (iterations). First is in terms of neurons number, because MLP is intended to do classification, then it is often preferable to have one output neuron for each group that input items are to be assigned into. The number of output in MLP depends on the number of classes in the dataset, while the input number depends on the attributes in the dataset. There are many rule-of-thumb for determining the right number of hidden neurons to provide a starting point. One piece of guidance is from [24]. It states that there are three rules for determining the correct number of neurons to

use in the hidden layer, namely the number should be between the size of input and output layer, should be 2/3 the size of input layer plus size of output layer, or should be less than twice the size of input layer.

The second parameter to be considered is learning rate. Too high a learning rate makes the perceptron periodically oscillate around the solution. At this condition, most networks do not converge or they converge to a poor solution and become stuck. Learning rate value is mostly less than or equal to 0.2 based on [2]. In addition, a typical ranges of learning rate is [0.05 $\leq$ learning rate $\leq$ 0.75] [37]. However, the common number of learning rate is 0.1 which is often suggested according to [42].

Momentum as third parameter is a method to reduce problems of instability while increasing the rate of convergence. In general, small learning rate values call for larger momentum values to increase the speed and probability of convergence. Large values of momentum will allow the algorithm to remember more terms in the adjustment history.

C. Vehicle Routing Problem

Vehicle Routing Problem (VRP) aims to design an optimal distribution of delivery from a central depot to a number of customers subject to constraints. The objective of VRP is to minimize efforts such as route length, total driving costs or driving time, while the common constraints include vehicle capacity, vehicle type, number of cities, and which precedence relations between pairs of cities for delivering services to a customer have to be satisfied [29]. Recent study in VRP for example is in [44] who proposes a new Electric Vehicle (EVs) route optimization model where green VRP is a relatively new promising research topic in terms of energy saving. Besides, various variants of VRP are observed in [35] by presenting a general heuristic for all. Some previous works on the latest advances of VRP can be found in [21].

In this study, Vehicle Routing HyFlex model is used which include an extra variant, namely time window limit. Solution will be valid if a customer is served within this time range. The objective function for VRP HyFlex domain is subject to minimization of the number of vehicles and travelled distance [41]. Objective function can be defined as follows :

$$obj = c \times v + d \tag{3}$$

where v is the number of vehicles, d is the travelled distance, and c, initially set to 1000, is the level of importance to the number of vehicles.

VRP instances which currently available in HyFlex framework are taken from two sources, Solomon and Gehring-Homberger with five instances from each source. For these two sources, there are three types of instances which depend on the determination of the customer's location. They are Random (R), Clustered (C), or Clustered Random (RC). Solomon and Gehring-Homberger have been utilized by many researchers as benchmark datasets in VRP. Further details on the list variant of instances can be seen in table I. The main difference between instances are the number of vehicles and their capacities.

TABLE I: VRP instances in HyFlex [27]

Instance	Name	No.Vehicles	Vehicle Capacity
0	Solomon/RC/RC207	25	1000
1	Solomon/R/R101	25	200
2	Solomon/RC/RC103	25	200
3	Solomon/R/R201	25	1000
4	Solomon/R/R106	25	200
5	Homberger/C/C1-10-1	250	200
6	Homberger/RC/RC2-10-1	250	1000
7	Homberger/R/R1-10-1	250	200
8	Homberger/C/C1-10-8	250	200
9	Homberger/RC/RC1-10-5	250	200

There are ten low-level heuristics for the VRP domain implemented within HyFlex, which are categorized as follows [32].

1) Mutation (MU): modifies solution component
2) Ruin-Recreate (RR): partially destroys then repairs solution
3) Hill Climbing (HC): conducts iterative moves in neighbourhood to improve quality of solution
4) Crossover (XO): takes two candidate solutions and then returns a new solution as offspring

The indices of the low level heuristics in HyFlex for HC are {4,8,9}, MU are {0,1,7}, RR are {2,3} and XO are {5,6}.

Hill Climbing is used for intensification to perform local search in a particular region, while mutation, ruin-recreate, and crossover are used for diversification to explore other good regions of the search space. Furthermore, there are two parameters that control the low-level heuristics behaviour namely Intensity of Mutation (IoS) for mutation and ruin-recreate heuristics and Depth of Search (DoS) for hill climbing.

D. Cross-Domain Heuristic Search Challenge

Because HyFlex is an easy to use platform in the form of flexible Java class library, HyFlex v1.0 was used to support the first Cross-Domain Heuristic Search Challenge (CHeSC 2011), which is a competition run and organised by Automated Scheduling, Optimisation and Planning (ASAP) group at the University of Nottingham, Nottingham, United Kingdom in 2011. The goal has been to promote development of effective general search methodologies and get more insights into different strategies in algorithm design [9]. CHeSC 2011 aimed at finding the state-of-the-art selection hyper-heuristic which performs the best across six different problem domains.

The scoring system was based on median performance inspired by Formula 1. The top eight algorithms receive 10, 8, 6, 5, 4, 3, 2 and 1 points respectively, while the remaining algorithms receive no point. These points are added across number of instances. Algorithm with highest points is the winner.

CHeSC 2011 attracted 20 participants across the globe to implement HyFlex as software interface. For testing, there were six problem domains provided including two hidden domains. Five instances from each domain that were selected consist of three instances from training set provided and another two were hidden. The hidden parts aimed to reach

generality of algorithm. Participants needed to benchmark their machines with program provided to know the time limit they can run their algorithms. This program corresponds to ten minutes of computer usage in the competition. Moreover, scoring was based on a typical 31 run by reusing the random seeds so that each algorithm and instance will start from the same initial state.

PHUNTER [14] ranked the first on the VRP problem domain in CHeSC 2011. The proposed selection hyper-heuristic is configured using an offline training session first via a decision tree. The approach is based on an iterative local search performing intensification and diversification when needed. Adaptive Hyper-heuristic (AdapHH) was the winner of CHeSC 2011 [30]. AdapHH outperformed its competitors in three problem domains: MAX-SAT, BP, and TSP. It ranked tenth in PS, second in FS, and fifth in VRP. AdapHH implements adaptive features to manage heuristic sets by adapting heuristic parameters on-line. Based on performance metric with quality indicators such as speed and improvement capability, it will decide which heuristics have to be excluded. It also investigates relay hybridisation to determine effective pairs of heuristics. For accepting mechanism, AdapHH implements adaptive iteration limited list-based threshold by using fitness values of previous best solutions.

III. Proposed Approach

Our approach is that the MLP algorithm constructs a classifier for each dataset. This datasets were generated by the training phase of Apprenticeship Learning (AL) based on the work in [5]. AL represents each state of search space by a feature vector. Then after defining necessary actions, extracting of corresponding actions from expert algorithm was executed. The extraction was for each state of the search that is inserted into various datasets. The data were collected only if the expert accepted the solution.

The expert was running on instance 0 from Solomon and instance 5 from Gehring-Homberger. These two instances were chosen arbitrarily to represent the generalization, because they were taken from each class of VRP HyFlex domain. After that, thirteen datasets were constructed, and the two instances combined together. In each dataset, there are eight values of delta as the attributes. Delta means the change in evaluation function (fitness value) from current solution to candidate solution in eight previous consecutive time. This number is sufficient in AL. Dataset was split into 13. One for predicting heuristic selection, one for estimating Depth of Search (DoS) parameter value, one for estimating Intensity of Mutation (IoM) parameter value, and the other ten for each low-level heuristics move acceptance criteria. These accepting criteria are :

1) Equal Accepted (EA) : accept candidate solution even if it is the same with current solution
2) Worsening Accepted (WA) : accept candidate solution even if it is worse than current solution

Improving solutions are always accept, however the policy to deal with non-improving (equal or worsening) solution is also critical in order not to get trapped in local optima. Besides, the systems can only accept the heuristic index chosen in previous step. Consequently, attention is given for both heuristic selection and acceptance process.

First phase in this work is generating classifier for each datasets using MLP. Since the procedure for finding a good enough MLP model requires trial and error, the number of training times was limited to 500 epochs. As stated in II-B, the parameters for MLP learning, consist of the learning rate, momentum, and number of hidden neurons. In this case, data normalization is not necessary. This is because MLP classifier can learn to apply appropriate scaling to the pattern attributes. Meaning that any rescaling of input can be effectively undone by changing the corresponding weights and biases. So that applying data normalization or not will give the same outputs in both ways [39].

Cross-validation with folds number ten is used as test options in doing the classification. This will split the whole dataset into equal sized subsets. It returns the averaged value of the prediction scores of each subset obtained on the union of all the other subsets [28]. Prediction scores for example are returned in the form of an estimation classifier's error rate. This approach does not waste too much data, which is a major advantage in problem where the number of samples is very small.

Second phase is the testing stage which applies the MLP classifiers to unseen instances. The workflow of MLP learning hyper-heuristics is illustrated in figure 2. There are five steps, namely initialize solution, select heuristic based on MLP classifier, set parameter (DoS or IoM) based on low level heuristic chosen, apply heuristic, and accept heuristic that is also based on classifiers generated by MLP. The process will stop when the time limit which uses benchmark time in CHeSC 2011 is reached.

Modifying low level heuristics can be done by the use of Depth of Search (DoS) and Intensity of Mutation (IoM) parameters. Initial value for both parameters in HyFlex is set to 0.2 in the range [0,1], which generally relates to number of improving steps. Changing the value will modify the behaviour of low level heuristics. Each parameter's meaning depends on the heuristic. DoS only affects hill-climbing, while IoM only affects mutation and ruin-recreate. Higher values in DoS means that the hill-climbing heuristic searches more neighbourhoods for improvement. Higher values in IoM means that more variables are changed for mutation heuristic. For example, IoM = 0.5 for ruin-recreate heuristic means that a half of solution will be destroyed and rebuilt. Since the outcome are real-valued, not discrete set, so regression is implemented rather than classification in task of learning process [15].

The two sided Wilcoxon signed-rank test is applied to verify how close the best value performance is between this work and MLP-ALHH throughout the text, and previous work or C4.5-ALHH throughout the text. The test is performed at 95 % confidence level. The test takes score differences in pairs and ranks them by absolute value in ascending order. The sign

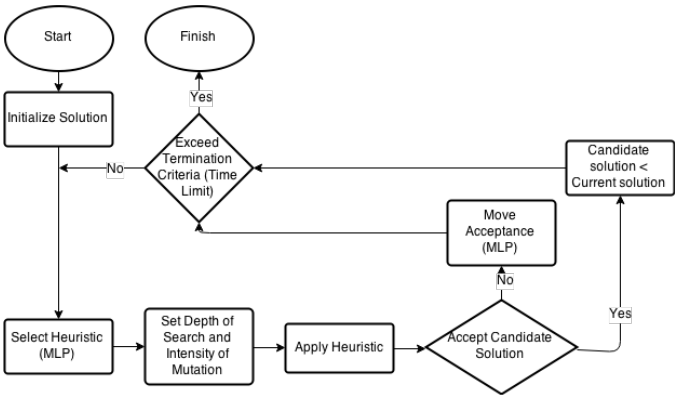


Fig. 2: Workflow of MLP Learning Hyper-heuristics

(negative or positive) of each difference is given to its rank as a label [26].

IV. Experimental Results

A. Experimental Design

We have applied the proposed approach on the VRP instances implemented in HyFlex. The results are then compared to those achieved in [5] and elsewhere. In order to maintain fairness in our comparisons, we have used precisely the same settings as in [5]. Our approach consists of a training phase followed by testing. Instances 0 and 5 have been chosen from Solomon and Gehring-Homberger datasets respectively for training (refer to Table I). Similar to the study in [5], a number of datasets are generated after the training process. That is, one dataset is trained to model expert's heuristic selection strategy, two datasets to separately model the choice of values for DoS and IoM parameters, and one dataset to predict the acceptance strategy for each low level heuristic available for VRP instances. After training, the generated models are applied on all the instances (indexed from 0 to 9 in Table I) where 31 independent runs have been repeated for each instance. Similar to the study in [5], the training instances (indexed 0 and 5 in Table I) are also included in the test phase. The inclusion of these instances play a confirmatory role assessing the success of the trained model. The experiments are performed on an Intel(R) Core(TM)i7 Windows 7 Enterprise (3.20 GHz) with 6 GB RAM. Each trial terminates after 600 nominal seconds with respect to the CHeSC 2011 machine.

Regarding the setting for the MLP, various configurations have been considered and experimented for the heuristic selection dataset. The best performing MLP model contains one hidden layer with 20 hidden neurons operating with a learning rate of 0.1, momentum of 0.1, and 500 iterations for training. Following the training, it was observed that the trained model has a 50.3 % accuracy. This value is worse than the accuracy rate of 65.0 % obtained by the apprenticeship learning approach using the C4.5 algorithm in [5]. This result however is not that surprising, considering the previous work by Amancio et al [3]. They have obtained an accuracy rate of 50.9 % and 67.1 % with MLP and C4.5 respectively, while classifying DB10F (artificial datasets modelling various characteristics of real data) which data pattern is also not well-localized. It is imperative to note that the accuracy of the heuristic selection machine is not vital. Firstly, an accuracy of 50.3 % does not suggest that the heuristic selection machine is choosing randomly among *all* heuristics. Instead, it merely means that the heuristic selection mechanism is nearly random and it chooses from five particular heuristics which are more favoured by the expert algorithm (AdapHH) as projected to the collected dataset. Here we ignore the cross-over heuristics since they are not favoured by the expert. Secondly, it has been observed in previous studies that it is only combined with the acceptance mechanism that the heuristic selection strategy influential on the overall performance of a hyper-heuristic [33]. This will be further confirmed throughout experiments where despite a seemingly low accuracy of the heuristic selection machine, extra-ordinary results are achieved for various instances.

Considering the heuristic acceptance datasets, and emphasizing that there are one such dataset per heuristic, the following network configuration has been observed to perform the best. The MLP model for heuristic index three is a network with a single hidden layer of 5 neurons, learning rate of 0.3 and a momentum of 0.2. The configuration for heuristic index six is one hidden layer of 20 neurons, learning rate of 0.1 and momentum isequal to 0.1. As for the rest of the heuristics, the expert (AdapHH) apparently has a fixed strategy of Equal Accepted (EA) for heuristics 0, 1, 4, 8 and 9 and Worsening Accepted (WA) for heuristics 2, 5 and 7. The prediction accuracy of the two MLP models trained for heuristics three and six are 80.0% and 93.5% respectively. The accuracy rate achieved in [5] was 90.0%.

By using Linear Regression (LR) to predict the real values of Depth of Search and Intensity of Mutation, value of 0.5824 is obtained with correlation coefficient 0.3. This correlation is described as moderate. The LR also supports the MLP-ALHH to improve the performance of hyper-heuristic.

B. Experimental Analysis

The results of experiments is shown in table II. The performances are measured by comparing average, minimum or maximum value of the objective function. In majority of instances (seven out of ten), MLP-ALHH manages to outperform C4.5-ALHH in terms of average and median objective function value. The MLP-ALHH outperforms its rival not only on training instances, but also on unseen test instances. Even though instance zero is a training instance, C4.5-ALHH has lower average and median performance compared to MLP-ALHH. However, there is no significant difference (statistically) between the two. This is an indication of generalization capability of our approach, encouraging us to extend this work to a cross-domain level in future.

A Two sided Wilcoxon signed-rank test is performed to compare the performance of MLP-ALHH to C4.5-ALHH (Table III). Z-value less than or equal to -1.96 or z-value

TABLE III: Two Sided Wilcoxon Signed Rank Test on the performance based on the objective values obtained by MLP-ALHH and C4.5-ALHH over 31 trial for each VRP instance. $\geq$ ($>$) indicates that MLP-ALHH performs slightly (significantly) better than C4.5-ALHH (within confidence interval of 95%), while $\leq$ ($<$) indicates vice versa

instance	0	1	2	3	4	MLP/C4.5
Solomon z-value	$\leq$ -1.94	$>$ -3.99	$>$ -4.50	$>$ -2.72	$>$ -2.23	4/1
instance	5	6	7	8	9	MLP/C4.5
Homberger z-value	$>$ -3.03	$<$ -4.85	$<$ -4.52	$>$ -4.80	$>$ -4.58	3/2

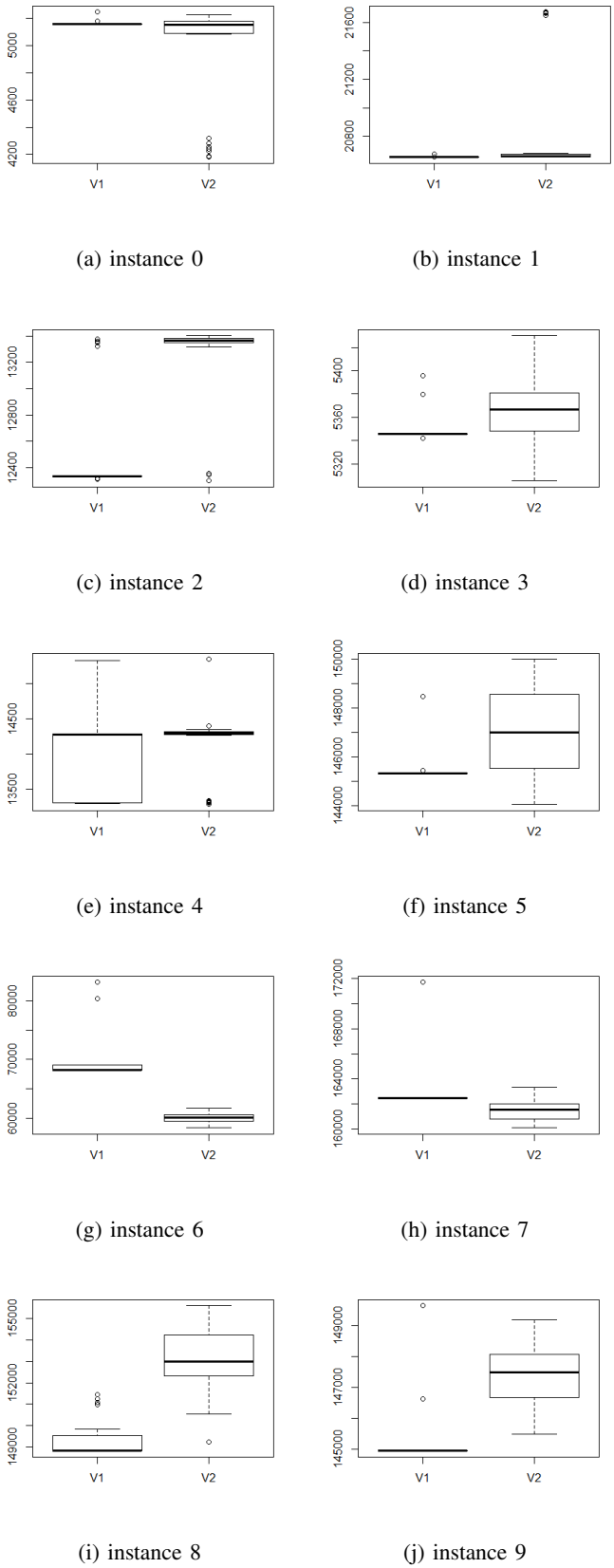
more than or equal to 1.96 indicates the rejection of the null hypothesis, leading to the conclusion that there is significant difference between the results. The table shows that MLP-ALHH performs significantly better than C4.5-ALHH on all the instances except for instance six and seven. On four out of five Solomon instances and three out of five Gehring-Homberger instances, MLP-ALHH perform significantly better than C4.5-ALHH. In order to provide a better visual difference between two algorithms, box plot comparison for all ten instances are provided in Figure 3.

On the majority of instances, the variance of MLP-ALHH is notably less than the C4.5-ALHH approach. This indicates that the MLP-ALHH method is more reliable in terms of consistency as compared to the case when apprenticeship learning uses decision trees. Interestingly, even though the accuracy level in heuristic selection and acceptance is lower than C4.5-ALHH, MLP-ALHH outperforms C4.5-ALHH in generalising to the unseen VRP instances better. This could be also an indicator that the data collected from a stochastic online learning hyper-heuristic algorithm contains some noise/imprecision even if the machine learning produces a high accuracy rate while detecting a pattern.

When MLP-ALHH is compared to the participants of CHeSC 2011, MLP-ALHH achieves the highest score of 29 out of the maximum score of 50, performing slightly better than the winner in the VRP domain, PHUNTER [14] which receives a Formula 1 score of 28 points. Table II shows that MLP-ALHH produces a better performance achieving a better median value on the instances five and nine.

MLP-ALHH is also compared to AdOr-ILS [41] which is an adaptive Iterated Local Search hyper-heuristic incorporating two components: adaptive operator selection and adaptive ordering of the local searcher heuristics.

AdOr-ILS was reported to have a superior performance based on twenty trials when compared to another adaptive ILS variant and a non-adaptive variant which randomly chooses a mutational and then local search heuristic invoking them successively at each iteration. The crossover operators are ignored by those approaches as our approach. Table II shows that MLP-ALHH performs even better than AdOr-ILS for 9 out of 10 instances on average.



(a) instance 0 (b) instance 1

(c) instance 2 (d) instance 3

(e) instance 4 (f) instance 5

(g) instance 6 (h) instance 7

(i) instance 8 (j) instance 9

Fig. 3: Difference Between MLP-ALHH (V1) and C4.5-ALHH (V2) in VRP HyFlex Domain

TABLE II: Performance comparison of MLP-ALHH to C4.5-ALHH and PHUNTER in VRP CHeSC 2011 instances. Comparing MLP-ALHH to C4.5-ALHH, the bold entries represent algorithm which give better performance in average, the underline entries correspond to algorithm which give better performance in median. Comparing MLP-ALHH to PHUNTER, the italic entries refer to algorithm with better performance in median. The '-' entries means non-competition instances

		Solomon					Homberger				
algorithm	instance	0	1	2	3	4	5	6	7	8	9
MLP-ALHH	mean	5168.7	**20656.5**	**12461.4**	**5349.4**	**13903.1**	**145723.9**	69772.0	163676.3	**149366.7**	**145420.1**
	min	5161.9	20653.5	12311.8	5341.9	13296.5	145309.7	68185.1	162484.9	148815.9	144942.5
	median	5161.9	20655.7	12333.0	5346.0	14270.3	*145309.7*	68185.1	162484.9	148815.9	*144942.5*
	std.dev	16.9	3.3	348.7	12.2	554.6	1072.3	3863.7	3146.6	861.0	1033.0
C4.5-ALHH [5]	mean	**4954.6**	20792.8	13266.7	5365.2	14113.8	147017.6	**60101.9**	**161491.5**	153132.2	147414.9
	min	4178.8	20653.3	12300.2	5305.2	13277.0	144037.7	58352.6	160084.5	149227.1	145478.3
	median	5156.4	20661.2	13365.5	5366.7	14294.0	146988.0	60163.0	161529.8	153000.2	147480.9
	std.dev	394.2	340.9	310.9	29.4	481.3	1780.5	790.0	842.7	1663.2	956.8
PHUNTER [14]	min	-	20650.8	12263.0	-	-	143663.9	61139.3	-	-	146472.9
	median	-	*20650.8*	*12290.0*	-	-	146944.4	*64717.8*	-	-	148659.0
AdOr-ILS [41]	mean	5281.7	21291.9	13605.0	6564.4	14280.8	155305.5	77302.7	163177.7	158941.9	149447.7
	std.dev	334.6	482.6	451.6	554.8	319.5	6154.2	3384.8	2100.1	2460.7	1500.9

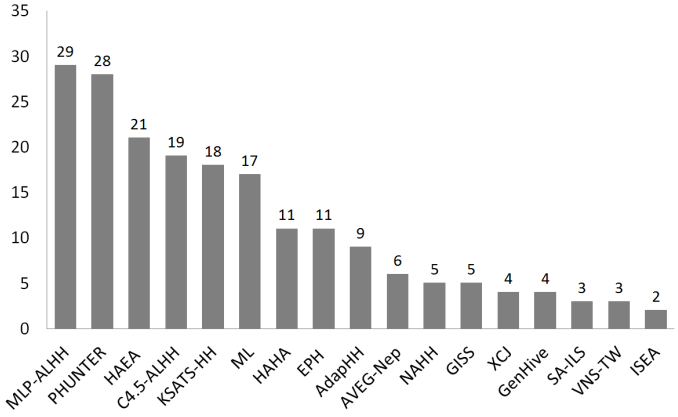


Fig. 4: Formula 1 scoring and ranking of MLP-ALHH, C4.5-ALHH and other CHeSC 2011 competing hyper-heuristics.

V. Conclusion

This preliminary work illustrates that MLP based approach is indeed capable of learning from the state-of-the-art expert hyper-heuristic and automatically generating classifiers forming a well performing selection hyper-heuristic to solve a vehicle routing problem. The apprenticeship learning MLP generalises and even improves the performance of the expert to unseen instances. The empirical results show that the generated selection hyper-heuristic performs better than some other previously proposed approaches.

Learning from a hyper-heuristic optimisation algorithm operating with limited information allowed through the domain barrier is extremely challenging. Different machine learning algorithms performs differently in hyper-heuristic generation. There is already an indication that machine learning algorithms would become even more useful in an information rich environment where they have access to the problem domain [6].

This work encourages us for a further study. We will test the performance of different machine learning algorithms, particularly, Adaptive Network based Fuzzy Inference System (ANFIS). Based on [19], ANFIS was able to achieve a better generalization capability in prediction as compared to MLP. Moreover, we will give the machine learning algorithm access to the problem domain as well for improving heuristic optimisation.

Acknowledgment

First author would like to thank Indonesia Endowment Fund for Education or *Lembaga Pengelola Dana Pendidikan (LPDP)*, Ministry of Finance, Republic of Indonesia as sponsor for PhD studies at the University of Nottingham.

References

[1] Pieter Abbeel and Andrew Y Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, page 1. ACM, 2004.

[2] Ethem Alpaydin. *Introduction to machine learning*. MIT press, 2014.

[3] Diego Raphael Amancio, Cesar Henrique Comin, Dalcimar Casanova, Gonzalo Travieso, Odemir Martinez Bruno, Francisco Aparecido Rodrigues, and L da F Costa. A systematic comparison of supervised classifiers. *PloS one*, 9(4):e94137, 2014.

[4] Rohit Arora and Suman Suman. Comparative analysis of classification algorithms on different datasets using weka. *International Journal of Computer Applications*, 54(13):21–25, 2012.

[5] Shahriar Asta and Ender Özcan. An apprenticeship learning hyper-heuristic for vehicle routing in hyflex. In *Evolving and Autonomous Learning Systems (EALS), 2014 IEEE Symposium on*, pages 65–72. IEEE, 2014.

[6] Shahriar Asta and Ender Özcan. A tensor analysis improved genetic algorithm for online bin packing. In *GECCO '15: Proceedings of the 17th Annual Conference on Genetic and Evolutionary Computation*, New York, NY, USA, 2015. ACM.

[7] Shahriar Asta, Ender Özcan, Andrew J. Parkes, and A. Şima Etaner-Uyar. Generalizing hyper-heuristics via apprenticeship learning. In *Proceedings of the 13th European Conference on Evolutionary Computation in Combinatorial Optimization*, EvoCOP'13, pages 169–178, Berlin, Heidelberg, 2013. Springer-Verlag.

[8] Edmund Burke, Graham Kendall, Jim Newall, Emma Hart, Peter Ross, and Sonia Schulenburg. Hyper-heuristics: An emerging direction in modern search technology. *International series in operations research and management science*, pages 457–474, 2003.

[9] Edmund K Burke, Michel Gendreau, Matthew Hyde, Graham Kendall, Barry McCollum, Gabriela Ochoa, Andrew J Parkes, and Sanja Petrovic. The cross-domain heuristic search challenge–an international research competition. In *Learning and Intelligent Optimization*, pages 631–634. Springer, 2011.

[10] Edmund K Burke, Michel Gendreau, Matthew Hyde, Graham Kendall, Gabriela Ochoa, Ender Özcan, and Rong Qu. Hyper-heuristics: A survey of the state of the art. *Journal of the Operational Research Society*, 64(12):1695–1724, 2013.

[11] Edmund K Burke, Matthew Hyde, Graham Kendall, Gabriela Ochoa, Ender Özcan, and John R Woodward. A classification of hyper-heuristic approaches. In *Handbook of metaheuristics*, pages 449–468. Springer, 2010.

[12] Edmund K Burke, Matthew R Hyde, Graham Kendall, and John Woodward. Automating the packing heuristic design process with genetic programming. *Evolutionary Computation*, 20(1):63–89, 2012.

[13] Konstantin Chakhlevitch and Peter Cowling. Hyperheuristics: recent developments. In *Adaptive and multilevel metaheuristics*, pages 3–29. Springer, 2008.

[14] Ching-Yuen Chan, Fan Xue, WH Ip, and CF Cheung. A hyper-heuristic inspired by pearl hunting. In *Learning and intelligent optimization*, pages 349–353. Springer, 2012.

[15] Vladimir Cherkassky and Filip M Mulier. *Learning from data: concepts, theory, and methods*. John Wiley & Sons, 2007.

[16] Wallace B Crowston, Fred Glover, Jack D Trawick, et al. Probabilistic and parametric learning combinations of local job shop scheduling rules. Technical report, DTIC Document, 1963.

[17] George B Dantzig and John H Ramser. The truck dispatching problem. *Management science*, 6(1):80–91, 1959.

[18] Luca Di Gaspero and Tommaso Urli. Evaluation of a family of reinforcement learning cross-domain optimization heuristics. In *Learning and Intelligent Optimization*, pages 384–389. Springer, 2012.

[19] Ritaban Dutta and Andrew Terhorst. Adaptive neuro-fuzzy inference system-based remote bulk soil moisture estimation: Using cosmoz cosmic ray sensor. *Sensors Journal, IEEE*, 13(6):2374–2381, 2013.

[20] Pablo Garrido and María Cristina Riff. Dvrp: a hard dynamic combinatorial optimisation problem tackled by an evolutionary hyper-heuristic. *Journal of Heuristics*, 16(6):795–834, 2010.

[21] Bruce L Golden, Subramanian Raghavan, and Edward A Wasil. *The Vehicle Routing Problem: Latest Advances and New Challenges: latest advances and new challenges*, volume 43. Springer Science & Business Media, 2008.

[22] Jonathan Gratch and Steve Chien. Adaptive problem-solving for large-scale scheduling problems: A case study. *Journal of Artificial Intelligence Research*, pages 365–396, 1996.

[23] Simon Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2nd edition, 1998.

[24] Jeff Heaton. *Introduction to neural networks with Java*. Heaton Research, Inc., 2008.

[25] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

[26] David Hull. Using statistical testing in the evaluation of retrieval experiments. In *Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 329–338. ACM, 1993.

[27] Matthew Hyde and Gabriela Ochoa. Hyflex competition instance summary. *summary of problem domains and instances used in the CHeSC*, 2011.

[28] Sotiris B Kotsiantis, I Zaharakis, and P Pintelas. Supervised machine learning: A review of classification techniques. In *Proceedings of the 2007 Conference on Emerging Artificial Intelligence Applications in Computer Engineering: Real Word AI Systems with Applications in eHealth, HCI, Information Retrieval and Pervasive Technologies*, pages 3–24, Amsterdam, The Netherlands, The Netherlands, 2007. IOS Press.

[29] Gilbert Laporte. The vehicle routing problem: An overview of exact and approximate algorithms. *European Journal of Operational Research*, 59(3):345–358, 1992.

[30] Mustafa Mısır, Katja Verbeeck, Patrick De Causmaecker, and Greet Vanden Berghe. An intelligent hyper-heuristic framework for chesc 2011. In *Learning and Intelligent OptimizatioN*, pages 461–466. Springer, 2012.

[31] Tom M Mitchell. Machine learning. wcb, 1997.

[32] Gabriela Ochoa, Matthew Hyde, Tim Curtois, Jose A Vazquez-Rodriguez, James Walker, Michel Gendreau, Graham Kendall, Barry McCollum, Andrew J Parkes, Sanja Petrovic, et al. Hyflex: A benchmark framework for cross-domain heuristic search. In *Evolutionary Computation in Combinatorial Optimization*, pages 136–147. Springer, 2012.

[33] Ender Özcan, Burak Bilgin, and Emin Erkan Korkmaz. A comprehensive analysis of hyper-heuristics. *Intelligent Data Analysis*, 12(1):3, 2008.

[34] Gisele L Pappa, Gabriela Ochoa, Matthew R Hyde, Alex A Freitas, John Woodward, and Jerry Swan. Contrasting meta-learning and hyper-heuristic research: the role of evolutionary algorithms. *Genetic Programming and Evolvable Machines*, 15(1):3–35, 2014.

[35] David Pisinger and Stefan Ropke. A general heuristic for vehicle routing problems. *Computers & operations research*, 34(8):2403–2435, 2007.

[36] Juan R Rabuñal. *Artificial neural networks in real-life applications*. IGI Global, 2005.

[37] Russell D Reed and Robert J Marks. *Neural smithing: supervised learning in feedforward artificial neural networks*. Mit Press, Cambridge, MA, USA, 1998.

[38] Peter Ross, Sonia Schulenburg, Javier G Marín-Blázquez, and Emma Hart. Hyper-heuristics: learning to combine simple heuristics in bin-packing problems. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '02, pages 942–948, San Francisco, CA, USA, 2002. Morgan Kaufmann Publishers Inc.

[39] Graham D Tattersall. Probability distribution normalisation of data applied to neural net classifiers. *Electronics Letters*, 30(1):56–57, 1994.

[40] Paolo Toth and Daniele Vigo. *The vehicle routing problem*. Society for Industrial and Applied Mathematics, 2001.

[41] James D Walker, Gabriela Ochoa, Michel Gendreau, and Edmund K Burke. Vehicle routing and adaptive iterated local search within the hyflex hyper-heuristic framework. In *Learning and Intelligent Optimization*, pages 265–276. Springer, 2012.

[42] D Randall Wilson and Tony R Martinez. The need for small learning rates on large problems. In *Neural Networks, 2001. Proceedings. IJCNN'01. International Joint Conference on*, volume 1, pages 115–119. IEEE, 2001.

[43] Ian H. Witten, Eibe Frank, and Mark A. Hall. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 3rd edition, 2011.

[44] Hongming Yang, Songping Yang, Yan Xu, Erbao Cao, Mingyong Lai, and Zhaoyang Dong. Electric vehicle route optimization considering time-of-use electricity price by learnable partheno-genetic algorithm. *Smart Grid, IEEE Transactions on*, 6(2):657–666, 2015.