An Apprenticeship Learning Hyper-Heuristic for Vehicle Routing in HyFlex

Shahriar Asta*, Ender Özcan†
ASAP Research Group
School of Computer Science
University of Nottingham
Nottingham NG8 1BB, UK
*sba@cs.nott.ac.uk, †exo@cs.nott.ac.uk

***Abstract*—Apprenticeship learning occurs via observations while an expert is in action. A hyper-heuristic is a search method or a learning mechanism that controls a set of low level heuristics or combines different heuristic components to generate heuristics for solving a given computationally hard problem. In this study, we investigate into a novel apprenticeship-learning-based approach which is used to automatically generate a hyper-heuristic for vehicle routing. This approach itself can be considered as a hyper-heuristic which operates in a train and test fashion. A state-of-the-art hyper-heuristic is chosen as an expert which is the winner of a previous hyper-heuristic competition. Trained on small vehicle routing instances, the learning approach yields various classifiers, each capturing different actions that the expert hyper-heuristic performs during the search process. Those classifiers are then used to produce a hyper-heuristic which is potentially capable of generalizing the actions of the expert hyper-heuristic while solving the unseen instances. The experimental results on vehicle routing using the Hyper-heuristic Flexible (HyFlex) framework shows that the apprenticeship-learning-based hyper-heuristic delivers an outstanding performance when compared to the expert and some other previously proposed hyper-heuristics.**

I. Introduction

Hyper-heuristics are automated search methodologies that take the search process one level higher to the space of heuristics rather than dealing with solutions directly. A hyper-heuristic as a high level method combines some low level and often simpler heuristics (move operators) or heuristic components to solve a given problem. There is a growing number of studies into hyper-heuristics since the initial ideas have emerged in 1960s [1], [2], [3].

Hyper-heuristics can be classified into two groups: *selection* and *generation* hyper-heuristic [4]. The former selects a heuristic from a set of existing low level heuristics at each step of the search process. The latter, generates new heuristics from certain components. Depending on the feedback mechanism they employ, hyper-heuristics can be categorized into three groups: *on-line learning*, *off-line learning* and *no learning* (using no feedback). If a hyper-heuristic uses feedback to improve its performance while performing the search, it is an on-line learning hyper-heuristic. An off-line learning hyper-heuristic gets feedback, often using a set of training problem instances and learns prior to solving an unseen problem instance. More on hyper-heuristics can be found in [5].

In this paper, we describe a novel generation hyper-heuristic which automatically builds a generalized selection hyper-heuristic after a single training phase for solving a given class of instances. We use a machine learning approach based on *apprenticeship learning* [6] to classify the actions of an expert algorithm for heuristic selection and move acceptance based on each visited search state. The vehicle routing problem (VRP) is a well known combinatorial optimization problem that has been studied since 1959 [7]. Many different solution techniques have been proposed for many variants of this problem [8], [9], [10], [11]. The proposed approach is initially tested on the VRP domain consisting of two problem instance classes; Solomon and Gehring-Homberger using the HyFlex framework which is a software platform implemented with the main goal to support the hyper-heuristic research. The training phase takes a small duration of time on a selected problem instance as a representative of each instance class.

The paper is organized as follows. Section II provides an overview of some selected studies related to our work. An introduction to the framework and the VRP problem domain is given in Sections III and IV. This is followed by a detailed account of the apprenticeship learning framework proposed in this work in Section V. Experimental design and the results achieved through these experiments are covered in Sections VI and VII. Finally, conclusive remarks are provided in Section VIII.

II. Related Work

Machine learning has been a crucial component in the design of effective hyper-heuristics. In a previous work, [12], apprenticeship learning (AL) technique was used to generalize hyper-heuristics in the Bin-Packing domain. The AL method has a wide range of applications in control and robotics and is heavily based on Inverse Reinforcement Learning (IRL). Although we do not follow an IRL approach in our framework, our study is mainly inspired from the work in [6]. The main idea is to create an algorithm which learns what course of action to take, by simply *watching* a couple of other algorithms which perform well in various problem domains. That is, the algorithm learns the behaviour of an *expert* algorithm by constructing a dataset via recording its actions at each state of the search process. The classifier produced from this dataset is used to predict the best action at a given search state while dealing with an unseen problem instance. The AL-based approach in [12] was trained on small problem instances and was capable of generalizing the extracted knowledge to larger problem instances. The major drawback of the approach

proposed in [12] was that the definition of the search state was problem dependent. In this paper, although we test our approach on VRP, we provide a general problem domain independent state definition and investigate into whether an AL hyper-heuristic is able to perform similar (or better) than the expert algorithm on a given problem domain. Furthermore, we investigate various hyper-heuristic components from which expert knowledge can be extracted. This, in consequence, leads to an extensive modification to the approach which was proposed in [12].

There are some similar studies in which some sort of machine learning and/or expert intervention has been employed during the search. Perhaps the most relevant of these approaches is Interactive Evolutionary Algorithms (IEA) [13] where the human expert intervenes in the search process. Interactive models are proposed by IEA's, in which collaboration between a specific evolutionary algorithm and human prevents the search from getting stuck in local optima [14]. In general, Interactive Evolutionary Algorithms (IEA) [13] have been method of choice in many experiments such as Evolutionary Robotics [15] and Computer Graphics [16].

Considering this study, a very similar approach has been considered in [17]. In [17] search states of a training scheduling problem are formulated as multi-labelled feature vectors. Based on these features, a classifier is trained and constructed which performs the heuristic selection tasks. Various classifying approaches, including the one which is used in this study, have been tested and their classification accuracies are compared to each other using a 10 fold validation approach. This work is different than ours in that it uses domain dependent feature design. Moreover, only heuristic selection task has been considered and move acceptance approaches are not included in the learning process. Also, the notion of expert is not used in [17]. This is natural since the study intends to find a heuristic subset which performs well under the hyper-heuristic approach for which the classifiers were trained. This is while our approach pursues an entirely different goal which is capturing the essential strategies which are employed by expert algorithm on different domains. Nevertheless, the work is extremely valuable since it provides us with performance capabilities of various classifiers, a knowledge which will be required in our future work regarding the apprenticeship learning framework.

The automated design of search methodologies that can solve computationally difficult problems is challenging. Genetic Programming (GP) is one of the commonly used generation hyper-heuristics [18]. Some problem domains for which GP is used to construct either a heuristic or component of a solution method include job shop scheduling [19], 0/1 knapsack [20] and strip packing [21]. Drake et al. [22] used grammatical evolution to generate components of a variable neighbourhood search approach for solving VRP. A recent work in [23] provided an evolutionary approach based on gene expression programming to generate selection hyper-heuristic components for cross domain search. The approach uses a long training time. Also, it seems that the training is performed for each instance and thus the generated hyper-heuristic is not generalizable to unseen problem instances. Parameter tuning for each domain has been deemed necessary in [23] which lengthens the "training" time even further. Our method is different in that a considerably short training time is employed. Also, our proposed framework builds a generalized selection hyper-heuristic which is applicable to other unseen problem instances.

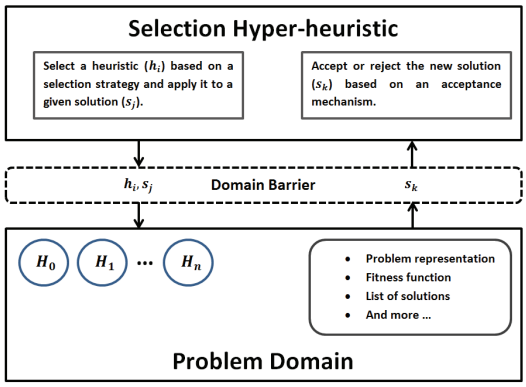

Fig. 1. A selection hyper-heuristic framework. The domain barrier restricts the information available to a high level hyper-heuristic.

III. HyFlex and First Cross-Domain Heuristic Search Challenge (CHeSC 2011)

Hyper-heuristics Flexible Framework (HyFlex) [24] is an interface to support rapid development and comparison of various hyper/meta-heuristics across various problem domains. The HyFlex platform promotes the reusability of hyper-heuristic components. In this platform, hyper-heuristics are separated from the problem domain via a *domain barrier* [25] to promote the design and development of domain-independent automated search algorithms. Hence, only the information which is not specific to a domain, such as the number of heuristics and objective value of a solution, is allowed to pass through the domain barrier up to the hyper-heuristic level from the problem domain level (Figure 1). On the other hand, pieces of domain specific information, such as, representation and objective function are kept hidden from the high level search algorithm. The separation of domain from the hyper-heuristic level in the prescribed manner is considered to be necessary to increase the level of generality of hyper-heuristics, since this way the same approach can be applied to a problem even from another domain without requiring any change.

HyFlex v1.0, implemented in Java respecting the interface definition, was the platform of choice at a recent competition referred to as the Cross-domain Heuristic Search Challenge (CHeSC 2011) [1]. The CHeSC 2011 competition aimed at determining the state-of-the-art selection hyper-heuristic judged by the median performance of the competing algorithms across thirty problem instances, five from each problem domain. Formula 1 scoring system was used to score the competing hyper-heuristics based on their median results over 31 runs for each instance. The top eight algorithms receive the scores of $10, 8, 6, 5, 4, 3, 2$ or 1, respectively, depending on their rank on a specific instance. Remaining algorithms receive a score of 0. These scores are then accumulated to produce the overall score of each algorithm on all problem instances. The number of competitors during the final round of the competition was

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

20. Moreover, a wide range of problem domains is covered in CHeSC 2011. Consequently, the results achieved in the competition along with the HyFlex v1.0 platform and the competing hyper-heuristics currently serve as a benchmark to compare the performance of novel selection hyper-heuristics.

The CHeSC 2011 problem domains include Boolean Satisfiability (SAT), One Dimensional Bin Packing (BP), Permutation Flow Shop (FS), Personnel Scheduling (PS), Travelling Salesman Problem (TSP) and Vehicle Routing Problem (VRP). Each domain provides a set of low level heuristics which are classified as mutation (MU), hill climbing (HC), ruin and re-create and crossover (XO) heuristics (operators). Each low level heuristic, depending on it's nature (i.e. whether it is a mutational or a hill climbing operator) comes with an adjustable parameter. For instance, in mutational operators, the *Intensity of Mutation* (IoM) determines the extent of changes that the selected mutation operator yields on a solution. A high mutation intensity indicates wider range of new values that the solution can take, relevant to its current value. Lower values suggest a more conservative approach where changes are less influential. As for the *Depth of Search* of hill climbing operators, this value relates to the number of steps completed by the hill climbing heuristic. Higher values indicate that hill climbing approach searches more neighbourhoods for improvement. The top three selection hyper-heuristics that generalize well across the CHeSC 2011 problem domains are AdapHH [26], VNS-TW [27] and ML [28].

The winning algorithm of CHeSC 2011, Adaptive Hyper-Heuristic (AdapHH) is a multi-phase learning hyper-heuristic [26]. AdapHH adaptively determines the subset of low-level heuristics to apply at each phase. The duration with which each heuristic is applied is also dynamically determined during the search. The algorithm accepts only improving solutions in the absence of which the algorithm refuses to accept worsening solutions until no improvements are observed within an adaptively adjusted number of iterations. The parameters of each low-level heuristic are dynamically modified via a reinforcement learning method. This is while low-level heuristics are selected based on a *quality index* measure. This measure uses few weighted performance metrics to compute the quality index for each heuristic. Among these metrics are the number of new best solutions explored, the total improvement and worsening during the run and also the current phase and finally the remaining execution time. A heuristic with a quality index less than the average of the quality indexes of all the heuristics is excluded from the selection process in the corresponding phase. Using a Tabu style memory, the number of phases in which the heuristic is consecutively excluded is maintained. Whenever this number exceeds a threshold the heuristic gets excluded permanently. AdapHH also employs a relay hybridization with which effective pairs of heuristics which are applied consecutively are identified. AdapHH algorithm ranked first in Max-SAT, BP and TSP domains. It also ranked fifth and tenth in VRP and PS domains, respectively.

IV. Vehicle Routing Problem

The Vehicle Routing Problem (VRP) is a combinatorial optimization problem with the main goal of addressing customer requirements using a fleet of vehicles [7]. The major challenge is to meet customer demands by using as few vehicles as possible while satisfying constraints such as vehicle capacity. There are many studies on different types of VRPs [8], [9], [10], [11]. In this paper, we deal with the VRP and use two classes of instances available in the HyFlex framework. These instances come with an extra constraint, namely, the time window constraint. To satisfy the time window constraint, the delivery of services to a customer must occur within a time window. Only then the produced solution would be considered valid.

TABLE I. VRP instances currently available in HyFlex framewrok. The letters R,C and RC in instance names refer to the manner with which customer locations are determined. R: Random, C Clustered, RC mixture of Random and Clustered.

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

Each vehicle starts and ends its journey from/to an initial location (the depot). The locations visited by the vehicle throughout its journey constitute the route which together with the number of vehicles used is subject to minimization efforts. Thus, the objective function can be described as:

$$f = c \times V + d \tag{1}$$

where V and d are the number of vehicles and the distance respectively. The variable c is the weight which is associated to the importance of the number of vehicles against the travelled distance. The initial value for c is set to be 1000.

Various VRP instances are available in the HyFlex framework, all of which are taken either from Solomon dataset or the Gehring-Homberger libraries. These instances (enlisted in Table I) are distinguished by few distinctive problem instance features such as number of vehicles and their capacities. The instances also differ in the way customer locations are determined. The locations are determined uniformly at random (R) or by grouping them into clusters (C) or using a mixture of both methods (RC) while generating the instances.

Ten low level heuristics are implemented for the VRP domain in HyFlex. These heuristics can be grouped into four categories: Mutational (MU), crossover (XO), Ruin and Recreate (RR) and Hill Climbing (HC). The number of heuristics in each group is $3, 2, 2$ and 3 respectively. Further documentation about the low level heuristics as well as the problem domain implementation can be found in [24].

V. The Proposed Approach

The AL approach presented here has two phases. During the first phase, say training, the expert algorithm is run on a given instance for a limited amount of time. While searching, for each search state the actions of the expert algorithm are recorded. There are several actions of the expert algorithm

that we are interested in. For a given search state, we are investigating the following actions:

- The index of the heuristic which is selected.
- The value for the parameter Depth of Search (DoS), given the chosen heuristic is a hill climber.
- The value for the parameter Intensity of Mutation (IoM), given the chosen heuristic is a mutational heuristic.
- For each heuristic index, the condition under which the solution generated by the heuristic is accepted.

A dataset for each action is constructed. For example, if there are 10 heuristics within the given problem domain, we will have 13 datasets. One dataset is used for the heuristic prediction task, one dataset for DoS and another one for IoM parameter value estimation. Moreover, for each heuristic provided in the given domain, a dataset is constructed which contains conditions under which the solution generated by that heuristic is accepted. These conditions are:

- Worsening Accepted (WA): the quality of the solution accepted by the expert algorithm is lower than that of the current solution in the memory.
- Equal Accepted (EA): The new solution offers no change in the current objective function value. Nevertheless it is accepted by the expert.

While selection hyper-heuristics accept all improving moves, their acceptance methods differ in the way they handle non-improving solutions from each heuristic. Depending on their policy, for a specific type of heuristic they sometimes accept a worsening solution while rejecting similar cases for some other heuristics. Capturing the acceptance mechanism of an expert algorithm for each heuristic separately is thus crucial in learning the overall behaviour of an expert algorithm.

In this study we represent each state of the search by a feature vector. This feature vector which defines the *search state*, can be used to determine the best course of action upon the reception of necessary feedback from the search space. The feedback available to a hyper-heuristic in the HyFlex framework is limited to the objective function value obtained after applying a selected heuristic. This is a consequent of having the domain barrier between the problem domain and the hyper-heuristic (Fig.1). Although the objective function value alone may not be very much useful when discerning between various states, the recent history of achieved objective function values might help us to produce a good level of discrimination between states.

Given that, $\delta_t = f_t - f_{t-1}$ where f_t and f_{t-1} are objective function values achieved in iterations t and $t-1$ respectively, the value δ_t describes the change in the objective function value at iteration/time t. Consequently, for all the datasets, a state is defined as $\phi_t = \{h_{previous}, \delta_t, \delta_{t-1}, \cdots, \delta_{t-n}\}$ where $h_{previous}$ is the index of the heuristic which was called before the last heuristic selection. Except the first feature in the state vector, all features are normalized by the maximum δ value achieved during the search.

In the first dataset, the action which the state feature vector refers to is $a_t \in \mathbf{h} = \{h_0, \cdots, h_k\}$ where $\mathbf{h}$ is the set of all available low level heuristics with a cardinality of k. The action defined for the second and third datasets are the values for DoS and IoM parameters (in case the action is hill climbing or mutation respectively) chosen by the expert algorithm. In the remaining datasets, each describing the acceptance method for a heuristic index, the action is defined as $a_t \in \mathbf{h} = \{EA, WA\}$ where EA and WA are described above.

Having determined the necessary state/action definitions, we can now use our expert algorithm to extract features and their corresponding actions for each search state. During training, we run the expert algorithm (π_e) once, on a single instance. While running, expert features, ϕ_t^e, are extracted for each state of the search (t). This state is then inserted into various datasets which are being built and labelled according to the action which each dataset is representing (next heuristic index, DoS and IoM value and acceptance conditions of the state). Moreover, the data entries for all the datasets are collected only if the expert algorithm accepts the solution provided by the heuristic it has selected. At the end of each run, for an expert algorithm π_e we will have $|\mathbf{h}|+3$ demonstration datasets (Eq.2).

$$\begin{aligned}
\mathcal{D}_{\pi_e}^{heu} &= \{(\phi_e^t, a_t)|\pi_e\} \quad a_t \in \mathbf{h} \\
\mathcal{D}_{\pi_e}^{DoS} &= \{(\phi_e^t, a_t)|\pi_e\} \quad a_t \in [0,1] \\
\mathcal{D}_{\pi_e}^{IoM} &= \{(\phi_e^t, a_t)|\pi_e\} \quad a_t \in [0,1] \\
\mathcal{D}_{\pi_e}^{h_k} &= \{(\phi_e^t, a_t)|\pi_e\} \quad a_t \in \{EA, WA\} \ for\ k = 0 \cdots |\mathbf{h}|
\end{aligned} \tag{2}$$

The actions in datasets $\mathcal{D}_{\pi_e}^{heu}$ and $\mathcal{D}_{\pi_e}^{h_k}$ are nominal while the actions in datasets $\mathcal{D}_{\pi_e}^{DoS}$ and $\mathcal{D}_{\pi_e}^{IoM}$ are real numbers between 0 and 1. For all the datasets with nominal actions, we employ a C4.5 algorithm to construct a decision tree for each dataset. The C4.5 algorithm [29] is a supervised learning algorithm which constructs the decision tree based on information entropy. While constructing the tree, at each node, the C4.5 algorithm decides on the feature which most effectively divides the training sample into one class or another. For $\mathcal{D}^{DoS}$ and $\mathcal{D}^{IoM}$ linear regression is used to predict the real values of the depth of search and intensity of mutation [30].

The procedure above creates an automated machine which will be referred to as the Apprenticeship Learning-based Hyper-Heuristic (ALHH) or π_g throughout the text. The ALHH, thus, consists of $|\mathbf{h}|+3$ classifiers where each of these classifiers are assigned to a different task. These tasks include Heuristic selection, DoS/IoM parameter value prediction and decision making for each heuristic on whether to acceptance or reject the recently produced solution. After constructing the classifiers, the classifier trained for heuristic selection is cross-validated on the training dataset to determine the accuracy of it's classifying abilities. This is estimated by counting the number of instances which are classified incorrectly. This number, when normalized by the total number of training instances, results in an error rate ϵ.

The second phase consists of applying what has been learned during the training to unseen instances. When applied to an unseen problem instance, for a given search state, the

heuristic prediction decision tree is consulted only with a probability equal to $1 - \epsilon$. Otherwise, a random heuristic is selected. Either way, regardless of the heuristic selection mechanism, the acceptance decision tree corresponding to the selected heuristic is consulted to determine whether the newly produced solution should be accepted or not. Also, prior to the acceptance checking, and in case the selected heuristic is hill climbing/mutational, the DoS/IoM regression machine is consulted to determine the value for the parameter DoS/IoM. This procedure continues until the maximum time allowed (T_{max}) is reached. It is worth mentioning that, except the heuristic selection classifier, the accuracy rate of cross-validation of all other classifiers are above 90%. In contrast to these classifiers, the accuracy rate achieved after cross-validating the heuristics selection classifier is around 65%.

Note that, few iterations (10) of random heuristic selection combined with IE acceptance criteria is run prior to the main search cycle to avoid infinite values for δ. Also, as was done during the training, during the search phase, all δ feature values in each search state are normalized by dividing the value to the maximum δ value which has been achieved so far during the search.

The framework described above is different from the work in [12] in that the state definition is formulated as problem domain independent. Also, the actions of an expert hyper-heuristic are of different natures. While heuristic selection procedure decides on a heuristic index, the move acceptance component of the expert decides on whether to accept or to reject the new solution provided by this heuristic. Moreover, this decision varies for different types of heuristics. Thus, in contrast to the work in [12] where a single dataset sufficed to describe all the actions of a single expert, here, different datasets are built for different actions that an expert is capable of executing. Also, due to the experimental nature of this study, and in contrast to the work in [12], only one expert (instead of multiple experts) is employed. This allows us to purely investigate the capabilities of AL scheme.

VI. Experimental Results

A. Experimental Design

The experiments are performed on an Intel i7 Windows 7 machine (3.6 GHz) with 16 GB RAM. This computer was given 438 seconds (corresponding to 600 nominal seconds on the competition machine) as the maximum time allowed per instance for each run by the benchmarking tool provided by the CHeSC 2011 organizers. The training is performed on a single instance of each class of instances once, using one tenth of the time allocated to each run by the benchmarking tool. The learned/generated selection hyper-heuristic is applied on that instance during the remaining time to complete the run plus another 30 runs on the same instance. This generated hyper-heuristic is also run 31 times on all other instances. This is to ensure that any comparison to other CHeSC 2011 competitors is fair.

In order to investigate the feasibility of employing apprenticeship learning method of [12] in a hyper-heuristic framework, in this study, we use only one expert. The expert algorithm employed in our study is the AdapHH algorithm, the winning algorithm of the CHeSC 2011 competition. A brief description of this approach is given in Section III. Throughout this paper we will refer to this algorithm using terms such as the expert, AdapHH or π_e, interchangeably.

During our experiment, rather than extending our previous work [12] to a cross-domain level, we are interested in seeing if the AL approach is able to (at least) mimic a given expert hyper-heuristic on a single problem domain. This initial study gives us further insights and illustrates the challenges that lie ahead of transferring our framework to a cross-domain level which is the next step in our research. Therefore, ALHH is once trained on a single instance of each problem class (Solomon/Gehring-Homberger) and applied on all the instances of the same class. To be precise, the arbitrarily chosen instances of 0 and 5 are used for training the ALHH on Solomon and Gehring-Homberger instance classes, respectively. Once the algorithm has been trained on instance 0 of the Solomon class, it is applied on instances $0 - 4$. A similar approach has been followed for the instances belonging to the Gehring-Homberger instances.

Applying the algorithm on the instance on which the training was performed would have an extra confirmatory role as to how successful the training has been. The stochastic nature of the expert algorithm renders such a decision plausible. Moreover, the training is only continued for a fraction of the time (10% of T_{max}) allocated to a single run on the selected instance. Hence, the training time is chosen to be substantially shorter than the maximum allowed time to avoid the dataset to grow large (which consequently would slow the ALHH down). Also, many adaptive algorithms tend to favour some heuristics/parameter values after having been adapted to the problem instance. A long training time would then create an increasingly unbalanced dataset with some actions over-represented.

VII. Results

The results of this experiment is shown in Table II. On both problem classes, ALHH (π_g) follows the expert algorithm (AdapHH or π_e) closely in terms of performance. For a number of instances the apprentice algorithm even manages to outperform the expert algorithm in terms of the average and/or the best performance. This is interesting since applying apprenticeship learning on problems in the field of robotics almost never results in an apprentice agent (robot) which outperforms its human instructor. In robotics, the expertise of the human demonstrator does not include the trial and errors which the expert has gone through while learning the task. In our case however, the data collected during the training session also includes these trial and errors made by the expert. From such a perspective, this achievement is expected to some extent. However, this is achieved despite working on datasets which include failures of the expert as well as its successes. The claim that the expert algorithm (AdapHH) has had wrong decisions is valid for there are other algorithms which performed better than AdapHH on the VRP domain (AdapHH ranked fifth in the VRP domain). This is one of the main reasons why the VRP domain has been chosen for this study. Nevertheless, ALHH outperforming its expert, AdapHH, only confirms the success of the learning mechanism. Moreover, the better performance is observed on instances other than the ones ALHH is trained on. This indicates the ability of the proposed ALHH framework

TABLE II. THE PERFORMANCE OF ALHH ON EACH CHESC 2011 INSTANCE IN THE VRP DOMAIN OVER 31 RUNS. μ AND σ ARE THE MEAN AND STANDARD DEVIATION OF OBJECTIVE FUNCTION VALUES. COMPARING ALHH TO ADAPHH, THE BOLD ENTRIES CORRESPOND TO THE ALGORITHM WITH A BETTER AVERAGE PERFORMANCE AND UNDERLINED ITALIC ENTRIES REFER TO THE ALGORITHM WHICH PRODUCES THE BEST (MIN) RESULT. ENTRIES WITH A '-' VALUE REPRESENT NON-COMPETITION INSTANCES.

		Solomon Instances					Homberger Instances				
		0	1	2	3	4	5	6	7	8	9
AdapHH (π_e)	μ	5093.4	**20656.9**	13388.0	**5321.9**	14293.0	**146791.0**	62809.4	161638.5	153164.3	**147360.5**
	min	4230.2	*20651.6*	13296.9	*5275.5*	14270.7	144040.9	58521.6	*160074.4*	*146584.7*	*145139.3*
	median	5125.7	20655.4	13349.9	5320.8	14291.1	146906.7	61985.8	161596.3	153083.7	147550.3
	σ	161.7	4.2	183.8	24.5	13.9	1369.0	4609.4	982.0	1841.3	1047.6
Apprentice (π_g)	μ	**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
	σ	394.2	340.9	310.9	29.4	481.3	1780.5	790.0	842.7	1663.1	956.8
P-Hunter [31]	min	-	20650.8	12263.0	-	-	143663.9	61139.3	-	-	146472.9
	median	-	20650.8	12290.0	-	-	146944.4	64717.8	-	-	148659.0
AdOr-ILS [32]	μ	5281.7	21291.9	13605.0	6564.4	14280.8	155305.5	77302.7	163177.7	158941.9	149447.7
	σ	334.614	482.56	451.64	554.77	319.54	6154.24	3384.83	2100.09	2460.71	1500.9

in generalizing the actions of the expert to unseen problems of different sizes, a promising sign encouraging us to research on cross-domain capabilities of the proposed framework.

The success of the apprenticeship learning approach in capturing the strategies used by the expert is further demonstrated in Fig.2. In most cases the performances of both algorithms are very close. This claim is verified through applying the Wilcoxon signed rank test which determines how close the performance of the two algorithms on 31 runs of each instance are. The results of this test is shown in Table III. On 3 out of 5 Solomon instances the performance of the two algorithms have no statistically significant difference. This is also the case on 4 out of 5 Homberger instances. On 2 Solomon instances AdapHH performs significantly better than ALHH. On the other hand, on one Homberger instance ALHH outperforms the expert in a statistically significance manner. On equal number of instances (3 out of 5 instances) on each problem class, ALHH performs either slightly or significantly better than the expert algorithm. In the overall, ALHH performs similar to the expert algorithm on most of instances regardless of the problem class in a satisfying fashion. This also indicates that transferring the apprenticeship learning framework to a cross domain level would be straightforward, a claim which is yet to be verified in our future works.

When the performance of ALHH is compared to the competing algorithms of the CHeSC 2011 competitions, it is interestingly ranked first with a score of 30 (Fig.3, Table II). This shows that on majority of instances, the apprentice hyper-heuristic maintained a good performance, even better than the expert algorithm which ranked fifth on this domain. This is despite the fact that, on 3 instances, the median performance of ALHH (based on which ranking is done in the CHeSC 2011 competition) is worse than the P-Hunter[31] algorithm which was ranked first in this domain. While P-Hunter secures the first rank on instances 1 and 2, it fails to perform in a similar fashion on other instances. It is well outperformed on instances 6 and 9 by ALHH (and some other competing algorithms). This comparison shows that the apprentice algorithm has a stable behaviour regarding various instances of the problem domains it is exposed to.

The ALHH is also compared to another approach which is proposed recently (Table II). The approach in [32], proposes an algorithm which is based on Iterated Local Search (ILS). Out of three variants proposed by the authors in [32], the variant named AdOr-ILS is shown to have a superior performance on 9 out of 10 VRP instances in HyFlex. Thus, AdOr-ILS is chosen for further comparison in which it is outperformed substantially by the ALHH on all the instances.

Finally, regarding the hyper-heuristic proposed in [23], it makes use of a long training time as well as an extensive tuning process. This is in contrast to ALHH in which the training time is very short and considered (for a fair comparison) while evaluating the performance of the algorithm. Thus, comparing ALHH (and those approaches in Table II) to the method in [23] would not be fair and therefore this performance comparison has been discarded.

TABLE III. AVERAGE PERFORMANCE COMPARISON OF ALHH (π_g) TO ADAPHH (π_e) ON TWO PROBLEM CLASSES IN THE VRP DOMAIN. WILCOXON SIGNED RANK TEST IS PERFORMED AS A STATISTICAL TEST ON THE OBJECTIVE FUNCTION VALUES OBTAINED OVER 31 RUNS FROM BOTH ALGORITHMS. $\geq$ ($>$) DENOTES THAT ALHH (π_g) PERFORMS SLIGHTLY (SIGNIFICANTLY) BETTER THAN THE COMPARED ALGORITHM (WITHIN A CONFIDENCE INTERVAL OF 95%), WHILE $\leq$ ($<$) INDICATES VICE VERSA. THE LAST COLUMN SHOWS THE NUMBER OF INSTANCES FOR WHICH THE ALGORITHM ON EACH SIDE OF "/" HAS PERFORMED BETTER (EITHER SLIGHTLY OR SIGNIFICANTLY).

	Instances					π_g/π_e
Solomon	0	1	2	3	4	
	$\geq$	$<$	$\geq$	$<$	$\geq$	3/2
Homberger	5	6	7	8	9	
	$\leq$	$>$	$\geq$	$\geq$	$\leq$	3/2

VIII. CONCLUSIONS

In this study, an apprenticeship learning based method is investigated for automatically designing a search methodology. The proposed approach observes various actions of a state-of-the-art hyper-heuristic as an expert for vehicle routing using the HyFlex framework. Various classifiers are constructed representing different components of a selection hyper-heuristic, such as, the acceptance method for each low level heuristic and heuristic selection method. The automated search method which builds the selection hyper-heuristic via classifiers is itself a generation hyper-heuristic.

At the end, the generated hyper-heuristic turns out to be indeed capable of successfully generalizing the actions of the expert while solving the unseen problem instances. It outperforms not only the expert, but also some other previously proposed selection hyper-heuristics in many occasions on the

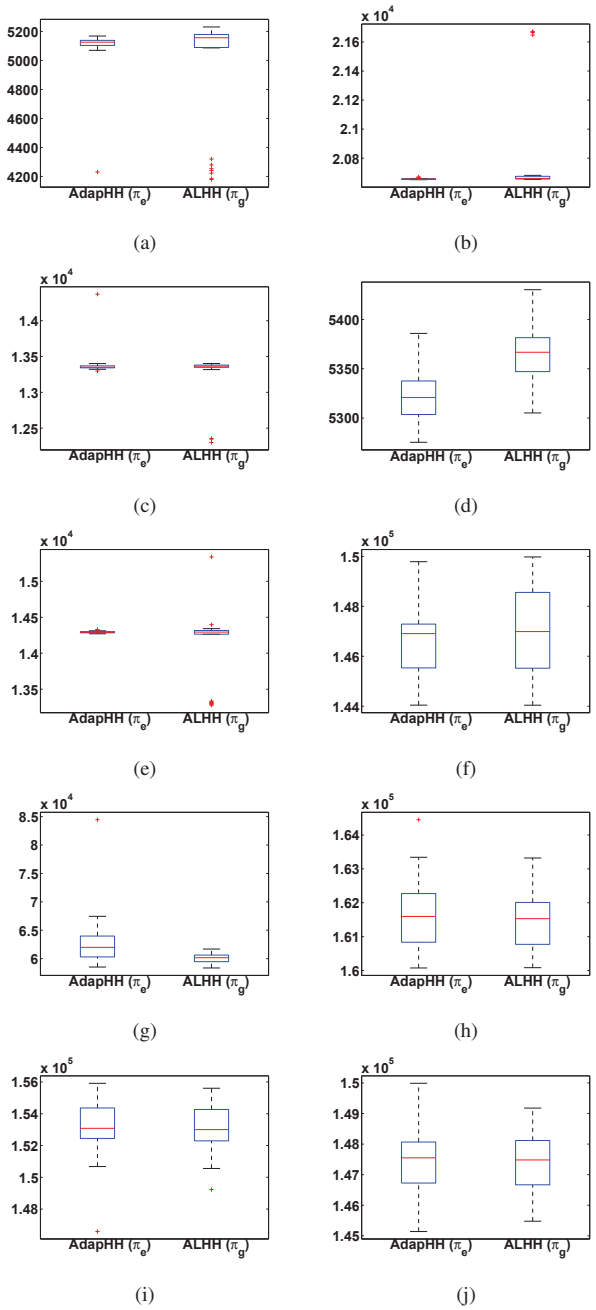

Fig. 2. Comparison of ALHH (π_g) with the expert algorithm (AdapHH) on the instances of the VRP domain.

HyFlex VRP instances. Achieving such promising results, exceedingly encourages us to extend our work and take it to a cross-domain level, where potentially training consists of few instances from each problem domain and testing is performed on all available instances in HyFlex.

REFERENCES

[1] W. B. Crowston, F. Glover, G. L. Thompson, and J. D. Trawick, "Probabilistic and parametric learning combinations of local job shop scheduling rules," *ONR Research memorandum, GSIA, Carnegie Mellon University, Pittsburgh*, no. 117, 1963.

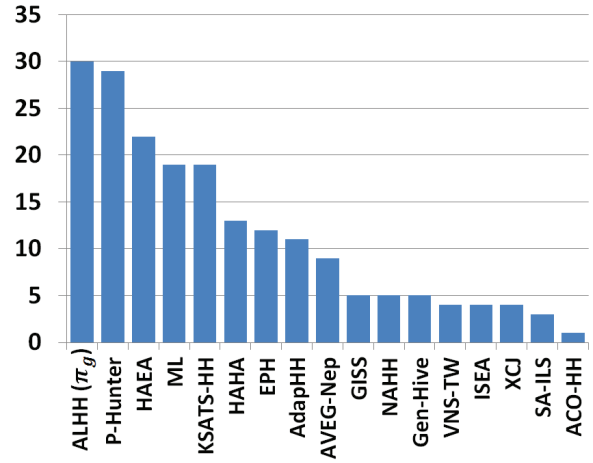

Fig. 3. Ranking of ALHH among CHeSC 2011 competitors. Algorithms with a score of 0 are not displayed here.

[2] H. Fisher and G. L. Thompson, "Probabilistic learning combinations of local job-shop scheduling rules," in *Industrial Scheduling*. Prentice-Hall, 1963, pp. 225–251.

[3] P. Cowling, G. Kendall, and E. Soubeiga, "A hyperheuristic approach to scheduling a sales summit," in *Selected Papers of the Third International Conference on the Practice And Theory of Automated Timetabling, PATAT 2000*, ser. Lecture Notes in Computer Science. Konstanz, Germany: Springer, August 2000, pp. 176–190.

[4] E. K. Burke, M. Hyde, G. Kendall, G. Ochoa, E. Özcan, and J. Woodward, *Handbook of Metaheuristics*, ser. International Series in Operations Research & Management Science. Springer, 2009, ch. A Classification of Hyper-heuristic Approaches, pp. 449–468.

[5] E. K. Burke, M. Gendreau, M. Hyde, G. Kendall, G. Ochoa, E. Ozcan, and R. Qu, "Hyper-heuristics: a survey of the state of the art," *J Oper Res Soc*, vol. 64, no. 12, pp. 1695–1724, Dec 2013. [Online]. Available: http://dx.doi.org/10.1057/jors.2013.71

[6] P. Abbeel and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning," in *Proceedings of the twenty-first international conference on Machine learning*, ser. ICML '04. New York, NY, USA: ACM, 2004, pp. 1–8.

[7] G. B. Dantzig and J. H. Ramser, "The truck dispatching problem," *Management Science*, vol. 6, no. 1, pp. 80–91, 1959.

[8] P. Toth and D. Vigo, Eds., *The Vehicle Routing Problem*. Society for Industrial and Applied Mathematics, 2002.

[9] B. Golden, S. Raghavan, and E. Wasil, Eds., *The Vehicle Routing Problem: Latest Advances and New Challenges*, ser. Operations Research/Computer Science Interfaces. Springer US, 2008, vol. 43.

[10] V. Pillac, M. Gendreau, C. Gueret, and A. L. Medaglia, "A review of dynamic vehicle routing problems," *European Journal of Operational Research*, vol. 225, no. 1, pp. 1 – 11, 2013.

[11] C. Lin, K. Choy, G. Ho, S. Chung, and H. Lam, "Survey of green vehicle routing problem: Past and future trends," *Expert Systems with Applications*, vol. 41, no. 4, Part 1, pp. 1118 – 1138, 2014.

[12] S. Asta, E. Özcan, A. J. Parkes, and A. c. Etaner-Uyar, "Generalizing hyper-heuristics via apprenticeship learning," in *Proceedings of the 13th European Conference on Evolutionary Computation in Combinatorial Optimization*, ser. EvoCOP'13. Berlin, Heidelberg: Springer-Verlag, 2013, pp. 169–178.

[13] H. Takagi, "Interactive evolutionary computation: fusion of the capabilities of ec optimization and human evaluation," *Proceedings of the IEEE*, vol. 89, no. 9, pp. 1275–1296, Sep 2001.

[14] S. E. Celis, G. S. Hornby, and J. C. Bongard, "Avoiding local optima with user demonstrations and low-level control," in *IEEE Congress on Evolutionary Computation*, 2013, pp. 3403–3410.

[15] J. C. Bongard and G. S. Hornby, "Combining fitness-based search and user modeling in evolutionary robotics," in *GECCO*, 2013, pp. 159–166.

[16] K. Sims, "Artificial evolution for computer graphics," *SIGGRAPH Comput. Graph.*, vol. 25, no. 4, pp. 319–328, Jul. 1991.

[17] F. Thabtah and P. Cowling, "Mining the data from a hyperheuristic approach using associative classification," *Expert Systems with Applications*, vol. 34, no. 2, pp. 1093 – 1101, 2008.

[18] E. K. Burke, M. Hyde, G. Kendall, G. Ochoa, E. Özcan, and J. Woodward, "Exploring hyper-heuristic methodologies with genetic programming," in *Computational Intelligence*, ser. Intelligent Systems Reference Library, C. L. Mumford and L. C. Jain, Eds. Springer Berlin Heidelberg, 2009, vol. 1, pp. 177–201.

[19] C. D. Geiger, R. Uzsoy, and H. Aytug, "Rapid modeling and discovery of priority dispatching rules: An autonomous learning approach," *J. of Scheduling*, vol. 9, no. 1, pp. 7–34, Feb. 2006.

[20] R. Kumar, A. H. Joshi, K. K. Banka, and P. I. Rockett, "Evolution of hyperheuristics for the biobjective 0/1 knapsack problem by multiobjective genetic programming," in *Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation*, ser. GECCO '08. New York, NY, USA: ACM, 2008, pp. 1227–1234.

[21] E. K. Burke, M. Hyde, G. Kendall, and J. Woodward, "A genetic programming hyper-heuristic approach for evolving 2-d strip packing heuristics," *Trans. Evol. Comp*, vol. 14, no. 6, pp. 942–958, Dec. 2010.

[22] J. H. Drake, N. Kililis, and E. Özcan, "Generation of vns components with grammatical evolution for vehicle routing," in *Proceedings of the 16th European Conference on Genetic Programming*, ser. EuroGP'13. Berlin, Heidelberg: Springer-Verlag, 2013, pp. 25–36.

[23] N. Sabar, M. Ayob, G. Kendall, and R. Qu, "The automatic design of hyper-heuristic framework with gene expression programming for combinatorial optimization problems," *IEEE Transactions on Evolutionary Computation*, vol. PP, no. 99, 2014.

[24] G. Ochoa, M. Hyde, T. Curtois, J. Vazquez-Rodriguez, J. Walker, M. Gendreau, G. Kendall, B. McCollum, A. Parkes, S. Petrovic, and E. Burke, "Hyflex: A benchmark framework for cross-domain heuristic search," in *European Conference on Evolutionary Computation in Combinatorial Optimisation, EvoCOP '12.*, ser. LNCS, J.-K. Hao and M. Middendorf, Eds., vol. 7245. Heidelberg: Springer, 2012, pp. 136–147.

[25] P. Cowling, G. Kendall, and L. Han, "An investigation of a hyperheuristic genetic algorithm applied to a trainer scheduling problem," in *Evolutionary Computation, 2002. CEC '02. Proceedings of the 2002 Congress on*, vol. 2, 2002, pp. 1185–1190.

[26] M. Mısır, K. Verbeeck, P. De Causmaecker, and G. V. Berghe, "An intelligent hyper-heuristic framework for chesc 2011," in *Learning and Intelligent Optimization*. Springer, 2012, pp. 461–466.

[27] P.-C. Hsiao, T.-C. Chiang, and L.-C. Fu, "A vns-based hyper-heuristic with adaptive computational budget of local search," in *Evolutionary Computation (CEC), 2012 IEEE Congress on*, 2012, pp. 1–8.

[28] M. Larose, "A hyper-heuristic for the chesc 2011," in *LION6*, 2011.

[29] J. R. Quinlan, *C4.5: Programs for Machine Learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1993.

[30] E. Alpaydin, *Introduction to Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2004.

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

[32] 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 *LION*, 2012, pp. 265–276.