Tyasnurita, Raras and Özcan, Ender and John, Robert (2017) Learning heuristic selection using a time delay neural network for open vehicle routing. In: IEEE Congress on Evolutionary Computation 2017, 5-9 June 2017, Donostia-San Sebastian, Spain.

Learning Heuristic Selection using a Time Delay Neural Network for Open Vehicle Routing

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

***Abstract*—A selection hyper-heuristic is a search method that controls a prefixed set of low-level heuristics for solving a given computationally difficult problem. This study investigates a learning-via demonstrations approach generating a selection hyper-heuristic for Open Vehicle Routing Problem (OVRP). As a chosen 'expert' hyper-heuristic is run on a small set of training problem instances, data is collected to learn from the expert regarding how to decide which low-level heuristic to select and apply to the solution in hand during the search process. In this study, a Time Delay Neural Network (TDNN) is used to extract hidden patterns within the collected data in the form of a classifier , i.e an 'apprentice' hyper-heuristic, which is then used to solve the 'unseen' problem instances. Firstly, the parameters of TDNN are tuned using Taguchi orthogonal array as a design of experiments method. Then the influence of extending and enriching the information collected from the expert and fed into TDNN is explored on the behaviour of the generated apprentice hyper-heuristic. The empirical results show that the use of distance between solutions as an additional information collected from the expert generates an apprentice which outperforms the expert algorithm on a benchmark of OVRP instances.**

I. Introduction

Many real-world optimisation problems carry high dimensionality and are resource-intensive [1]. Moreover, an exhaustive search for an optimal solution while solving such problems is not highly preferable due to the immense search space size making the search process computationally expensive. In some cases, even a 'high quality' near optimal solution could not be found. On the other side, heuristics are often preferred as alternative search methods considering that they are 'reasonably' fast providing a 'reasonably' good solution to a given problem.

Hyper-heuristics aim to automate the design of heuristic search in problem solving by controlling/selecting heuristics and applying them to the solution in hand for iterative improvement or generating and testing new heuristics built from available components [2]. The former type of methods are referred to as *selection* hyper-heuristics, while the latter one as *generation* hyper-heuristics. This paper studies a Time Delay Neural Network (TDNN) as a generation hyper-heuristic which automatically builds a selection hyper-heuristic using a data science technique based on apprenticeship learning for solving an open vehicle routing problem (OVRP), a variant of well known vehicle routing problem.

Instead of manually constructing a hyper-heuristic algorithm from scratch, automated generation of hyper-heuristics based on apprenticeship learning seeks ways of putting relevant algorithmic components together through examples. In majority of the previous studies, Genetic Programming (GP) is naturally used for automated generation of heuristics [3] applied to a range of problems including boolean satisfiability, online bin packing, load balancing in networking, routing and more. An example of learning generation hyper-heuristic can be found in Burke et al. (2010) [4], which applied genetic programming for automatically build reusable (2-D) strip packing heuristics. On the other hand, the use of GP or any other method for generation of selection hyper-heuristics or such components is a growing area of research. Sabar et al. (2015) [5] proposed a gene expression programming algorithm to automatically generate heuristics, validated on a benchmark, referred to as the Hyper-heuristic Flexible framework (HyFlex) [6], of six combinatorial optimization problem domains. In a previous study, Grammatical Evolution (GE) is used to evolve and generate the components of a Variable Neighbourhood Search (VNS) for Vehicle Routing Problem (VRP) [7]. VNS as a 'generated' hyper-heuristic switches between different neighbourhoods, each representing a low-level heuristic during the search process. The data science approaches based on apprenticeship learning for automatically creating a search procedure can also generalise well learning from the expert's behaviour even with a shorter training time as compared to the evolutionary computation techniques. Asta et al. [8] tested a k-means clustering algorithm used for generation of policies to solve an online bin packing problem. Moreover, Asta et al. [9] applied a decision tree based approach using C4.5 classifier to generate an effective selection hyper-heuristic for VRP.

This paper explores a generation hyper-heuristic automatically building a selection hyper-heuristic using a data science technique, namely a machine learning algorithm TDNN. An apprentice algorithm learns observing an expert algorithm. Hence, we make a record of heuristic selection choices of an expert algorithm for apprentice to learn with additional information alongside the change in the cost for the solution before and after the invocation of a selected heuristic for OVRP for a number of steps. The inclusion of more information in the learning process may help to improve the performance of the generated hyper-heuristic. Moreover, previous studies show that the local optima and their distances to the global optima in the solution space can be correlated [10]. Hence, in this study we explore an additional learning environment which provides

978-1-5090-4601-0/17/$31.00

the apprentice an additional information which is the distance between solutions alongside the cost difference (fitness).

TDNN is one of several technologies in machine learning [11]. A previous study observed that neural networks (Multi-layer Perceptron) generated classifiers forming a well-performing generation hyper-heuristic for VRP in the HyFlex framework [12]. TDNN investigates the behaviour from an expert selection hyper-heuristic through training how to perform heuristic selection using a set of sample instances. It generates a classifier based on a dataset which record actions from one expert algorithm at each stage of the search process. The classifier imitates the behaviour of the expert operating as a 'new' hyper-heuristic for solving unseen instances. This study focuses on learning how to perform heuristic selection, while the move acceptance is simply accepting all moves. This is the reason why we have choosen MCF-AM (Modified Choice Function - All Moves) [13] as an expert hyper-heuristic. In addition, MCF-AM is the second best algorithm for VRP instances in the Cross-Domain Heuristic Search Challenge (CHeSC). MCF-AM hyper-heuristic choose the best low-level heuristic at each iteration and accept all moves. The low-level heuristic selection is based on a score generated from three factors: the recent effectiveness of the applied heuristic, the recent effectiveness of heuristics pairs in a consecutive time, and the amount of time since the heuristic was applied.

Moreover, we use Taguchi method to ensure that the TDNN design quality is taken into consideration by defining an optimum setting of TDNN parameters in order to improve the learning performance [14]. The Taguchi method for Design of Experiment (DoE) is an effective technique to analyse the relationships between several parameters within the smallest number of possible experiments [14].

This paper is structured as follows. Section II introduces open vehicle routing problem, time delay neural network, Taguchi method for design of experiments, and learning from search landscape successively. Section III describes the proposed approach and experimental design. Section IV summarises the empirical results. Finally, section V presents the conclusions of this study.

II. Background

A. Open Vehicle Routing Problem (OVRP)

There is a trend among researchers recently to include real-life assumptions towards Vehicle Routing Problem (VRP) which generate several variants of VRP, including Open-VRP (OVRP). OVRP in real life occurs in home delivery service such as packages or newspapers, school bus routing, coal mines routing, and hazardous materials shipment [15]. The main feature of this problem, which distinguishes it from the classic VRP, is that the vehicles are not required to return to the depot. This problem is faced by companies which contract their product distribution to external couriers. The reason is because the companies do not own vehicle fleet which adequates to satisfy customer's demand. Another case is that the companies have a large number of deliveries or customer's demand varies significantly over time. In this case, even though the companies have their own vehicle fleet, they prefer combining with hired vehicles and perform open rather than closed routes so that maintenance costs do not occur.

TABLE I
OVRP INSTANCES

Instance	No.Customers	No.Vehicles	Vehicle Capacity
C1	50	5	160
C2	75	10	140
C3	100	8	200
C4	150	12	200
C5	199	16	200
C6	50	5	160
C7	75	10	140
C8	100	8	200
C9	150	12	200
C10	199	16	200
C11	120	7	200
C12	100	10	200
C13	120	7	200
C14	100	10	200

The objective of OVRP is the minimisation of the total travelling cost, which is assumed that the operating cost of an extra vehicle will always outweigh the reduction in distance [16]. Consequently, the primary objective is minimising the number of vehicle v with least distance d.

$$obj = c \times v + d \tag{1}$$

c is set to 1000 to show the level of importance of vehicle number which is much more higher than the distance.

The problem consists of finding the set of routes that satisfy the following three criteria: 1) each route originates at the depot and terminates at one of the customers; 2) partial delivery is not allowed; 3) the vehicle capacity constraint is not violated. Further details on the OVRP datasets benchmark can be seen in Table I [17].

B. Time Delay Neural Network

In a previous study, an Multi-layer Perceptron (MLP) was applied as a learning algorithm which successfully captures the expert behaviours and even improve the performance by utilising fitness only information [12]. Considering the type of data, which is previous fitness values, as a sequential or time-series, and having more information fed into the learning, we choose a Time Delay Neural Network (TDNN) as our learning algorithm. TDNN has a topology which is owned by a neural network in general with three layers: input, output, and the hidden layer which handles input manipulation through filters. TDNN is a part of a general class of dynamic networks, in which the dynamics appear only at the input layer (a static multi-layer feedforward network with a tapped delay line at the input) [18].

TDNN feature is the ability to express a relation between inputs in time. In order to represent data at different points in time, a set of delays (time lags) are added to the input. The delays are an attempt to add a temporal dimension or a memory structure to the network which make TDNN is also called as an MLP with sliding window. Machine learning has

a capability to select the best window length for the learning task [19]. By doing this, TDNN capable to hold past samples of the input signal.

C. Taguchi Method for the Design of Experiments

A Time Delay Neural Network (TDNN) has several hyper-parameters for model selection which include micro and macro level of the networks [14]. The micro-structural parameters involve the type of activation function, learning rule (learning rate, momentum, learning duration), and the selection of input representation format (number of features, number of cases). The macro-structural parameters include the selection of a suitable number of layers and neurons. Choosing the optimum parameters will help to achieve a fast convergence training speed and meet the accuracy level. The latest Taguchi applications to optimise neural network parameters are listed in a study by Sukthomnya and Tannock (2005) [20]. The number of neurons in a hidden layer was a parameter chosen by most of researchers while the other parameters are more varied. Moreover, Packianather et al. (2000) [21] reported that the parameters which were found to be most important were the number of neurons and learning rate.

In order to create an optimum TDNN structure, there are three types of design of experiments (DoE) which can be used: one-factor-at-a-time, full factorial, and fractional factorial [21]. One-factor-at-a-time is treated as a sequence of trial and error which does not systematically target a near optimal solution and full factorial is too time consuming. Therefore, fractional factorial is preferred whose one of DoE matrix is developed by Dr. Genichi Taguchi, referred to as Orthogonal Arrays (OAs) [22]. This matrix reduce the number of experiments but still obtain reasonably rich information [23]. A recent study indicates the success of Taguchi method for configuring the parameters of a memetic algorithm [24].

Orthogonal Arrays (OAs) can be viewed as plans for multi-factor experiments where the columns correspond to the factors (parameters), the columns entries correspond to the test levels of the factors, and the rows correspond to the test runs [25]. Taguchi's format for an OA has the property that the entries in the columns on the left of the matrix change less frequently than the entries on the right, with a gradual rise in changes towards the right. Moreover, the columns of all OAs are balanced in two ways: 1) the columns are balanced within themselves such that they all have an equal number of levels of the factor; 2) the columns are balanced between any two columns such that together they form an equal number of possible level combinations (pairwise orthogonal);

The Taguchi method assesses which of several varying factors have the most noticeable effect on the desired outcome. It involves the following steps [14]: (i) Specifying objective functions and identify the design factors and levels; (ii) Designing the matrix experiments by selecting an appropriate OA; (iii) Conduct the matrix experiments and analyse the results; (iv) Determining the most suitable design parameters; and (v) Performing a confirmatory experiment. The results of the Taguchi is analysed in two phases: (i) Evaluating the factorial effects (main effects) to identify the most suitable design parameters; (ii) Performing an analysis of variance (ANOVA) on the result from the average of repetitive runs. ANOVA is a statistical test to determine the percentage contribution of an individual factor. This study implements a Taguchi OA for DoE to define a minimum set of TDNN parameter-level combinations to be tested in the experiment. Taguchi estimates the average effects of each parameter on the measured response when it is changed from one level to another.

D. Learning from Search Landscape

In this study, the influence of having more information utilised by the machine learning algorithm on the performance of generated selection hyper-heuristic is explored. The distance between solutions is used as an additional source of information for machine learning. This choice originated from previous studies on search or fitness landscape analysis which has extensively been applied in the evolutionary computation field [26]. It is a valuable tool used to gain a deeper understanding of the interdependence between heuristics, their behaviour and performance. A heuristic can be considered as a guidance mechanism for traversing a landscape in order to get close to the highest peak or optimal solution. Knowledge of landscape features could help to improve the design of hyper-heuristics since it would influence the effectiveness of the heuristic search.

A solution to a Vehicle Routing Problem (VRP) is most naturally represented as a set of permutations of customers. For permutation representation, there is a large number of distance metrics in the scientific literature [27]. One important property in solving VRP is an adjacency (e.g., of customers, nodes). Therefore, adjacency-based distance is deployed in this study as in Kubiak [28].

III. Methodology

A. Proposed Apprenticeship Learning Approach

Apprenticeship learning in this study involves learning from examples provided by an expert to build a selection hyper-heuristic focusing on the heuristic selection component. Our approach operates in a train and test fashion in two phases. Figure 1 illustrates the general process flow for the proposed approach.

In the first phase, we generated a classifier for each dataset using a Time Delay Neural Network (TDNN). We named this process as the training part. We built a predictive model which was then tested by cross-validation with ten folds to evaluate the model's accuracy. Cross-validation will split the whole dataset into equal sized subsets and returns the averaged value of the prediction scores of each subset obtained on the union of all the other subsets. The learning process occurs in the training part and the output is a model which describes the dataset in a generalised form.

The dataset were collected by running an expert algorithm (MCF-AM) on four instances where every instance was chosen arbitrarily from each class of OVRP. This means that the

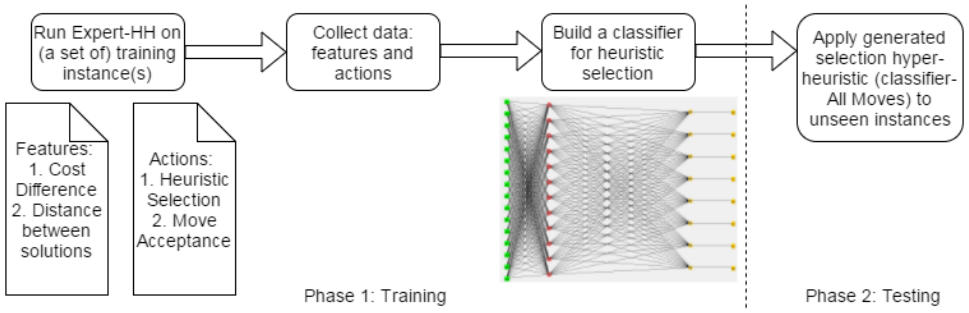

Fig. 1. Proposed apprenticeship learning approach

training instances were taken from different variants such as the number of customers, the number of vehicles, and vehicle capacity in order to represent generality. The expert algorithm accepts all solutions including equal or worsening, and therefore we only focused on the heuristic selection demonstration. Following this, one dataset was constructed and the four instances were combined together.

The dataset is an example for machine learning. The dataset consists of several search states where each one is represented by a feature vector. After running the expert algorithm, we have a dataset in which a state is defined in Eq. 2.

$$s_t = \{h_t, f_t, d_t, f_{t-1}, d_{t-1}, , f_{t-n}, d_{t-n}\} \tag{2}$$

Here, f_t is the change in evaluation function (fitness value), while d_t is the change in solution (distance metric). Both show the difference in value from current solution to candidate solution in eight previous consecutive times. Moreover, $f_t = o_t - o_{t-1}$ represents the change in the evaluation function value at iteration t, where o_t and o_{t-1} are evaluation function values achieved in iterations t and $t-1$ respectively. The action in the dataset h is nominal, which is the set of all available low-level heuristics. There are eight low-level heuristics which are categorised as mutation, ruin-recreate, and local search. The low-level heuristic details for VRP are the same as a study by Walker et al. (2012) [29].

In addition, d_t is a distance measure which counts the number of positions at which a pair of permutation nodes (adjacency-based) mismatch and thus the distance values range from 0 (both are the same) to n (the length of the sequences). Given that p_1 and p_2 are valid solutions to an OVRP instance specified, the distance between the solutions p_1 and p_2 is defined in Eq. 3 [30].

$$d(p_1, p_2) = |\{i \in \{1, ..., n\} | p_1(i) \neq p_2(i)\}| \tag{3}$$

In the second phase, we applied the model to the unseen instances. We named this process as the testing part. This phase occurred inside the TDNN-based selection hyper-heuristic framework, whose pseudo-code can be seen in Algorithm 1 [6]. There are four steps: initialise solution, select heuristic based on TDNN classifier, apply a heuristic, and accept all moves. The process stops when the time limit is reached.

Algorithm 1 TDNN-based selection hyper-heuristic

Require: problem domain to be solved
1: *numberOfHeuristics* ← number of search operators
2: *currentObjValue* ← current objective function value
3: Initialise one solution in the memory
4: **repeat**
5: h ← **TDNN Heuristic Selection**
6: *newObjValue* ← *applyHeuristic (h)*
7: **Move acceptance: All-Moves**
8: *newObjValue* ← *currentObjValue*
9: **until** $timeExpired = TRUE$

B. Experimental Design for Parameter Tuning of Time Delay Neural Network

Since using the best performing (classifier or function) is crucial with the apprenticeship learning framework, we applied an experimental design approach namely Taguchi method to obtain the best settings for the design parameters. We identified three parameters with five levels of each parameter. Testing all the combinations means having 125 settings which would result in an exhaustive experiment. However, by using the Taguchi Orthogonal Array this has been reduced to 25 settings based on L_{25} table design.

Three design parameters were considered here for optimisation: number of neurons, learning rate, and momentum. The first level for a number of neurons in the hidden layer was set to the average of the number of input neurons and output neurons. This is considered as being the minimum acceptable number of hidden neurons [21]. There were 16 input neurons (consist of 8 of f and 8 of d) and 8 output neurons (the number of search operators h). Thereby, 12 was the number of hidden neurons at the first level. Four was added to this number to represent the experimental range with a consistent increase. We limited the neuron to less than twice the size of inputs [31], set the number of epochs (learning duration) to 8192 and the number of samples or cases to 71936.

The most widely used method for training a Time Delay Neural Network (TDNN) is gradient descent [31], also known as the back-propagation algorithm. The training algorithm adjusts the weights to reduce error, with the amount of adjustment controlled by two parameters, namely learning

TABLE II
LEVEL CONFIGURATIONS DEPLOYED DURING TDNN PARAMETER TUNING

Parameters	Levels
Neurons	{12, 16, 20, 24, 28}
Learning rate	{0.01, 0.03, 0.05, 0.07, 0.09}
Momentum	{0.1, 0.3, 0.5, 0.7, 0.9}

rate and momentum. Learning rate is a scale or steps size along the error surface to correct the weight, while momentum is a weight change proportion which has been previously calculated [21]. In this study, both values cover a range commonly used in literature [32]. The values or level options for each parameter can be seen in Table II.

IV. EMPIRICAL RESULTS

Four sets of experiments are conducted to identify the best parameter setting for a Time Delay Neural Network (TDNN) and observe the influence of inclusion of additional distance information on the performance of generated hyper-heuristic. The experiments were performed on an Intel(R) Core(TM)i7 Windows 7 Enterprise (3.40 GHz) with 16 GB RAM. After the generation of selection hyper-heuristic, they are tested based on 30 trials, each terminating after 600 nominal seconds.

A. Parameter Tuning of a Time Delay Neural Network

There are 25 settings which were carried out based on the set of level combination in L_{25} Taguchi Orthogonal Array [22]. In the first set of experiment, we investigate the best parameter setting via Taguchi using three instances from 14 instances and fitness only data. The three instances represent the variants in term of number of customers. It is imperative to note that different samples generate hyper-heuristic with totally different performance hence emphasize the importance of sampling instances for parameter tuning under the proposed framework. We ranked over 25 settings for each instance according to the mean objective value (routing cost) and then achieved the average rank over 30 trials as a final score. The average effect of each parameter was computed by averaging over five configurations for each parameter. For example, the average effect of the number of neurons of 12 was calculated as averaging the score (24.99+13.29+14.80+12.58+19.07)/5 = 16.94. The detailed values of score for each settings are presented in Table III, while the average effect of each parameter at each level is summarised in the main effects plot which is provided in Figure 2. The level which has the lowest value (rank 1 is the highest) would be estimated as the best setting for each parameter from the main effects plot. Therefore, the best parameter setting for the Time Delay Neural Network (TDNN) parameters obtained from Taguchi is 24 for neurons, 0.07 for learning rate and 0.9 for momentum.

In the second set of experiment, we repeat the Taguchi experiments (the same 25 settings) using all instances and having additional information of distance along with fitness. The main effects plot is summarised in Figure 3. As illustrated, the best setting obtained via tuning on all instances is consistent with the best configuration identified using the training

TABLE III
PERFORMANCE COMPARISON BASED ON TAGUCHI L_{25} ORTHOGONAL ARRAY

Settings	Ins1	Ins2	Ins3	Score
1	25.00	24.97	25.00	24.99
2	15.73	13.57	10.57	13.29
3	14.53	15.53	14.33	14.80
4	12.47	12.17	13.10	12.58
5	10.23	23.57	23.40	19.07
6	13.33	13.30	14.03	13.56
7	13.17	13.17	7.63	11.32
8	15.23	13.90	13.03	14.06
9	7.30	10.13	2.33	6.59
10	12.63	12.10	12.93	12.56
11	15.27	14.40	13.17	14.28
12	13.73	13.43	14.27	13.81
13	2.40	20.80	14.53	12.58
14	13.00	12.17	11.87	12.34
15	15.90	13.50	13.07	14.16
16	11.37	11.67	11.47	11.50
17	12.13	11.57	12.37	12.02
18	10.33	7.57	7.90	8.60
19	10.97	8.73	9.80	9.83
20	12.47	11.87	13.47	12.60
21	12.97	12.03	12.37	12.46
22	11.70	6.83	11.07	9.87
23	13.47	8.63	14.13	12.08
24	12.70	7.17	11.40	10.42
25	12.93	8.13	12.07	11.04

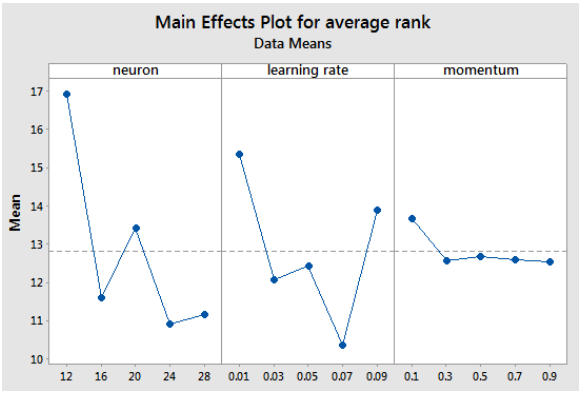

Fig. 2. Main effects plot for fitness only information based on three instances from OVRP

instances. The second experiment result shows that the best setting generated from three instances is similar with the best setting tested on all instances even if the training data contains additional information of distance along with fitness.

In the third set of experiment, we performed a confirmation or validation experiment for Taguchi based on the results from the second experiment. Since the best identified parameter setting has not been performed on the instances, a confirmation result is needed to verify the performance by comparing the best setting performance against the 25 settings. The confirmation result shows that the best obtained parameter setting (Taguchi) manages to outperforms all of the other settings. It can be seen from its lowest value in cost with a score of 3.5, followed by setting number 9 with a score of 4.7. The performance comparison of each configuration can

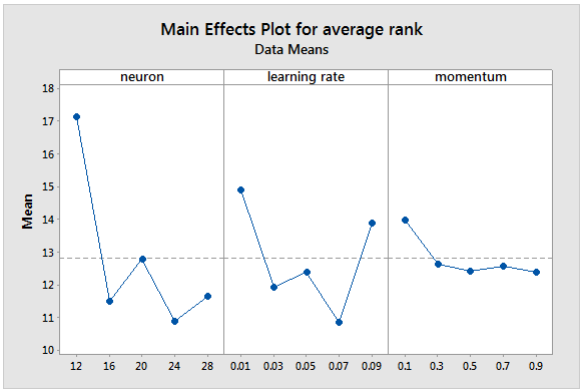

Fig. 3. Main effects plot for fitness and distance information based on fourteen instances from OVRP

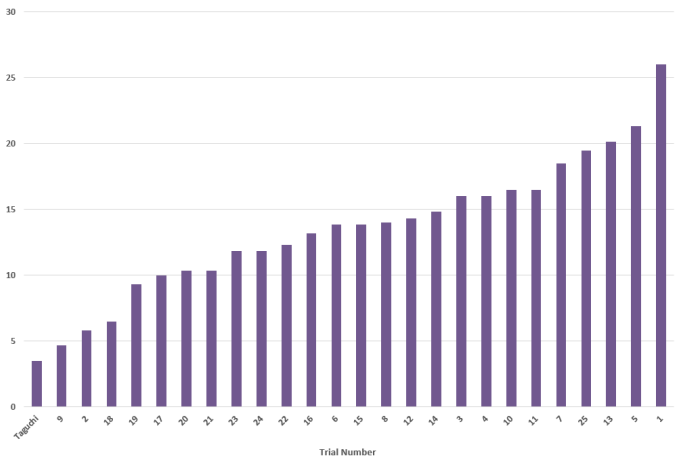

Fig. 4. Cost comparison across TDNN design parameters

be seen in Figure 4.

The learning accuracy of heuristic selection is 51.2 %. Even though the trained TDNN model of heuristic selection has nearly 50% accuracy, it is as predicted as a previous study which is using Multi-layer Perceptron (MLP) as a learning algorithm obtained accuracy of 50.3 % in the heuristic selection [12]. To the best of our knowledge, it does not suggest that the heuristic selection machine is choosing randomly among all heuristics since we have eight heuristics (eight classes in the classification). It is important to note that in order to make the heuristic selection influential to the hyper-heuristic performance, it need to be combined with move acceptance [33].

From three parameters, the number of neuron significantly contribute to the performance with the highest percentage contribution of 43.87%. Table IV shows the ANOVA analysis with percentage contribution of each parameter. This is consistent with the results described in a previous study by Packianather et al. (2000) [21], that the number of neurons in a hidden layer had a significant effect on the resulting performance for a classification type neural network application. In this study, the performance of neural network is insensitive to the momentum within the level range observed. This is shown in Figure 2 and

TABLE IV
ANOVA ANALYSIS (DF: DEGREES OF FREEDOM, SS: SUM OF SQUARES, MS: MEAN SQUARES, F: VARIANCE RATIO)

Parameters	DF	SS	MS	F	p-value	% cont.
Neurons	4	125.97	31.49	4.46	0.019	43.87
Learning rate	4	71.89	17.97	2.55	0.094	25.04
Momentum	4	4.63	1.16	0.16	0.953	1.61
Residual	12	84.66	7.06	-	-	29.48
Total	24	287.15	-	-	-	100

Figure 3 by the flat line of momentum. This is as expected, which is also consistent with the result in the previous study [21].

B. The Influence of Inclusion of Distance as a Solution Feature

The two-tailed Wilcoxon Signed-Rank test at 95 % confidence level was applied in the fourth set of experiment to evaluate the significance of the performance difference between TDNN learning using enriched knowledge with the best setting and experts. The results are summarised in Table V. Given and algorithm A, in notation $W(A)$, $\leq$ ($<$) indicates that TDNN-HH performs slightly (significantly) better than the algorithm A (within a confidence interval of 95%), while $\geq$ ($>$) indicates vice versa. Based on the mean performance, TDNN-HH is significantly better than MCF-AM on 10 out of 14 instances. We also compared the performance between learning with fitness only information to learning with enriched knowledge (fitness and distance information). TDNN-HH with enriched knowledge performs significantly better than with fitness only information (TDF) on 8 out of 14 instances.

V. CONCLUSION

In this study, we used a Time Delay Neural Network (TDNN) as a apprenticeship learning hyper-heuristic employing learning by demonstration approach to generate a selection hyper-heuristic focusing on heuristic selection for the Open-VRP. The apprentice selection hyper-heuristic is generated by observing the actions of an expert approach in operation while making decisions regarding which low level heuristic to apply at each step. Then that apprentice hyper-heuristic is tested on unseen instances. The TDNN performance is improved via the Taguchi Orthogonal Array method. Findings from a series of computational experiments include: the number of neuron in the hidden layer is the most influential factor on the hyper-heuristic performance followed by learning rate, while momentum has only minor effect.

In addition, this study investigates the learning and so generalising capability selection hyper-heuristics generated in an environment in which heuristic choice and fitness (cost) of the new solution after its application are recorded as opposed to an environment which additionally includes the distance between solutions before and after invocation of the selected heuristic. The empirical results indicate that enriching the information with a proper feature in the learning environment

TABLE V
PERFORMANCE COMPARISON OF TDNN-HH BY UTILISING ENRICHED KNOWLEDGE (1) TO MCF-AM (2) AND TDF (3) BASED ON THE MEAN OBJECTIVE VALUE OVER 30 TRIALS FOR EACH OVRP INSTANCE

	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14
(1)	5432.4	10727.2	8728.9	12934.1	18054.1	5551.6	10671.2	8945.4	14094.3	18131.6	7976.8	10699.9	11579.9	10963.9
(2)	5560.5	10944.4	8925.6	13216.4	18168.5	5543.7	10937.9	8925.4	14229.3	18578.7	8279.1	10901.8	11539.2	10935.3
$W(2)$	$<$	$<$	$<$	$<$	$<$	$\geq$	$<$	$\geq$	$<$	$<$	$<$	$<$	$\geq$	$\geq$
(3)	5534.3	10856.6	8838.5	13091.2	17502.2	5524.4	10860.8	8958.4	13135.4	17511.1	8068.7	10827.6	11417.2	10839.9
$W(3)$	$<$	$<$	$<$	$<$	$>$	$\geq$	$<$	$<$	$>$	$>$	$<$	$<$	$\geq$	$\geq$

has the potential to lead to generation of a selection hyper-heuristic with increased generalisation capability delivering a significantly better performance.

Furthermore, the successful learning from the heuristic selection behaviour motivates us to include the move acceptance learning in the future studies. Besides, we consider to investigate learning from multiple experts in order to further improve the performance.

VI. ACKNOWLEDGMENTS

Raras Tyasnurita has been sponsored by Indonesia Endowment Fund for Education Scholarship (LPDP) from the Ministry of Finance, the Republic of Indonesia.

REFERENCES

[1] Y. Tenne and C.-K. Goh, *Computational intelligence in expensive optimization problems*. Springer Science & Business Media, 2010, vol. 2.

[2] 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.

[3] E. K. Burke, M. R. Hyde, G. Kendall, G. Ochoa, E. Ozcan, and J. R. Woodward, "Exploring hyper-heuristic methodologies with genetic programming," in *Computational intelligence*. Springer, 2009, pp. 177–201.

[4] E. K. Burke, M. Hyde, G. Kendall, and J. Woodward, "A genetic programming hyper-heuristic approach for evolving 2-d strip packing heuristics," *IEEE Transactions on Evolutionary Computation*, vol. 14, no. 6, pp. 942–958, 2010.

[5] N. R. Sabar, M. Ayob, G. Kendall, and R. Qu, "Automatic design of a hyper-heuristic framework with gene expression programming for combinatorial optimization problems," *IEEE Transactions on Evolutionary Computation*, vol. 19, no. 3, pp. 309–325, 2015.

[6] G. Ochoa, M. Hyde, T. Curtois, J. A. Vazquez-Rodriguez, J. Walker, M. Gendreau, G. Kendall, B. McCollum, A. J. Parkes, S. Petrovic *et al.*, "Hyflex: A benchmark framework for cross-domain heuristic search," in *European Conference on Evolutionary Computation in Combinatorial Optimization*. Springer, 2012, pp. 136–147.

[7] J. H. Drake, N. Kililis, and E. Özcan, "Generation of vns components with grammatical evolution for vehicle routing," in *European Conference on Genetic Programming*. Springer, 2013, pp. 25–36.

[8] S. Asta, E. Özcan, A. J. Parkes, and A. Ş. Etaner-Uyar, "Generalizing hyper-heuristics via apprenticeship learning," in *European Conference on Evolutionary Computation in Combinatorial Optimization*. Springer, 2013, pp. 169–178.

[9] S. Asta and E. Özcan, "An apprenticeship learning hyper-heuristic for vehicle routing in hyflex," in *Evolving and Autonomous Learning Systems (EALS), 2014 IEEE Symposium on*. IEEE, 2014, pp. 65–72.

[10] G. Ochoa, J. A. Vázquez-Rodríguez, S. Petrovic, and E. Burke, "Dispatching rules for production scheduling: a hyper-heuristic landscape analysis," in *2009 IEEE congress on evolutionary computation*. IEEE, 2009, pp. 1873–1880.

[11] K. P. Murphy, *Machine learning: a probabilistic perspective*. MIT press, 2012.

[12] R. Tyasnurita, E. Ozcan, S. Asta, and R. John, "Improving performance of a hyper-heuristic using a multilayer perceptron for vehicle routing," in *The 15th UK Workshop on Computational Intelligence*, 2015.

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

[14] J. F. Khaw, B. Lim, and L. E. Lim, "Optimal design of neural networks using the taguchi method," *Neurocomputing*, vol. 7, no. 3, pp. 225–245, 1995.

[15] K. Braekers, K. Ramaekers, and I. Van Nieuwenhuyse, "The vehicle routing problem: State of the art classification and review," *Computers & Industrial Engineering*, 2015.

[16] F. Li, B. Golden, and E. Wasil, "The open vehicle routing problem: Algorithms, large-scale test problems, and computational results," *Computers & operations research*, vol. 34, no. 10, pp. 2918–2930, 2007.

[17] N. Christofides, "The vehicle routing problem. combinatorial optimization. christofides n., mingozzi a., toth p., sandi c.(eds) j," 1979.

[18] K. K. Htike and O. O. Khalifa, "Rainfall forecasting models using focused time-delay neural networks," in *Computer and Communication Engineering (ICCCE), 2010 International Conference on*. IEEE, 2010, pp. 1–6.

[19] N. Charaniya and S. Dudul, "Focused time delay neural network model for rainfall prediction using indian ocean dipole index," in *Computational Intelligence and Communication Networks (CICN), 2012 Fourth International Conference on*. IEEE, 2012, pp. 851–855.

[20] W. Sukthomya and J. Tannock, "The optimisation of neural network parameters using taguchis design of experiments approach: an application in manufacturing process modelling," *Neural Computing & Applications*, vol. 14, no. 4, pp. 337–344, 2005.

[21] M. Packianather, P. Drake, and H. Rowlands, "Optimizing the parameters of multilayered feedforward neural networks through taguchi design of experiments," *Quality and Reliability Engineering International*, vol. 16, no. 6, pp. 461–473, 2000.

[22] R. K. Roy, *A primer on the Taguchi method*. Society of Manufacturing Engineers, 2010.

[23] J. Ortiz-Rodríguez, M. Martínez-Blanco, and H. Vega-Carrillo, "Robust design of artificial neural networks applying the taguchi methodology and doe," in *Proceedings of the Electronics, Robotics and Automotive Mechanics Conference-Volume 02*. IEEE Computer Society, 2006, pp. 131–136.

[24] D. B. Gümüş, E. Ozcan, and J. Atkin, "An investigation of tuning a memetic algorithm for cross-domain search," in *Evolutionary Computation (CEC), 2016 IEEE Congress on*. IEEE, 2016, pp. 135–142.

[25] R. N. Kacker, E. S. Lagergren, and J. J. Filliben, "Taguchis orthogonal arrays are classical designs of experiments," *Journal of research of the National Institute of Standards and Technology*, vol. 96, no. 5, pp. 577–591, 1991.

[26] T. Schiavinotto and T. Stützle, "A review of metrics on permutations for search landscape analysis," *Computers & operations research*, vol. 34, no. 10, pp. 3143–3153, 2007.

[27] K. Sörensen, "Distance measures based on the edit distance for permutation-type representations," *Journal of Heuristics*, vol. 13, no. 1, pp. 35–47, 2007.

[28] M. Kubiak, "Distance measures and fitness-distance analysis for the capacitated vehicle routing problem," in *Metaheuristics*. Springer, 2007, pp. 345–364.

[29] J. D. Walker, G. Ochoa, M. Gendreau, and E. K. Burke, "Vehicle routing and adaptive iterated local search within the hyflex hyper-heuristic framework," in *Learning and Intelligent Optimization*. Springer, 2012, pp. 265–276.

[30] P. Merz and B. Freisleben, "Fitness landscape analysis and memetic algorithms for the quadratic assignment problem," *IEEE Transactions on Evolutionary Computation*, vol. 4, no. 4, pp. 337–352, 2000.

[31] J. Heaton, *Introduction to neural networks with Java*. Heaton Research, Inc., 2008.
[32] M. A. Bramer, *Artificial Intelligence in Theory and Practice II: IFIP 20th World Computer Congress, TC 12: IFIP AI 2008 Stream, September 7-10, 2008, Milano, Italy*. Springer, 2010, vol. 276.
[33] 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.