Design and Analysis of an Evolutionary Selection Hyper-heuristic Framework

Mustafa Mısır, Katja Verbeeck, Patrick De Causmaecker, and Greet Vanden Berghe

***Abstract*—In the present study, an evolutionary single-point search selection hyper-heuristic was developed. The development was carried out by considering possible characteristics based on a high-level library, i.e. HyFlex. The aim was to develop multiple online adaptive procedures for a selection hyper-heuristic and simultaneously coordinating them. The developed approach was expected to evolve itself for different search environments with no human intervention. The design process was initiated by identifying the requirements for solving different combinatorial optimisation problems using distinct heuristic sets. The information gathered was used to develop a number of evolvable components to handle these requirements for a general solver. The developed components were combined using dynamic decision mechanisms to provide proficient cooperation between them. This approach has been applied to a set of instances from six problem domains present in HyFlex. For each problem domain, a heuristic set including four types of heuristics are provided by the developers of HyFlex: ruin-recreate heuristics, local search algorithms, mutation and crossover operators. The experimental results indicated that the approach is effective in solving the target instances from distinct problem domains. The proposed hyper-heuristic won the first international Cross Domain Heuristic Search Challenge 2011 against 19 high-level algorithms developed by the other academic competitors.**

***Index Terms*—Hyper-heuristic, general solver, HyFlex, CHeSC**

I. Introduction

SELECTION hyper-heuristics are competent high-level search strategies for solving different instances from different problem domains in a problem independent manner. Therefore, they are understood as general problem solvers delivering high quality solutions across distinct problem domains. A traditional selection hyper-heuristic consists of two main sub-mechanisms to manage a set of low-level heuristics directly operating on the solution space. The first sub-mechanism, i.e. *heuristic selection*, determines which heuristic should be applied at each decision step. Thus, a selection hyper-heuristic can also be considered as an evolutionary approach for generating a low-level heuristic list in which each heuristic is successively applied. A *move acceptance* strategy is employed to evaluate the acceptability of the generated solutions by these applied heuristics.

This study investigates particular components to build an evolutionary hyper-heuristic for the purpose of generality. First of all, the most relevant features reflecting the performance of a low-level heuristic were identified. Then, a number of components was developed relying on these features and possible changes that can be encountered during a run on the search space. The main components are listed as *heuristic subset selection*, *heuristic selection*, *heuristic hybridisation*, *move acceptance* and *re-initialisation*. The subset selection operation was handled by an adaptive dynamic heuristic set (ADHS) strategy that explores the best heuristic subsets for different parts of the search. The selection process among these heuristics was guided by a probability vector that is updated based on the improvement capabilities and the speed of the heuristics. For using the heuristic set more effectively, a pairwise heuristic hybridisation approach was introduced. In addition, an adaptive threshold accepting strategy, i.e. adaptive iteration limited list-based threshold accepting (AILLA), was devised as the move acceptance mechanism. In the case of discovering good quality solutions immediately, the solution re-initialisation is triggered by the move acceptance mechanism to reach higher quality solutions faster. Fig. 1 shows the whole working process of the designed hyper-heuristic. For empirically examining the generality level of the proposed hyper-heuristic, ADHS-AILLA, a series of experiments over the instances from six problem domains provided by a high-level search library, HyFlex [1], have been conducted. Prior to these experiments, the designed hyper-heuristic competed against 19 other high-level, problem-independent algorithms in the first International Cross Domain Heuristic Search Challenge (CHeSC 2011). ADHS-AILLA beats all these competing algorithms with respect to their overall performance across six problem domains by a large score difference. Furthermore, the executed experiments indicated that ADHS-AILLA outperforms different hyper-heuristics from the literature for different running time limits.

The remainder of the paper will treat hyper-heuristics in Section 2. Then, the hyper-heuristic framework for the generality purpose with all the algorithmic details is demonstrated in Section 3. After that, the HyFlex software environment used for the experiments is explained in Section 4. Section 5 provides computational results as well as an analysis of the performance and the behaviour of ADHS-AILLA. Section 6 concludes the paper and discusses possible future research directions for further improvements.

II. Hyper-heuristics

A. Literature Review

Hyper-heuristics have been extensively applied to different instances from various problem domains such as schedul-

M. Mısır, K. Verbeeck, and G. Vanden Berghe with CODeS Research Group, KAHO Sint-Lieven, K.U.Leuven Campus Kortrijk, Belgium (e-mail: mustafa.misir@kahosl.be; katja.verbeeck@kahosl.be; greet.vandenberghe@kahosl.be).

P. De Causmaecker with CODeS Research Group, Department of Computer Science, K.U.Leuven Campus Kortrijk, Belgium (e-mail: patrick.decausmaecker@kuleuven-kortrijk.be).

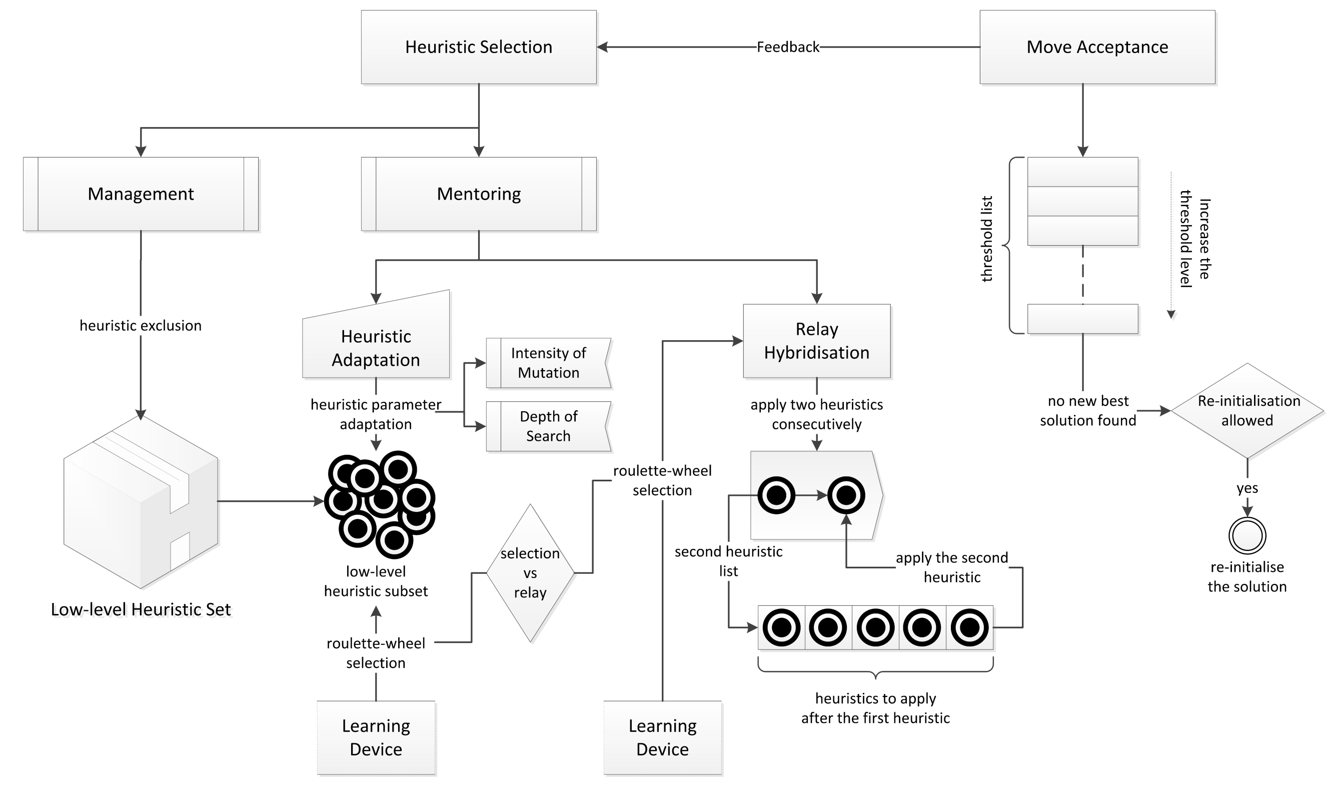

Fig. 1. An evolutionary hyper-heuristic framework

ing [2]–[5], timetabling [6]–[8], routing [9]–[11] and cutting/packing [12], [13]. The published results on these studies indicated that hyper-heuristics are easy-to-use yet powerful methods. Most of these hyper-heuristics utilise evolutionary algorithms and various learning devices to deliver more intelligent and adaptive approaches through a population-based or a single-point search (Table I). A genetic algorithm (GA) has been used to explore effective heuristic sequences encoded as chromosomes via applying one-point crossover and a simple mutation operator [14]. Heuristics in a chromosome are consecutively applied to construct a solution. The GA-based hyper-heuristic was improved by dynamically changing the size of the chromosomes through adding effective heuristics whilst removing the poor ones in [15]. [16] added a tabu function that prevents the heuristics delivering poor performance to be called for a number of generations. An offline method for determining the relations between problem states and low-level heuristics, i.e. a messy-GA hyper-heuristic, was applied to the 1D bin packing problem by [17] In this approach, each chromosome contains blocks and each block indicates the proportion of the items to be packed as well as the heuristic responsible for the instances similar to this block. In [18], a GA-based hyper-heuristic was proposed for solving 2D-regular cutting stock problem. Chromosomes mainly contain groups of parameters representing a selection heuristic, a placement heuristic, a rotation procedure and the number of pieces that will be used by these three methods in the population. A similar representation strategy as [17] for finding effective selection and placement heuristics concerning a set of problem state features was employed to solve cutting stock [12], [19] and bin packing [20], [21] problems. Ant colony optimisation (ACO) is another population-based evolutionary algorithm used within selection hyper-heuristics. An ACO-based hyper-heuristic considering ants as hyper-heuristic agents and vertices as low-level heuristics was designed for solving the project presentation scheduling problem in [22]. The idea was to determine useful heuristics (visibility) and effective heuristic pairs (pheromone) based on their performance related measures which are updated online. For each cycle, the best solution found among all the ants at the previous cycle is used as their current solution and all the ants start from the vertex whose heuristic found this solution. A similar approach was utilised to solve the travelling tournament problem [23]. Another ACO-based hyper-heuristic was applied to routing and wavelength assignment problem [24]. [25] studied on an ACO-based hyper-heuristic with space reduction for solving the p-median problem through consecutively applying diversifying and intensifying heuristics. Particle swarm optimisation (PSO) was used to design two distributed asynchronous frameworks through running low-level heuristics as agents while also running hyper-heuristics with simple random as agents [26]. In [27], a PSO-based hyper-heuristic was applied to the exam timetabling problem using particles representing partial solutions and heuristic sequences. Another population-based strategy, i.e. scatter search, was used within two hyper-heuristics to determine good chains of priority rules for the mixed-model assembly line problem by [28].

Genetic programming (GP) was employed to automatically generate heuristics for the online 1D bin packing problem in [29], [30]. Trees in the population represent the methods that decide about placing an item to a bin. Each tree is composed of some arithmetic functions and a set of terminals representing the characteristics and states of the bins. A similar approach was applied to the 3D knapsack problem [31], 2D strip packing problem [13] and single machine production scheduling problem for building dispatching rules [32]. A GP-based hyper-heuristic evolving heuristics as grammars consisting of elements extracted from widely used local search solvers for the SAT problem was proposed in [33]. [34] devised an approach, named Inc*, which repeatedly adds or removes SAT clauses to or from a clause list and solves the updated problem using a local search solver. The number of clauses to be added or removed is determined by a GP method. This method uses simple mathematical functions and terminals from a target SAT instance for constructing effective trees.

Inc* was modified [35] by adding the grammar-based heuristic generation method for solving the partial instances in [33]. Another GP-based hyper-heuristic was proposed to create heuristics for timetabling problems [36], [37]. A grammar was designed using the elements of some exam selection and slot selection heuristics from the literature. In [38], genetic programming was employed to construct trees for adaptively applying existing heuristics based on the current state of a search. The function set includes certain conditional elements as well as some acceptance mechanisms and the terminal set is composed of the low-level heuristics.

Aside from these population-based evolutionary hyper-heuristics, other types of learning strategies have been used for choosing heuristics. The choice function was employed for selecting heuristics with respect to their single and pairwise performance [39]. A reinforcement learning based selection mechanism using various scoring strategies was studied [40], [41]. The idea is to update the score of each heuristic relying on its performance and to conduct the selection process using these scores. In [42], [43], the authors studied a learning automaton maintaining a probability vector for heuristic selection. Together with a simple scoring strategy, a simple tabu mechanism that eliminates worse performing heuristics for a period of time was developed in [4]. The heuristics were selected from a dynamically determined elite heuristic subsets for different regions of the search space in [43], [44]. A learning automaton was additionally utilised for choosing heuristics from these subsets [43]. As an offline approach, case-based reasoning was used as a heuristic selection strategy under hyper-heuristics in [45]. The motivation is to determine similarities between heuristics and a set of previously solved instances (case base) with respect to the most relevant problem related features to choose the best heuristics for the target problem instances. Fuzzy systems were studied to determine the scheduling order of each element using multiple ordering heuristics and course timetabling problems respectively by [46]. In [47], a 3-layer neural network and a logistic regression model were developed to predict the quality of the solutions generated by a graph based hyper-heuristic [48] on the exam timetabling problem. The motivation here is to fasten the search process by avoiding evaluating the generated or constructed solutions each time. In [49], another neural network based hyper-heuristic was suggested to determine a correlation between the constraint satisfaction problem instances with certain features and the heuristics to solve these instances. A solution for a CSP instance is incrementally constructed and a backpropagation neural network with a sigmodial transference function decides which ordering heuristic to apply at each decision step with reference to the current characteristics of the subproblem. Most of these studies cover instances from only one problem domain and this limits to measure the quality of these hyper-heuristics from a generality perspective. A recently proposed hyper-heuristic software framework, HyFlex [1], finally draws the attention of hyper-heuristic researchers to generality. For a review, refer to [50].

TABLE I
EVOLUTIONARY ALGORITHMS AND LEARNING METHODS USED IN HYPER-HEURISTICS

Approach	References
Genetic Algorithms	[12], [14]–[19]
Ant Colony Optimisation	[22]–[25], [51]–[53]
Particle Swarm Optimisation	[26], [27]
Scatter Search	[28]
Genetic Programming	[13], [29]–[38]
Reinforcement Learning	[10], [40]–[43], [54]
Case-based Reasoning	[45]
Learning Classifier Systems	[55]–[58]
Fuzzy Systems	[46]
Neural Networks	[47], [49]

B. Generality Level

Generality is considered as the key concept and underlying motivation behind hyper-heuristics. The idea is to design a high-level approach that is intelligent enough to determine the strengths and weaknesses of a set of low-level heuristics [50] so that it can be successfully applied to several problems with different heuristic sets. In other words, a selection hyper-heuristic's job is to identify the characteristics of the heuristics and to match them with the problem states. A very naive hyper-heuristic could say that a specific heuristic should be utilised during the first half of the search and after that, another heuristic should take the lead to discover better solutions for a problem instance. Based on this example, the main question for the hyper-heuristic research is how far a hyper-heuristic can be made more specific in relation to heuristic-problem state matchings. A perfect selection hyper-heuristic would systematically select the best heuristic in each application. The primary limitation to this perfect case is the problem-independent nature of the hyper-heuristics. This means that, it is not possible to provide certain problem specific information to a hyper-heuristic. For this reason, a hyper-heuristic should consist of a number of learning devices to explore the search space of heuristics and gather the required knowledge to lift its generality level.

C. Search Space

Intelligent search and optimisation algorithms use the features and dynamic characteristics of the search space corresponding with the target problem. They concentrate on certain rules and structures to direct the search process as efficiently as possible. Furthermore, for finding high quality solutions, some

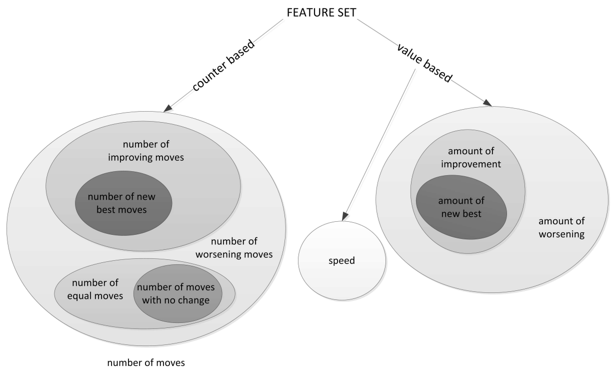

Fig. 2. The feature set consisting of individual features regarding the heuristic search space

problem specific information is encoded in the algorithms. Even if such an approach is expected to be successful on the target problem instances, it is not sure that it will be successful for solving other instances of the same problem. Supposing that the representation is not an issue, applying the same method to the instances from another domain may perform even worse. In brief, there is a trade-off between the performance of an algorithm and its dependency level to the problem instances. Hyper-heuristics function at a higher level without direct knowledge about the solution space. Differently from an algorithm aiming at solving a problem instance, hyper-heuristics focus on managing a set of existing algorithms called the low-level heuristics. Therefore, the performance of a hyper-heuristic depends on the quality of these algorithms.

The size of the heuristic search space is limited by the number of available heuristics. However, it is possible to extend the set size by increasing the number of heuristic options based on the same set of available heuristics. For instance, new heuristics can be constructed combining multiple heuristics in consecutive calls. Furthermore, the parametric heuristics can be differentiated with different parameter values. Such options can facilitate flexibility for performance improvement, but at the same time, require additional learning components.

1) Feature Set: The number of features regarding low-level heuristics can be extracted from two main classes, namely *value based* features and *counter based* features. Fig. 2 illustrates the individual features for a heuristic set together with their relations. These features are the major elements to assess a heuristic's performance in a problem-independent way. Especially useful information for the performance of a heuristic is obtained from value based features. However, using each of them separately might be misleading. Thus, combining different features is required to deliver high quality performance measures for the reliability level of the utilised components with learning [59].

III. A Selection Hyper-heuristic Framework

An evolutionary selection hyper-heuristic was designed for solving the given problem instances within HyFlex. In this approach, the selection process contains two main components, namely *management* and *mentoring*. The management operation deals with the selection of heuristics. The mentoring is available to manipulate or instruct the parametric heuristics. A simple hybridisation method that applies two heuristics consecutively is additionally considered under mentoring. For judgement purposes concerning the explored solutions by the selected heuristics, an adaptive *move acceptance* mechanism is utilised. The proposed framework is displayed in Fig. 1. The details of all the developed mechanisms are presented in the following sections.

A. Design Principles

In most of the hyper-heuristic studies, the approach developed has been applied to a set of instances from one problem domain using one heuristic set via running it with a specific execution time limit. Moreover, these methods have been designed or tuned to solve the target instances of only one problem to deliver high quality solutions compared to the state of the art methods. However, the main motivation of hyper-heuristics as mentioned before is to provide generality. Therefore, a hyper-heuristic should be capable of working under different circumstances, namely distinct heuristic sets with varying size, different instances from different problem domains and different execution time.

The following list shows the main questions to consider while designing a selection hyper-heuristic.

- which heuristic should be applied at each decision step?
- what are the elements to manage relatively large heuristic sets?
- what are the effects of different heuristic sets?
- how can a heuristic set be used to get the best performance?
- what should be the quality of a solution for accepting it?
- how should the search be diversified?

These questions can be answered before starting to solve a problem instance (offline) or while solving it (online). Offline learning methods need to be trained on a group of instances. Thus, the performance of a hyper-heuristic with offline learning is limited by these instances. It should be noted that giving certain decisions while solving a problem instance based on the changing characteristics of a search space is always useful for better hyper-heuristics. Thus, utilising online learning components is an advantage for hyper-heuristics. However, for some cases, even a very naive approach can deliver similar or even better quality results compared to a hyper-heuristic with learning. The execution time limit is a major factor responsible for such empirical outcomes. For instance, if the number of improving or new best solutions is small with respect to the total number of iterations spent and the given execution time is relatively high, then choosing poor heuristics many times will not cost much to the hyper-heuristic. On the other hand, in the case of a limited execution time, each decision is very valuable. Therefore a learning mechanism may be extremely useful. For the purpose of generality, a hyper-heuristic should have some learning components to deal with these cases.

In hyper-heuristic studies, the number of heuristics used is generally small. However, utilising as many heuristics as possible might help to provide better performance. At the same, this requires additional components for the management of such large sets. Therefore, a hyper-heuristic should accommodate certain features to deal with different heuristic sets with different sizes. For using heuristics in a more efficient way, heuristic hybridisation opportunities and parameter adaptation methods for parametric heuristics should be investigated.

The decision about whether to accept or reject a solution generated by the selected heuristics has a major effect on the overall performance of a hyper-heuristic. If the heuristic set consists of only a few heuristics, then its effect is more significant. The general approach for the move acceptance is to utilise a method with diversification capabilities. The search process is diversified by accepting worsening solutions using threshold values under certain conditions. The evolvability characteristic of the search space is the most important criterion to decide on this threshold level. In addition, the number of neighbouring solutions around a current solution should be taken into account for the diversification strategy. Moreover, restarting and re-initilisation methods should be devised to support the diversification.

IV. Management

The present tendency toward solving combinatorial search and optimisation problems involves combining multiple mechanisms to reach more competent solution techniques. Aggregating the strength of multiple search mechanisms is the primary motivation underlying selection hyper-heuristics. Various components exist for achieving this management process in a favourable way. A considerable number of these approaches involve certain learning devices to facilitate the selection of these mechanisms. For this purpose, the characteristics and performance of these search mechanisms are monitored in an *offline* or *online* manner. In this study, an online learning strategy, i.e. adaptive dynamic heuristic set, is employed to identify effective heuristic subsets for specific regions of the search space while addressing a target problem instance.

A. Adaptive Dynamic Heuristic Set Strategy

The availability of a large set of low-level search mechanisms can be considered as an advantage because of its potential to explore large parts of the search space. However, such a situation can turn into a disadvantage due to the hardness of managing the high number of low-level heuristics. This could happen especially when these heuristics have different characteristics and a diverse speed range. Therefore we introduce the adaptive dynamic heuristic set (ADHS) strategy [11], [43]. It is a method assessing the performance of each heuristic at the end of a number of iterations to keep the best performing heuristics in the set whilst excluding the other ones. The contribution of this method to the whole framework is shown in Fig. 1. The number of iterations passed to determine such an elite heuristic subset refers to a phase. For the exclusion process, a performance metric representing the most obvious performance elements is used. These elements are related to the improvement capabilities of the heuristics as well as their speed. The details of this metric are shown in the following equation. $C_{p,best}(i)$ denotes the number of new best solutions found. $f_{imp}(i)$ and $f_{wrs}(i)$ correspond to the total improvement and worsening. $f_{p,imp}(i)$ and $f_{p,wrs}(i)$ refer to the same measurement but only during a single phase. t_{remain} refers to the remaining time to finish the whole search process. $t_{spent}(i)$ and $t_{p,spent}(i)$ are the time spent by heuristic i until that moment, the same measurement during a phase respectively. Each w_i denotes the weight of its corresponding performance element. The weights are set as $\{w_1 >> w_2 >> w_3 >> w_4 >> w_5\}$ to provide a strict priority between the given performance elements.

$$\begin{aligned} p_i &= w_1\Big[\Big(C_{p,best}(i)+1\Big)^2\Big(t_{remain}/t_{p,spent}(i)\Big)\Big]\times b + \\ & w_2\Big(f_{p,imp}(i)/t_{p,spent}(i)\Big) - w_3\Big(f_{p,wrs}(i)/t_{p,spent}(i)\Big) + \\ & w_4\Big(f_{imp}(i)/t_{spent}(i)\Big) - w_5\Big(f_{wrs}(i)/t_{spent}(i)\Big) \quad (1) \\ & \quad (2) \\ b &= \begin{cases} 1, & \sum_{i=0}^{n} C_{p,best}(i) > 0 \\ 0, & otw. \end{cases} \end{aligned}$$

In the proposed approach, the number of phases during which a heuristic stays out of an elite heuristic set is denoted by its tabu duration. For decreasing user dependency, the tabu duration and the number of iterations for one phase are calculated based on the number of heuristics available in the elite heuristic subset. The tabu duration is set to $d = \sqrt{2n}$ and the phase length (*pl*) is defined as the product of the tabu duration and a constant value ($ph_{factor} = 500$). These values are updated at the end of each phase. The calculated p_i values are combined with different quantities and they appear to be noisy. Therefore, we decided to convert the performance values into a quality index ($QI \in [1, n]$) value. The heuristic with the lowest p_i gets 1, the other ones get one unit more based on their order. The average (avg) of these QI values is calculated to specify the heuristics that will be excluded. Tabu heuristics also have $QI = 1$.

$$avg = \Big\lfloor \Big(\sum_{i}^{n} QI_i\Big)/n \Big\rfloor \quad (3)$$

1) Phase length adaptation: The phase length (*pl*) is the number of iterations required to check the performance of the heuristics within the ADHS strategy. Using the same pl for different heuristic sets may be unfair in the case of speed differences between heuristic subsets. Thus, pl is updated with respect to the average speed required per move of the heuristic subset. As a reference point, the number of phases requested ($ph_{requested} = 100$) during the whole run is used to determine a possible phase duration. $pl \in [d \times ph_{base}, d \times ph_{factor}]$ is calculated depending on this information as shown in Equation 4. pl is kept within certain limits if the calculated pl is too low or too high.

$$\begin{aligned} pl &= ph_{duration}/t_{subset} \quad (4) \\ ph_{duration} &= t_{total}/ph_{requested} \end{aligned}$$

2) Tabu duration adaptation: Whenever a heuristic is excluded, its status becomes tabu and it waits to get back to the heuristic set for a number of phases, which is described as tabu duration. The tabu duration of each heuristic is the same at the beginning of the search and they are updated during the run. If an excluded heuristic enters the active set for the next phase and is excluded immediately after this phase, its tabu duration is incremented by 1. If this heuristic stays in the elite heuristic subset at the end of the phase, its tabu duration is set to its initial value. This incrementation continues until the tabu duration reaches its upper limit, resulting in a permanent exclusion of this heuristic.

3) Extreme heuristic exclusion: An additional heuristic exclusion procedure is called at the end of each phase to fasten the search process by immediately eliminating slow and ineffective heuristics. The metric employed for this purpose is shown in Equation 4. $t_{perMove}(i)$ denotes the average time spent per move by heuristic i and $t_{perMove}(fastest)$ refers to the same measure for the fastest heuristic. The standard deviation (σ) and the average (ϖ) of the $exc(i)$ values are used to determine the heuristics that should be excluded. If such a slow heuristic would find a new best solution in the current phase, it is not considered for extreme exclusion. The cases leading to the extreme exclusion process are presented in Equation 5. In particular, if the heuristic residing in the current heuristic subset did not generate any new best solution ($nb = 0$), this heuristic exclusion process is discarded for the current phase. The idea here is to keep at least one heuristic that explored new best solutions before in the heuristic set.

$$exc(i) = t_{perMove}(i)/t_{perMove}(fastest) \tag{4}$$

$$\sigma > 2.0\ ;\ exc(i) > 2\varpi\ ;\ nb > 1 \tag{5}$$

4) Heuristic selection: Most of the available heuristic selection mechanisms consider the improvement capability of the heuristics. The general trend is choosing effective heuristics subject to their performance changes. In the proposed hyper-heuristic, the selection operation is carried out based on a probability vector determined and updated by the number of new best solutions and speed for each heuristic. Equation 6 shows the rule to specify the probability of being selected for heuristic i. tf is a linearly decreasing time factor from 1 to 0.

$$pr_i = \left((C_{best}(i) + 1)/t_{spent}\right)^{(1+3tf^3)} \tag{6}$$

V. Mentoring

In a traditional selection hyper-heuristic, one heuristic is selected at each decision step and applied as is. It might be useful to additionally command heuristics instead while selecting them. In the mentoring part, possible performance enhancement opportunities are investigated based on this idea. The first approach consists in exploring effective heuristic hybridisations or heuristic pairs, which can result in a better performance, rather than applying heuristics alone. In addition to that, manipulating heuristics through playing with their parameters is another approach under mentoring. The following sub-sections present how these two mentoring approaches work.

A. Relay Hybridisation

Relay hybridisation is a method to consecutively apply (meta-)heuristics where each (meta-)heuristic uses the solution generated by the preceding (meta-)heuristic. In this study, we introduce a relay hybridisation technique that consecutively applies two low-level heuristics for finding new best solutions. A heuristic list involving the heuristics that found new best solutions when applying them as follow-up heuristics is utilised for each heuristic as illustrated in Fig. 1. A simple decision mechanism is used for this hybridisation approach as presented in Algorithm 1. In this equation, C_{phase} is a counter showing the number of iterations passed during the current phase. $C_{best,s}$ and $C_{best,r}$ refer to the number of new best solutions found by the regular, single heuristic selection mechanism and by relay hybridisation respectively. $\gamma \in [0, 1]$ is a variable denoting the usage frequency of relay hybridisation. δ is a dynamic variable to determine the selection share between regular single heuristic selection and relay hybridisation. Thus, if the heuristics are more effective for locating new best solutions alone compared to their pairwise performance, then single heuristics are preferred at each iteration. If the calculated value is out of its predetermined limits $\{(1/R, R)$ for $R = 50\}$ then it is set to one of its bounds. The predetermined values in the pseudocode were set based on some preliminary experiments and no automated offline tuning device was used.

Algorithm 1: Relay hybridisation

```
  Input: $list_{size} = 10; \gamma \in (0.02, 50); p, p' \in [0 : 1]$
1 $\gamma = (C_{best,s} + 1)/(C_{best,r} + 1)$
2 if $p \leq (C_{phase}/pl)^{\gamma}$ then
3     select $LLH$ using a LA and apply to $S \rightarrow S'$
4     if $size(list_i) > 0$ and $p' <= 0.25$ then
5         select a $LLH$ from $list_i$ and apply to $S' \rightarrow S''$
6     else
7         select a $LLH$ and apply to $S' \rightarrow S''$
      end
  end
```

In addition, the tabu approach used for ADHS is employed to disable relay hybridisation if no best solution was found during a phase. If this is repeated during consecutive phases, the tabu duration is incremented by 1 like in ADHS.

1) Learning Automata: A learning automaton maintaining the selection probabilities of the heuristics (p_i) was employed for the relay hybridisation. The learning automata had been previously used to choose single heuristics in [42]. For the update operations, a *linear reward-inaction* update scheme was used. This scheme rewards heuristics discovering new best solutions. No other type of outcome was used to update the probabilities. The corresponding learning rate was set as $\lambda_1 = 0.5$. The details of this operation are shown in Equation 6 and 7. The first equation shows the update function for the heuristic applied. The second equation was utilised to decrease the updated amount from the rest of the heuristics to keep the total probability 1.

$$\begin{aligned} p_i(t+1) &= p_i(t) \quad +\lambda_1\, \beta(t)(1-p_i(t)) \\ &\qquad -\lambda_2(1-\beta(t))p_i(t) \qquad (6) \\ &\qquad \text{if } a_i \text{ is the action taken at time step } t \end{aligned}$$

$$\begin{aligned} p_j(t+1) &= p_j(t) \quad -\lambda_1\, \beta(t)p_j(t) \\ &\qquad +\lambda_2(1-\beta(t))[(r-1)^{-1}-p_j(t)] \quad (7) \\ &\qquad \text{if } a_j \neq a_i \end{aligned}$$

B. Parameter Adaptation of the Parametric Heuristics

In HyFlex, each heuristic set associated with each problem consists of four type of low-level heuristics, i.e. mutational heuristics, ruin-recreate heuristics, local search heuristics (hill climbers) and crossover operators. All the heuristics are applied to a single solution except crossover operators that require two solutions. For them, a population of five solutions including previously explored new best solutions is accommodated. They are applied using the current solution and a randomly selected solution from these five solutions. Each time a new best solution is found, a randomly chosen solution from these solutions is replaced by the new solution.

Some of the heuristics require a parameter called "*intensity of mutation*" denoting the impact of the perturbation. Heuristics only aiming at improving solutions have a parameter called "*depth of search*" that refers to the number of steps to check for improvement. Setting right values for these heuristics require a learning component that adapts their parameters relying on their performance changes. For this purpose, a heuristic type based reward-penalty strategy is employed. Besides the provided information about heuristic types, heuristics are categorised as *ImprovingOrEqual*, *ImprovingMore*, *WorseningMore*, *WorseningOrEqual*, *OnlyEqual*. These types are always checked to reflect any change. For instance, one heuristic can be ImprovingMore during early iterations, then it can turn into a WorseningMore type. In such cases, changing reward and penalty functions may be very effective. The details of the parameter (val_i) update operations are shown in Algorithm 2. The θ values are the update rates for the given environmental feedbacks, i.e. a new best solution is found, the current solution is improved, the current solution is worsened, the quality of the current solution is the same. They were set as $\theta_1 = 0.01$, $\theta_2 = 0.001$, $\theta_3 = 0.0005$, $\theta_4 = 0.0001$. This adaptation process was handled by setting some predetermined values. These values were set based on a simple logic of finding the best parameter not just based on their improvement capabilities, but also according to their speed changes. Thus, the parameter of a heuristic with high improvement performance should not be consistently increased and at the same time, the parameter of a heuristic with high worsening performance should not be immediately decreased. All these parameters should be set in such a way that their strengths could be used in balance.

Algorithm 2: Parameter adaptation for the heuristics

```
   $u = +1$; $p \in rand(0,1)$; $val_i \in [0.2, 1]$
1  if $f(S') < f(S_b)$ then
2      if $heuristic\_type = ImprovingOrEqual$ then
3          if $p < 0.5$ then $u = 0$
4      else if $heuristic\_type = ImprovingMore$ then
5          if $p \leq 0.25$ then $u = -1$
6          else if $p \leq 0.5$ then $u = 0$
7      else if $heuristic\_type = WorseningMore$ then
8          if $p < 0.5$ then $u = 0$
       end
9      $val_i = val_i + \theta_1 \times u$
10 else if $f(S') < f(S)$ then
11     if $heuristic\_type = ImprovingOrEqual$ then
12         if $p < 0.5$ then $u = 0$
13     else if $heuristic\_type = ImprovingMore$ then
14         if $p < 0.25$ then $u = -1$
15         else if $p < 0.5$ then $u = 0$
16     else if $heuristic\_type = WorseningMore$ then
17         if $p < 0.5$ then $u = -1$
       end
18     $val_i = val_i + \theta_2 \times u$
19 else if $f(S') > f(S)$ then
20     if $heuristic\_type = ImprovingMore$ then
21         if $p < 0.5$ then $u = 0$
       end
22     $val_i = val_i - \theta_3 \times u$
23 else
24     if $heuristic\_type = ImprovingOrEqual$ then
25         if $p < 0.25$ then $u = -1$
26         else if $p < 0.5$ then $u = 0$
27     else if $heuristic\_type = ImprovingMore$ then
28         if $p < 0.5$ then $u = 0$
29     else
30         $u = -1$
       end
31     $val_i = val_i - \theta_4 \times u$
   end
```

1) Oscillating parameters: Fig. 1 shows a feedback transmission from the move acceptance mechanism through the heuristic selection. This feedback warns the selection mechanism if the system fails to generate improvements. This warning is sent if re-initialisation is disabled due to the fact that re-initialisation is employed for the same purpose. It means that this issue could not be handled by the move acceptance mechanism, i.e. threshold level reaches l. Whenever these conditions occur, the parameters of the parametric heuristics oscillate. For the *OnlyImproving* heuristics, val_i is linearly updated between 0.5 and 1.0. For the other heuristics, val_i is changed between 0.2 and 0.5. At each 5000th iteration, the corresponding parameters are updated between their limits.

VI. Move Acceptance

A. Adaptive Iteration Limited List-based Threshold Accepting

Adaptive iteration limited list-based threshold accepting (AILLA) is an acceptance mechanism determining the threshold level in an adaptive manner using the fitness values of the previously found new best solutions [43]. Its simplest version, i.e. iteration limited threshold accepting (ILTA) [42], requires two parameters. The first parameter determines the number of consecutively found worsening solutions. If this number reaches a predetermined iteration limit, then a worsening solution is accepted if it meets a threshold value. This threshold value is determined based on the current best solution and a constant range factor. That is, the threshold value changes whenever a new best solution is found. Adaptive ILTA (AILTA) [44] adds another iteration limit for increasing the threshold value if required. Each time the number of consecutively encountered worsening solutions reach this value, the range value is updated by another constant factor. This operation provides a larger search region to get out of the point the search process converged to. The value of the range parameter is allowed to be incremented to an upper bound. On the other hand, AILLA determines threshold values for accepting worsening solutions from previously found new best solutions constituting a fitness list as displayed in Fig. 1. It accepts improving and equal quality solutions. A worse solution is accepted if no new best solutions were found after k consecutive worsening solutions and if the fitness value of the new solution is lower than a threshold value from the fitness list. In addition, it involves certain adaptive components to bring high performance with less user dependency. The details of AILLA is presented in Algorithm 3.

Algorithm 3: AILLA move acceptance

```
   Input: $i = 1, K \geq k \geq 0, l > 0$
   for i=0 to l-1 do $best_{list}(i) = f(S_{initial})$
1  if $adapt\_iterations \geq K$ then
2      if $i < l - 1$ then
3          $i++$
       end
   end
4  if $f(S') < f(S)$ then
5      $S \leftarrow S'$
6      $w\_iterations = 0$
7      if $f(S') < f(S_b)$ then
8          $i = 1$
9          $S_b \leftarrow S'$
10         $w\_iterations = adapt\_iterations = 0$
11         $best_{list}.remove(last)$
12         $best_{list}.add(0, f(S_b))$
       end
13 else if $f(S') = f(S)$ then
14     $S \leftarrow S'$
15 else
16     $w\_iterations++$
17     $adapt\_iterations++$
18     if $w\_iterations \geq k$ and $f(S') \leq best_{list}(i)$ then
19         $S \leftarrow S'$ and $w\_iterations = 0$
       end
   end
```

1) Adaptive iteration limit: The iteration limit ($k \in [5, \infty)$) is an important parameter for AILLA. Different k can be effective for different problem instances. Thus, this value is updated by a simple parameter control strategy as explained in Equation 8.

$$k = \begin{cases} ((l-1).k + iter_{elapsed})/l, & \text{if } cw = 0 \\ ((l-1).k + \sum_{i=0}^{cw} k.0.5^i.tf)/l, & \text{otherwise} \end{cases} \quad (8)$$
$$tf = (t_{exec} - t_{elapsed})/t_{exec}$$
$$cw = iter_{elapsed}/k$$

2) Decreasing list length: The list length of AILLA is decreased over time in order to reduce the level of diversification towards the end of the search process. The implemented update method is shown in Equation 7. In this study, the parameters were set as follows: $l_{base} = 5$ and $l_{initial} = 10$.

$$l = l_{base} + (l_{initial} - l_{base} + 1)tf^3 \quad (7)$$

3) Re-initialisation: For solving computationally intractable problems, fast and efficient methods without optimality guarantee are common. Due to the stochastic nature of such methods, it is impossible to determine the exact execution time required to find a solution at a specific quality level. Thus, the performance of the same algorithm running multiple times may differ. Re-initialisation methods and restarting mechanisms may be useful to take advantage of such circumstances. These approaches should be considered together with the other mechanisms capable of diversifying the search. The acceptance mechanism suggested is already responsible for diversification. The level of diversification provided by the acceptance depends on its list length. Finding a good value for the list length can be problematic. An adaptive parameter based on the correlation between the solutions belonging to a specific sub-region may be effective. However, the requirement of exploitation in a limited time should be handled using a more effective method. For that purpose, whenever the threshold level reaches l, with additional limitations a new initial solution is generated as depicted in Fig. 1. Based on remaining execution time, the cost of re-initialisation and the possibility of finding a new best solution after re-initialisation, it is decided whether the re-initialisation should be disabled or not. Each time it is disabled, the best solution found after re-initialisations is used as the current solution for further improvement. For consistency with the acceptance mechanism, the list constituted for this specific solution is also employed.

VII. HyFlex

HyFlex is a software environment providing a number of instances from different problem domains [1]. The idea is to design and apply a selection hyper-heuristic to show its level of generality across the given instances. As one of the main traits of hyper-heuristics, the problem-independency is

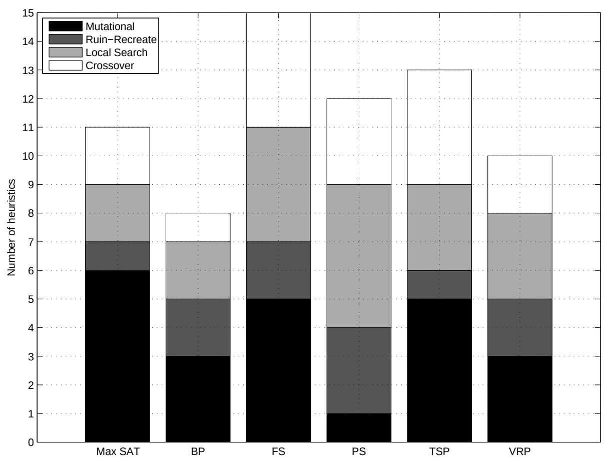

Fig. 3. Heuristic-type distributions and heuristic set sizes for each problem domain

guaranteed by this environment. Therefore, a specific heuristic set for each problem domain is also provided. For the sake of heuristic diversity, four types of heuristics were implemented in each heuristic set. These types are *mutational heuristics*, *ruin-recreate heuristics*, *local search* and *crossovers* as mentioned in the previous section. Here, local search heuristics (hill climbers) guarantee to find improved or equal quality solutions with respect to the solutions they start from. The rest of the heuristics may also generate worsening solutions. As mentioned in Section V, some of these heuristics can be manipulated by changing their parameters.

A. Problem Domains

The current version of HyFlex involves six problems, namely 1D bin packing, max SAT, permutation flowshop scheduling, personnel scheduling, travelling salesman and vehicle routing problems. For each problem domain, a heuristic set involving distinct numbers of heuristics from aforementioned types is available. The heuristic type distributions of each heuristic set are shown in Fig. 3. The problem instances provided in HyFlex are displayed in the following subsections. More details about the instances are available in [1], [60].

1) Bin Packing: The 1D bin packing problem aims at minimising the number of homogeneous bins required for placing a set of given items. Equation 8 shows the fitness function proposed by HyFlex, n refers to the number of bins used, C is the capacity of a bin and $fullness_i$ means the total size of the elements in bin i. The fitness function considers the fullness of a bin while the objective is to minimise n to make the improvement easier.

$$f = 1 - \Big(\frac{\sum_{i=1}^{n} (fullness_i / C)^2}{n} \Big) \tag{8}$$

Table II shows the instances for this domain.

2) Max SAT: The max SAT problem deals with minimising the number of broken clauses of a boolean formula in a conjunctive normal form. The problem instances in the testbed are given in Table III. All the instances were taken from the SAT competitions.

TABLE II
BIN PACKING INSTANCES [60] (THE COLUMN 'BEST' SHOWS THE NUMBER OF BINS USED. THE SOLUTIONS WITH * ARE OPTIMAL)

Inst.	Name	Best
0	falkenauer/u1000-00	399*
1	falkenauer/u1000-01	406*
2	schoenfieldhard/BPP14	62
3	schoenfieldhard/BPP832	60
4	10-30/instance1	N/A
5	10-30/instance2	N/A
6	triples1002/instance1	N/A
7	triples2004/instance1	N/A
8	test/testdual4/binpack0	N/A
9	test/testdual7/binpack0	N/A
10	50-90/instance1	N/A
11	test/testdual10/binpack0	N/A

TABLE III
MAX SAT INSTANCES [60] (THE SOLUTIONS WITH $*$ ARE OPTIMAL AND > 0 SHOWS THAT THE CORRESPONDING SOLUTIONS SHOULD HAVE A FITNESS VALUE BIGGER THAN ZERO)

Inst.	Name	Best
0	contest02-Mat26.sat05-457.reshuffled-07	> 0
1	hidden-k3-s0-r5-n700-01-S2069048075.sat05-488.reshuffled-07	N/A
2	hidden-k3-s0-r5-n700-02-S350203913.sat05-486.reshuffled-07	N/A
3	parity-games/instance-n3-i3-pp	0*
4	parity-games/instance-n3-i3-pp-ci-ce	0*
5	parity-games/instance-n3-i4-pp-ci-ce	N/A
6	HG-3SAT-V250-C1000-1	6
7	HG-3SAT-V250-C1000-2	6
8	HG-3SAT-V300-C1200-2	8
9	MAXCUT/SPINGLASS/t7pm3-9999	N/A
10	jarvisalo/eq.atree.braun.8.unsat	> 0
11	HG-3SAT-V300-C1200-4	7

3) Permutation Flowshop: The permutation flowshop scheduling problem requires minimising the completion time of the last processed job (makespan) while assigning a set of jobs to a group of machines. Table III presents the related problem instances provided by HyFlex.

TABLE IV
PERMUTATION FLOWSHOP SCHEDULING INSTANCES [60]

Inst.	Name	Best
0	100x20/1	6202
1	100x20/2	6183
2	100x20/3	6271
3	100x20/4	6269
4	100x20/5	6314
5	200x10/2	10480
6	200x10/3	10922
7	500x20/1	26059
8	500x20/2	26520
9	500x20/4	26456
10	200x20/1	11181
11	500x20/3	26371

4) Personnel Scheduling: The personnel scheduling problem requires the assignment of employees to certain timeslots for a given planning period. The instances tested here belong to the nurse rostering domain and they are listed with the values of the best known solutions in Table V.

5) Travelling Salesman: The objective of the travelling salesman problem is to find the shortest route for visiting a set of cities and returning to the starting city. The travelling salesman problem instances are illustrated in Table VI.

6) Vehicle Routing: The vehicle routing problem deals with visiting a number of locations within their given time windows using a small number of vehicles. That is, the objective is to minimise the total number of vehicles used (n) and the total

TABLE V
PERSONNEL SCHEDULING INSTANCES [60]

Inst.	Name	Best
0	BCV-3.46.1	3280
1	BCV-A.12.2	1953
2	ORTEC02	270
3	Ikegami-3Shift-DATA1	2
4	Ikegami-3Shift-DATA1.1	3
5	Ikegami-3Shift-DATA1.2	3
6	CHILD-A2	1095
7	ERRVH-A	2142
8	ERRVH-B	3121
9	MER-A	9017
10	BCV-A.12.1	1294
11	ORTEC01	270

TABLE VI
TRAVELLING SALESMAN INSTANCES [60]

Inst.	Name	Best (Optimal)
0	pr299	48191
1	pr439	107217
2	rat575	6773
3	u724	41910
4	rat783	8806
5	pcb1173	56892
6	d1291	50801
7	u2152	64253
8	usa13509	19982859
9	d18512	645238

distance travelled ($\sum d_i$). Equation 9 represents the fitness function to be minimised.

$$f = n * 1000 + \sum_{i=1}^{n} d_i \tag{9}$$

Table VII provides the vehicle routing instances.

TABLE VII
THE VEHICLE ROUTING PROBLEM INSTANCES [60]

Inst.	Name	Best	# Vehicles	Distance
0	rc207	4061.14	3	1061.14
1	r101	20645.79	19	1645.79
2	rc103	12261.67	11	1261.67
3	r201	5252.37	4	1252.37
4	r106	13251.98	12	1251.98
5	c1-10-1	142478.95	100	42478.95
6	rc2-10-1	83373.15	20	63373.15
7	r1-10-1	153904.23	100	53904.23
8	c1-10-8	135499.59	93	42499.59
9	rc1-10-5	136631.89	90	46631.89

VIII. COMPUTATIONAL RESULTS

The hyper-heuristic developed, ADHS-AILLA, was applied 10 times on each problem instance using Pentium Core 2 Duo 3GHz PCs with 3.23 GB memory on Windows XP. There are 12 instances for the max SAT, bin packing, permutation flowshop scheduling and personnel scheduling problems. For the travelling salesman and vehicle routing problems, 10 instances are available per domain. The proposed approach was tested for 10 minutes and 1 hour execution time on these instances.

Table XI shows the scores of the different hyper-heuristics after 10 minutes of execution time based on the CHeSC scoring system. The overall scores show that ADHS-AILLA is clearly the best approach for the given problem instances. The rest of the hyper-heuristics are ranked as ADHS-GD, ADHS-LATE, AHDS-SA, AHDS-IE, SR-AILLA, SR-LATE, SR-SA, SR-GD from better to worse. ADHS-AILLA is the best approach for the max SAT, permutation flowshop, personnel scheduling and travelling salesman domains. It comes second for the bin packing problem and third for the vehicle routing problem. AILLA performs best among the hyper-heuristics with SR. This indicates that the proposed acceptance strategy is capable of effectively working on different problem domains. ADHS compared to SR for all the acceptance mechanisms provides a high performance improvement. In particular, the score of GD goes from 35 to 418 by using ADHS. Table XII presents the scores after running the hyper-heuristics for 1 hour. ADHS-AILLA is still the best performing method. There are slight differences in the ranking of the hyper-heuristics based on their overall performance. SR-AILLA has a higher score than ADHS-IE and ADHS-SA is slightly better than ADHS-LATE.

A Wilcoxon test with a confidence interval of 95% was utilised to determine the significance of the performance difference between the best hyper-heuristic and the rest after 1 hour of execution for each problem instance. The hyper-heuristics delivering similar performance for a specific instance compared to the best approach for the corresponding problem are coloured in Table IX and X. In the case of the max SAT problem, ADHS-AILLA outperforms the other tested hyper-heuristics. When dealing with the bin packing problem, ADHS-IE is the best approach, but its performance is not significantly better than ADHS-AILLA. It also performs similarly with ADHS-LATE for all the instances except two. ADHS-AILLA provides the highest average fitness for almost all the permutation flowshop problem instances. For five or six instances, there is no significant performance difference between ADHS-AILLA and the other hyper-heuristics using ADHS. Moreover, the ADHS hyper-heuristics are significantly better than the hyper-heuristics with SR for this problem domain. In consideration of the personnel scheduling problem, the hyper-heuristics with ADHS perform statistically similar. In addition, ADHS-AILLA could not achieve significantly better solutions compared to SR-AILLA and SR-LATE for most of the instances. Regarding the travelling salesman problem, ADHS-AILLA performs similarly with the other hyper-heuristics with ADHS. The ADHS hyper-heuristics perform significantly better than the SR hyper-heuristics. In view of the vehicle routing problem, ADHS-GD provides significantly better results for most of the instances compared to the other hyper-heuristics. However, for five instances, there is no significant performance difference with the AILLA hyper-heuristics.

Table XIII shows the % difference between the best known solution and the best solution found by ADHS-AILLA after 1 hour. For the bin packing problem instances, only the solutions for the first four instances are available. ADHS-AILLA finds solutions with the same number of bins for these instances. The proposed approach finds a solution with only one unsatisfiable clause for the max SAT instance 3. The hyper-heuristic found the same quality solutions with the best known solutions for the instances 4 and 11. Moreover,

TABLE VIII

THE EXPERIMENTAL RESULTS WITH 10 MINUTES OF EXECUTION TIME (REINITIALISATION IS AVAILABLE ONLY FOR ADHS-AILLA)

	Inst.	ADHS-AILLA		ADHS-GD		ADHS-LATE		ADHS-SA		ADHS-IE		SR-AILLA	
		min	avg	min	avg	min	avg	min	avg	min	avg	min	avg
Max SAT	0	2	2,7	4	11,6	4	8,4	2	3	7	12	5	8,7
	1	20	23,8	23	29,6	23	25,6	20	26,8	22	28,2	21	24,5
	2	15	18,1	21	25,7	18	22,3	19	21,5	19	24,4	14	19,2
	3	1	3,5	5	12	7	14,4	4	9,1	15	19,6	5	9,3
	4	1	2,7	2	17,6	2	22	2	15,6	13	28,4	6	10
	5	3	13,2	18	38,9	19	39,3	7	26,9	24	42	16	28,6
	6	5	5,3	7	8,1	5	9,3	5	7,9	5	10,2	5	6,8
	7	5	5,4	7	8,3	7	9,6	5	7,5	8	10,4	5	6,1
	8	5	7,1	9	11,9	9	11,4	9	10,1	9	13,6	8	8,8
	9	209	212	213	216,2	213	217,2	209	214,3	215	218,4	211	212,9
	10	1	2,2	11	17,7	9	14,5	1	7,3	15	18,8	11	14,4
	11	7	8	10	11,5	9	12,1	7	9,7	12	13,6	7	9,1
Bin Packing	0	0,00276	0,00565	0,00637	0,00686	0,00264	0,00544	0,00595	0,00673	0,00244	0,00449	0,00659	0,01024
	1	0,00288	0,00442	0,00671	0,00716	0,00281	0,00433	0,00670	0,00706	0,00335	0,00483	0,00751	0,00940
	2	0,02030	0,02115	0,01871	0,01942	0,02030	0,02128	0,01840	0,01928	0,01971	0,02160	0,02106	0,02260
	3	0,01970	0,02089	0,01872	0,01911	0,01970	0,02185	0,01872	0,01902	0,01970	0,02143	0,02039	0,02236
	4	0,00034	0,00250	0,00501	0,00542	0,00034	0,00440	0,00541	0,00566	0,00034	0,00422	0,00034	0,00375
	5	0,00306	0,00348	0,00374	0,00403	0,00306	0,00358	0,00360	0,00545	0,00306	0,00343	0,00306	0,00335
	6	0,01082	0,01522	0,00572	0,01310	0,00561	0,01549	0,00563	0,01013	0,01069	0,01477	0,01107	0,01569
	7	0,01340	0,01524	0,02149	0,02623	0,01300	0,01533	0,01964	0,03199	0,01358	0,01530	0,05635	0,05989
	8	0,04081	0,04263	0,03701	0,03820	0,03733	0,04080	0,04243	0,04419	0,03997	0,04215	0,11697	0,12356
	9	0,00141	0,00363	0,00153	0,00354	0,00049	0,00273	0,00152	0,00301	0,00146	0,00303	0,03402	0,03641
	10	0,10828	0,10828	0,10828	0,10828	0,10828	0,10828	0,10834	0,10840	0,10828	0,10828	0,11051	0,11089
	11	0,00234	0,00378	0,00428	0,01164	0,00352	0,00601	0,00556	0,01101	0,00222	0,00335	0,05932	0,06371
Permutation Flowshop	0	6251	6271,9	6266	6300	6258	6300,7	6266	6297,8	6265	6304,7	6326	6345,2
	1	6225	6238	6232	6257	6244	6262,9	6231	6259,8	6243	6262,5	6261	6297,2
	2	6309	6316,1	6313	6341,5	6324	6343	6307	6339,9	6313	6336,1	6342	6364,5
	3	6318	6340,6	6306	6339,9	6310	6355,1	6305	6342,8	6323	6342,7	6331	6363,6
	4	6364	6386,8	6361	6384,8	6368	6391,3	6357	6387	6362	6386,8	6394	6418,8
	5	10493	10497,3	10497	10502,2	10497	10501,5	10494	10505,1	10494	10503,4	10497	10514
	6	10922	10925	10922	10923,3	10922	10922,9	10923	10923	10922	10925,3	10922	10927,2
	7	26245	26296	26247	26299,3	26366	26399,3	26275	26303,3	26220	26297,5	26328	26391
	8	26778	26812,7	26771	26809,6	26804	26883,3	26704	26786	26779	26815,7	26787	26829,6
	9	26614	26678,3	26578	26654,2	26646	26710,1	26558	26662,9	26593	26649,4	26599	26690,3
	10	11319	11364,6	11333	11360,6	11336	11398,5	11344	11379,5	11319	11375,7	11397	11436,8
	11	26585	26630,1	26514	26611,1	26640	26680,9	26574	26644,8	26577	26626,9	26595	26659,9
Personnel Scheduling	0	3300	3332,3	3321	3335,4	3310	3336,3	3296	3325,5	3311	3338,1	3323	3347,6
	1	2203	2346,1	1975	2280,4	2230	2357	2025	2443,9	2050	2348,5	2228	2475,7
	2	335	373	355	389,5	345	386	330	379,5	315	390,5	385	1178,1
	3	16	22,1	20	25	18	23,9	20	29,3	20	27,8	13	24,1
	4	20	25,6	17	24	22	26,6	25	32,1	21	27,2	21	25,3
	5	18	25,9	23	25,7	19	28,8	25	31	29	34	11	27,3
	6	1108	1169,2	1128	1152,1	1110	1147,2	1121	1229,6	1106	1251,2	1108	1234,9
	7	2189	2282,7	2221	2304,8	2210	2265	2182	2279,9	2174	2267	2184	2273
	8	3173	3258,9	3170	3257	3267	3324,4	3177	3349,6	3154	3355,9	3172	3311,7
	9	9406	9614,8	9619	10016,9	10181	11463,7	9440	9663,4	9390	9614,2	9522	9718,6
	10	1535	1670,8	1620	1723	1489	1652,9	1655	1839,2	1560	1750,1	1689	1995,9
	11	315	335,5	330	353,1	310	345,5	290	318	305	466,5	355	943,5
Travelling Salesman	0	48194,92	48195,44	48194,92	48213,29	48194,92	48213,29	48194,92	48222,47	48194,92	48250,02	48689,30	49109,89
	1	107509,62	109023,22	108207,20	109058,58	108121,24	109295,50	109212,79	109485,25	108300,28	109113,91	110437,02	111682,08
	2	6797,74	6809,59	6799,68	6810,81	6799,67	6816,14	6795,97	6809,00	6799,30	6808,64	6944,03	6979,64
	3	41965,50	42025,67	41951,46	42022,91	41972,26	42048,51	41945,83	42038,86	41970,75	42027,68	42702,84	43038,84
	4	8857,43	8875,78	8861,95	8881,52	8863,07	8886,16	8842,99	8887,62	8867,05	8885,60	9010,54	9112,40
	5	57305,54	57473,51	57201,15	57506,05	57246,11	57451,09	57261,60	57453,98	57177,79	57442,20	59593,48	60063,03
	6	52513,79	53097,43	52870,50	53993,95	52587,69	53519,85	52028,89	53437,29	52965,01	53661,89	53873,99	54773,29
	7	66219,91	66700,34	66531,26	66925,67	66316,43	66674,21	66154,66	66674,01	66360,95	66761,00	68816,69	69137,90
	8	20688884,44	20791227,71	20713409,15	20804743,56	20766756,65	20857643,74	20703118,69	20798709,26	20688847,10	20815142,59	21203912,18	21335269,51
	9	667481,20	669868,77	667899,50	671185,01	669271,95	670847,38	668223,98	670190,82	668466,98	669942,61	675137,61	676934,66
Vehicle Routing	0	5093,77	5123,63	4158,04	4775,20	5057,15	5122,85	5102,44	5149,55	5135,82	5201,24	5100,96	5140,79
	1	20654,06	20758,71	20652,47	20755,40	20660,03	21265,34	21650,18	21774,53	20695,61	21989,32	20651,26	20656,30
	2	13286,60	13340,39	12330,09	13152,21	13361,03	13787,77	13312,37	13873,27	14384,55	14521,91	13319,40	13335,66
	3	5301,64	5330,35	5317,38	5355,76	5313,71	5338,24	5331,32	5552,44	5349,24	5664,67	5310,36	5345,73
	4	14264,83	14289,90	13277,95	14074,31	14277,80	14424,07	14282,24	15022,00	14310,28	14930,38	14257,83	14283,32
	5	145390,19	148755,39	142479,18	144776,16	142490,60	146715,69	145424,21	149171,17	145436,10	149344,45	147491,90	152626,07
	6	58170,85	61050,39	56757,46	59768,08	57152,31	60290,38	58782,04	60672,66	58788,27	60177,41	59234,69	59932,63
	7	159715,50	162125,26	159266,06	160807,34	159383,38	160535,57	159587,27	161230,18	160448,29	161754,46	161810,90	162411,09
	8	148316,30	152533,05	145129,59	148821,77	148903,27	151465,09	151092,14	154398,56	153854,25	157180,94	155569,62	158679,76
	9	145998,40	148724,85	145598,32	147328,19	145118,16	146872,12	145824,15	147828,00	146885,89	148502,40	146922,79	148065,62

it explored new best solutions for instances 6, 7 and 8. The best solutions for the other instances are not known, therefore it is not possible to make a comparison for them. The hyper-heuristic found the best known solution for permutation flowshop problem instance 6. The % performance difference is usually lower than 1% except for instance 10 (1.14%). The best results on the personnel scheduling problem instances 6 and 9 are new best solutions. For the instances 2 and 10, the difference increases above 10%. The % difference on the instances 3, 4 and 5 is very high since the fitness values of the best known solutions are less than 10. For the travelling salesman problem, the difference between the best solutions found by ADHS-AILLA and the optimal solutions is smaller than 1% for the first six instances. This difference increases up to 3.51% for larger instances.

The best solutions found by ADHS-AILLA for the first five VRP instances utilise the same number of vehicles and the travelled distance is very near to best known solutions. For the last five instances, the proposed approach finds the same number of vehicles only for instance 7 and the difference is relatively high. For the other problem domains, the number of vehicles is different, therefore it is hard to make a direct comparison between them.

TABLE XIII

THE % DIFFERENCE BETWEEN THE BEST KNOWN SOLUTIONS AND THE BEST SOLUTIONS FOUND BY ADHS-AILLA AFTER 1 HOUR

Inst.	**MSAT**	**BP**	**FS**	**PS**	**TSP**	**VRP**
0	N/A	0	0.69	0.43	0.01	9.72
1	N/A	0	0.47	2.30	0.07	0.41
2	N/A	0	0.21	11.11	0.34	7.94
3	100	1.67	0.29	450	0.04	1.09
4	0	N/A	0.46	480	0.46	2.80
5	N/A	N/A	0.05	433.33	0.08	N/A
6	-16.67	N/A	0	-0,18	3.51	N/A
7	-16.67	N/A	0.53	0.33	2.18	11.08
8	-37.50	N/A	0.58	0.38	3.03	N/A
9	N/A	N/A	0.37	-0.43	2.98	N/A
10	N/A	N/A	1.14	13.60	-	-
11	0	N/A	0.35	3.70	-	-

TABLE IX
THE EXPERIMENTAL RESULTS WITH 1 HOUR OF EXECUTION TIME (REINITIALISATION IS AVAILABLE ONLY FOR ADHS-AILLA)

	Inst.	ADHS-AILLA		ADHS-GD		ADHS-IE		ADHS-LATE		ADHS-SA	
		min	avg	min	avg	min	avg	min	avg	min	avg
Max SAT	0	2	2,5	4	8,1	5	12,6	2	7,4	2	2,5
	1	20	21,8	24	27,4	21	25,5	23	25,9	21	24,1
	2	15	16,7	16	23,5	19	22,6	18	21,6	17	20,3
	3	1	2,2	4	9,2	7	15,9	4	10,6	2	4,8
	4	0	1,7	1	10,7	11	24,9	5	22,2	2	7,7
	5	1	3,6	9	31,8	20	40,7	9	28,7	1	22
	6	5	5,1	5	7,9	8	10,1	6	8,9	5	6,9
	7	5	5	7	8,3	8	10,5	6	8,9	5	6,6
	8	5	6,6	8	11	11	12,7	10	11,8	7	9,1
	9	209	210,4	211	214,8	213	218,5	213	216,9	211	215,4
	10	1	1,3	11	14	16	18,2	2	10,3	1	5,5
	11	7	7,9	9	12	11	14,2	8	11,3	9	9,8
Bin Packing	0	0,00214	0,00324	0,00593	0,00607	0,00220	0,00294	0,00229	0,00402	0,00594	0,00612
	1	0,00277	0,00329	0,00358	0,00547	0,00254	0,00294	0,00273	0,00374	0,00365	0,00595
	2	0,01923	0,02024	0,01837	0,01894	0,02009	0,02124	0,01986	0,02113	0,01818	0,01881
	3	0,01873	0,02010	0,01872	0,01875	0,01937	0,02042	0,01900	0,02159	0,01871	0,01886
	4	0,00034	0,00161	0,00456	0,00484	0,00034	0,00249	0,00034	0,00395	0,00501	0,00529
	5	0,00306	0,00328	0,00347	0,00376	0,00306	0,00314	0,00306	0,00337	0,00387	0,00402
	6	0,01001	0,01428	0,00496	0,00543	0,00994	0,01384	0,01055	0,01397	0,00485	0,00528
	7	0,01062	0,01220	0,00809	0,01234	0,01017	0,01224	0,01014	0,01281	0,00829	0,01326
	8	0,03975	0,04116	0,03266	0,03318	0,03733	0,04138	0,03188	0,03421	0,03696	0,04025
	9	0,00042	0,00207	0,00051	0,00136	0,00042	0,00174	0,00042	0,00137	0,00054	0,00202
	10	0,10828	0,10828	0,10828	0,10828	0,10828	0,10828	0,10828	0,10828	0,10830	0,10834
	11	0,00126	0,00213	0,00427	0,00542	0,00036	0,00177	0,00033	0,00218	0,00461	0,00858
Permutation Flowshop	0	6245	6257,4	6257	6281,9	6245	6285,2	6263	6290,1	6272	6286,7
	1	6212	6223,5	6232	6247,2	6227	6247,8	6235	6251,4	6217	6244,3
	2	6284	6307	6315	6332,2	6300	6325,2	6302	6320,5	6299	6327,1
	3	6287	6307,4	6303	6322,3	6303	6336,3	6303	6340,6	6303	6337,6
	4	6343	6361,2	6352	6369,5	6342	6365,2	6350	6367,8	6344	6364,4
	5	10485	10492,9	10487	10495,4	10494	10496,4	10494	10496,6	10485	10495,2
	6	10922	10922,2	10922	10922,7	10922	10922,8	10922	10922,6	10922	10922,6
	7	26197	26231,5	26199	26250,7	26193	26227,6	26191	26241	26215	26241,3
	8	26674	26732,1	26721	26749,5	26687	26758,2	26694	26738,3	26689	26732,3
	9	26553	26584	26549	26596,1	26546	26595,5	26549	26596,6	26549	26594,9
	10	11308	11317,6	11319	11345,7	11308	11336,5	11319	11344,3	11313	11343,6
	11	26464	26507,8	26537	26586	26529	26578,3	26527	26582,2	26486	26564
Personnel Scheduling	0	3294	3308,4	3290	3301,9	3291	3311,1	3292	3315,2	3292	3311,2
	1	1998	2177,7	1978	2101,7	2120	2267,3	1995	2117,8	2078	2308,6
	2	300	342	335	356	280	313	300	328	290	315
	3	11	16,8	14	18	18	25,3	12	18,4	12	19,3
	4	15	18,6	15	19,5	23	27,1	14	18,5	21	25,5
	5	16	18,6	16	20	19	26,3	17	21	19	23,9
	6	1093	1115,6	1101	1114,1	1102	1169,9	1100	1105,9	1098	1139,1
	7	2149	2221,5	2191	2219,7	2155	2264,9	2186	2227,2	2157	2249,5
	8	3133	3181,3	3139	3204,1	3121	3196	3138	3169,8	3135	3215,7
	9	8978	9337,9	9503	9658,1	9194	9385,5	9524	9721,9	9253	9343,1
	10	1470	1559,7	1400	1476,4	1530	1840,2	1515	1590,3	1523	1716,4
	11	280	303	300	323,5	280	298,5	300	312	280	298,5
Travelling Salesman	0	48194,92	48194,92	48194,92	48222,47	48194,92	48222,47	48194,92	48213,29	48194,92	48204,10
	1	107294,71	108541,14	107215,30	108997,10	108121,24	109024,87	107215,30	109119,62	107215,30	108842,12
	2	6795,97	6801,87	6795,97	6807,32	6795,97	6802,83	6795,97	6803,26	6795,97	6805,95
	3	41925,50	41970,03	41935,03	41987,48	41934,72	41995,07	41918,65	41989,03	41912,61	41985,74
	4	8846,81	8863,45	8852,68	8875,22	8855,73	8869,37	8848,67	8870,53	8848,67	8865,17
	5	56938,34	57136,71	57161,45	57322,03	56971,70	57353,65	56991,95	57315,07	57142,49	57322,33
	6	52582,83	52862,89	51928,01	53005,08	52684,92	53293,96	52497,38	53236,37	52389,69	53154,93
	7	65651,15	66234,31	66039,03	66315,11	65562,45	66153,53	65800,85	66153,77	65730,81	66078,90
	8	20587684,00	20628368,50	20642078,05	20708254,72	20602094,23	20692600,71	20579110,06	20667170,61	20556131,96	20654062,46
	9	664479,75	667026,02	666798,86	668122,40	666567,89	667311,87	667666,43	668826,31	665877,60	667219,99
Vehicle Routing	0	4164,31	4918,88	4152,99	4754,00	5092,29	5148,17	5075,22	5147,68	5082,57	5129,20
	1	20652,47	20654,19	20650,80	20750,46	20676,70	21372,07	20656,49	21267,62	20672,64	21778,58
	2	12361,88	13040,32	12289,92	12730,22	13316,18	14094,39	13300,80	13778,65	13342,02	13867,59
	3	5265,99	5300,69	5288,31	5324,26	5341,89	5561,12	5315,48	5355,97	5297,97	5435,10
	4	13287,09	14079,33	13267,47	13787,66	14299,57	14928,77	14266,88	14612,32	14263,55	14489,12
	5	145321,58	146507,32	142479,08	144035,49	146775,65	148934,00	144153,69	146942,49	144062,78	149416,32
	6	59555,53	65282,61	56950,10	58007,13	58457,22	60492,64	57327,48	60245,43	58045,27	60131,58
	7	159878,05	160253,58	158404,97	159828,78	160645,44	162229,53	159321,12	160608,18	160735,40	161714,55
	8	146523,89	152184,37	142726,85	144540,10	151237,31	155370,15	146106,56	151197,18	152788,51	155142,48
	9	146331,27	148737,45	144534,23	145606,71	146805,26	147785,55	145243,45	146852,47	146614,73	147457,57

A. Improvement provided based on execution time

Table XIV shows the performance difference between running ADHS-AILLA for 10 minutes and for 1 hour. The performance difference between the 10 minutes and 1 hour experiments is limited considering the permutation flowshop scheduling and travelling salesman problems. For these problems, finding a good quality solution appears to be very easy after short amount of time and it becomes hard to improve. The relative improvement provided after 1 hour is quite large for the max SAT problem compared to 10 minutes, however the objective values are small in values. This also shows that for the corresponding SAT instances, the proposed approach finds high quality solutions at the very beginning of the search. A similar case is valid for some personnel scheduling problem instances. The gap is relatively large due to the particular fitness function used to evaluate the bin packing problem solutions. This indicates that there is large room for improvement after 10 minutes running time. The improvement provided after 10 minutes is small for most of the vehicle routing problem instances. For instance 6, the average fitness after 1 hour of execution appears to be worse than the 10 minutes case. The underlying reason is that the hyper-heuristic prematurely converged during certain runs.

B. ADHS

Fig. 4 shows for each problem domain how many times each heuristic was applied during a 10 minutes run. For the bin packing problem, there is a trend regarding the number of calls. A hill climber (LLH_6) is called more frequently than the others. It is followed by the ruin-recreate heuristics (LLH_2, LLH_1) and a mutational heuristic (LLH_0). The distribution of calling different heuristics change over time in consideration of the max SAT problem. The frequently selected heuristics are listed from most to the least frequent as follows: LLH_4, LLH_3 and LLH_5 that are hill climbers. A crossover operator

TABLE X
THE EXPERIMENTAL RESULTS BY SR HYPER-HEURISTICS WITH 1 HOUR OF EXECUTION TIME

	Inst.	SR-AILLA		SR-GD		SR-IE		SR-LATE		SR-SA	
		min	avg	min	avg	min	avg	min	avg	min	avg
Max SAT	0	4	6,6	118	182,8	8	14	6	9,3	2	4,3
	1	20	21,9	183	227,9	19	26,5	22	24,2	18	20,5
	2	14	17,4	166	229	15	20	15	17,9	15	16,5
	3	5	8,9	61	78,4	20	29	7	10,2	2	5,9
	4	5	7,1	60	79,8	36	41,1	5	8,5	1	3,9
	5	10	14,8	152	186,3	34	53,9	8	12,7	5	10,8
	6	5	5,6	13	17,1	7	9,9	8	9,7	5	5,6
	7	5	6,1	14	16,1	7	9,5	6	7,8	5	6
	8	5	7,5	15	21,8	8	11,4	9	10,3	5	7,7
	9	209	211,2	231	243	216	218,6	211	215,4	209	212
	10	3	11,1	105	134,7	24	28,8	6	11,9	1	2
	11	7	7,7	20	23,8	11	12,9	8	10,9	7	8,4
Bin Packing	0	0,00282	0,00558	0,03183	0,03512	0,00281	0,00574	0,00280	0,00551	0,00778	0,01055
	1	0,00329	0,00412	0,03213	0,03518	0,00346	0,00485	0,00335	0,00483	0,00766	0,00823
	2	0,01926	0,02074	0,02577	0,02665	0,02253	0,02372	0,02188	0,02320	0,01849	0,01889
	3	0,01937	0,02034	0,02844	0,02934	0,02349	0,02534	0,02155	0,02525	0,01871	0,01872
	4	0,00034	0,00034	0,01923	0,01991	0,00034	0,00119	0,00456	0,00463	0,01193	0,01263
	5	0,00306	0,00306	0,01821	0,01852	0,00314	0,00332	0,00358	0,00376	0,00966	0,01081
	6	0,01005	0,01185	0,09091	0,09738	0,00958	0,01232	0,00548	0,00937	0,00999	0,01250
	7	0,01329	0,01464	0,09546	0,10201	0,01334	0,01525	0,01412	0,01595	0,01437	0,02011
	8	0,05466	0,05719	0,07386	0,07619	0,05009	0,05288	0,05833	0,06008	0,03956	0,04078
	9	0,01159	0,01408	0,02458	0,02550	0,01247	0,01427	0,01461	0,01523	0,01127	0,01250
	10	0,10836	0,10839	0,11845	0,11980	0,10841	0,10844	0,10869	0,10875	0,11022	0,11062
	11	0,02093	0,02427	0,04205	0,04363	0,02077	0,02572	0,02991	0,03206	0,01979	0,02153
Permutation Flowshop	0	6301	6331,5	6364	6376,3	6319	6338,6	6298	6324,5	6290	6326,8
	1	6266	6286,5	6293	6322,3	6271	6308,1	6248	6294,5	6289	6303,6
	2	6328	6351,4	6384	6398,7	6318	6357,7	6338	6354,1	6336	6354,8
	3	6323	6347,8	6361	6378,3	6323	6354,5	6326	6350	6327	6354,9
	4	6377	6399,5	6447	6460	6377	6400,5	6377	6400,1	6367	6396,3
	5	10499	10515,6	10515	10529,4	10501	10528,3	10528	10539,9	10507	10534,8
	6	10922	10922,7	10947	10963,2	10922	10927,5	10922	10927,5	10923	10935
	7	26272	26328,8	26380	26430,6	26360	26451,4	26327	26384,9	26389	26474,4
	8	26764	26814,8	26869	26914,2	26873	26932,8	26787	26834,8	26841	26919,2
	9	26612	26647	26713	26732,2	26697	26754,9	26665	26697,7	26674	26748,3
	10	11377	11412,5	11454	11491,9	11389	11464,2	11432	11461,3	11426	11481,1
	11	26574	26629,4	26662	26703,8	26718	26753,9	26608	26663,4	26649	26721,8
Personnel Scheduling	0	3296	3326,6	3313	3343,3	3295	3336,9	3294	3317,3	3295	3328,9
	1	2120	2377,4	2310	2410,4	2260	2399,2	2040	2159,9	2130	2409,3
	2	350	404	360	387	350	1004,5	340	364	485	1210
	3	14	18,9	18	28,5	17	24,8	11	16,8	18	23,4
	4	14	20,7	31	33,6	17	26,5	17	19	22	27,5
	5	15	20,1	24	32,1	22	27	18	19,9	20	27
	6	1095	1123,8	1155	1241,3	1099	1106,3	1097	1115,7	1092	1125,3
	7	2142	2203,4	2321	2437	2140	2234	2166	2217,9	2156	2235,1
	8	3121	3224,7	3354	3645,6	3135	3248,4	3157	3188,2	3132	3202,3
	9	9225	9418,5	11879	15854,7	9325	9371,8	9600	9710,6	9204	9370,6
	10	1545	1690,1	1635	1832	1680	2003,7	1480	1606,7	1430	1830
	11	350	364,5	340	352	355	1348	320	341,5	370	798,5
Travelling Salesman	0	48783,64	49156,77	48599,79	48915,61	49668,09	50946,67	49107,00	49833,44	49259,48	50297,25
	1	110564,87	111419,31	110713,99	111896,49	112419,39	113973,87	112963,60	114513,31	112212,38	114839,34
	2	6926,94	6959,94	6952,33	6970,01	7011,57	7083,83	6979,99	7016,49	7011,57	7048,73
	3	42715,80	43049,30	42835,63	43068,60	43408,08	43744,32	43223,11	43403,01	43385,76	43738,36
	4	9067,84	9110,53	9055,54	9087,11	9186,73	9248,54	9091,63	9156,40	9171,94	9207,92
	5	59412,81	59735,21	59295,64	59634,71	60087,77	60807,46	59571,67	60087,70	59996,40	60635,19
	6	53557,30	54039,64	54840,37	55783,46	53722,65	54905,25	54695,22	55473,46	54434,51	54979,89
	7	68471,97	68999,59	69383,13	69711,01	68790,31	69962,01	69629,79	70178,41	68881,81	69314,48
	8	21119457,88	21316901,87	21153503,04	21262559,87	21255472,60	21401367,92	21145364,26	21322066,84	21268106,35	21448099,47
	9	674371,48	675933,05	675305,29	676503,62	676435,22	678735,34	674723,33	676058,97	675565,00	678768,49
Vehicle Routing	0	4183,81	4846,15	5216,65	5274,65	5178,67	5236,05	5147,14	5180,91	5126,35	5186,24
	1	20651,10	20652,89	20667,04	21476,93	20663,28	21176,92	20661,69	21170,60	20656,64	21364,02
	2	12289,05	12925,35	13342,67	14528,52	13358,84	14107,57	12402,50	13267,71	13331,80	13584,72
	3	5309,32	5333,93	5360,17	5507,90	5383,55	5596,26	5373,71	5408,55	5316,05	5388,25
	4	13308,27	14076,91	15325,53	15658,31	14311,70	15147,16	13320,87	14211,03	14280,54	14411,75
	5	143908,77	146723,57	261278,71	267206,18	142509,88	147480,42	143871,49	145783,48	143904,85	145716,25
	6	58849,50	59651,23	107161,68	113411,06	58342,60	59458,97	60844,93	61663,48	57397,60	59271,45
	7	159753,92	161198,40	197189,41	204495,39	159959,74	161332,56	159476,14	160247,01	160781,49	161358,38
	8	150175,78	153348,54	237084,51	241517,40	148801,51	153461,12	145646,27	148316,12	150398,65	152841,70
	9	145941,03	147019,25	183843,11	189242,81	145716,09	146924,70	144650,00	145780,46	144781,65	146554,35

TABLE XI
THE SCORES OF THE TESTED HYPER-HEURISTICS AFTER 10 MINUTES OF EXECUTION BASED ON THE CHESC SCORING SYSTEM (THE HIGHEST POSSIBLE SCORE IS 680). THE RESULTS OF SR HYPER-HEURISTICS EXCEPT SR-AILLA WERE TAKEN FROM [61]

Hyper-heuristic	Max SAT	Bin Packing	Perm. Flowshop	Pers. Scheduling	TSP	VRP	OverAll
ADHS-AILLA	116	77	95	89	85	56	518
ADHS-GD	29	72	93	75	60	89	418
ADHS-IE	8	81	76	44	65	28	302
ADHS-LATE	29,5	74	53	75	56	69	356,5
ADHS-SA	59	68	73	54,5	64	36	354,5
SR-AILLA	73	33	38	41	25	59	269
SR-GD	0	2	7	6	20	0	35
SR-IE	12,5	40	4	30	2	1	103,5
SR-LATE	58	17	24	26	13	20	158
SR-SA	83	4	5	27,5	0	18	137,5

TABLE XII
THE SCORES OF THE TESTED HYPER-HEURISTICS AFTER 1 HOUR OF EXECUTION BASED ON THE CHESC SCORING SYSTEM (THE HIGHEST POSSIBLE SCORE IS 680)

Hyper-heuristic	Max SAT	Bin Packing	Perm. Flowshop	Pers. Scheduling	TSP	VRP	OverAll
ADHS-AILLA	113	70	118	87	95	52	535
ADHS-GD	31	67	62,5	72	51,5	94	378
ADHS-IE	10	76	71	41	54,5	10	262,5
ADHS-LATE	31	62	62	72	55	39	321
ADHS-SA	63,5	53	81	49	74	20	340,5
SR-AILLA	76	53	34,5	40	26	59	288,5
SR-GD	0	0	5	5	20	1	31
SR-IE	10	31	5,5	20,5	3	21	91
SR-LATE	44,5	24	21,5	62	8	51	211
SR-SA	89	32	7	19,5	3	43	193,5

TABLE XIV
THE % PERFORMANCE DIFFERENCE BETWEEN 10 MINUTES AND 1 HOUR CASES BASED ON THEIR AVERAGE RESULTS BY ADHS-AILLA

Inst.	MSAT	BP	FS	PS	TSP	VRP
0	35,56	7,41	0,23	0,72	0,00	7,05
1	26,04	8,40	0,23	7,18	0,44	0,51
2	4,37	7,73	0,14	8,31	0,11	0,12
3	5,73	37,14	0,52	23,98	0,13	0,61
4	35,60	37,04	0,40	27,34	0,14	0,75
5	7,43	72,73	0,04	28,19	0,59	2,02
6	15,78	3,77	0,03	4,58	0,44	-9,12
7	22,53	7,41	0,25	2,68	0,70	0,63
8	1,87	7,04	0,30	2,38	0,78	1,03
9	52,90	0,75	0,35	2,88	0,42	0,63
10	0,00	40,91	0,41	6,65	-	-
11	30,06	1,25	0,46	9,69	-	-

is selected most after these hill climbers. For the permutation flowshop scheduling problem, heuristics can be considered as divided into groups. After 400 seconds, one crossover operator (LLH_{13}) was applied most often. Other crossover operators are also called very frequently after LLH_{13}. Only a ruin-recreate heuristic (LLH_5) is called more than 1000 iterations for the personnel scheduling problem. In the case of the travelling salesman problem, two crossover operators (LLH_9, LLH_{12}) are chosen more often than others. Also for the vehicle routing problem, two heuristics take the lead, i.e. two mutational heuristics (LLH_0, LLH_1).

1) Heuristic set size: Fig. 5 depicts the changes of the heuristic set size for each problem domain. The fluctuations of the heuristic set size are very frequent concerning the bin packing, max SAT and vehicle routing problems. Changes mostly take place due to the presence of fast heuristics in the heuristic subset. The set size changes are relatively slower with the remainder of the problem domains. A set of four heuristics is most of the time sufficient when dealing with the bin packing problem. This value is generally around 5 for the max SAT problem. The heuristic set size decreases till 5 during the search by excluding 10 heuristics at the same time in the case of the permutation flowshop scheduling problem. The size of the heuristic set associated with the personnel scheduling problem oscillates around 5. A set with 5 or 6 heuristics is often enough for the travelling salesman problem. For the vehicle routing problem, the size of the heuristic set is generally 4 and for some cases, the set size even reduced to 2.

These results indicate that there is no need to use the whole heuristic set during the entire search process. The subset

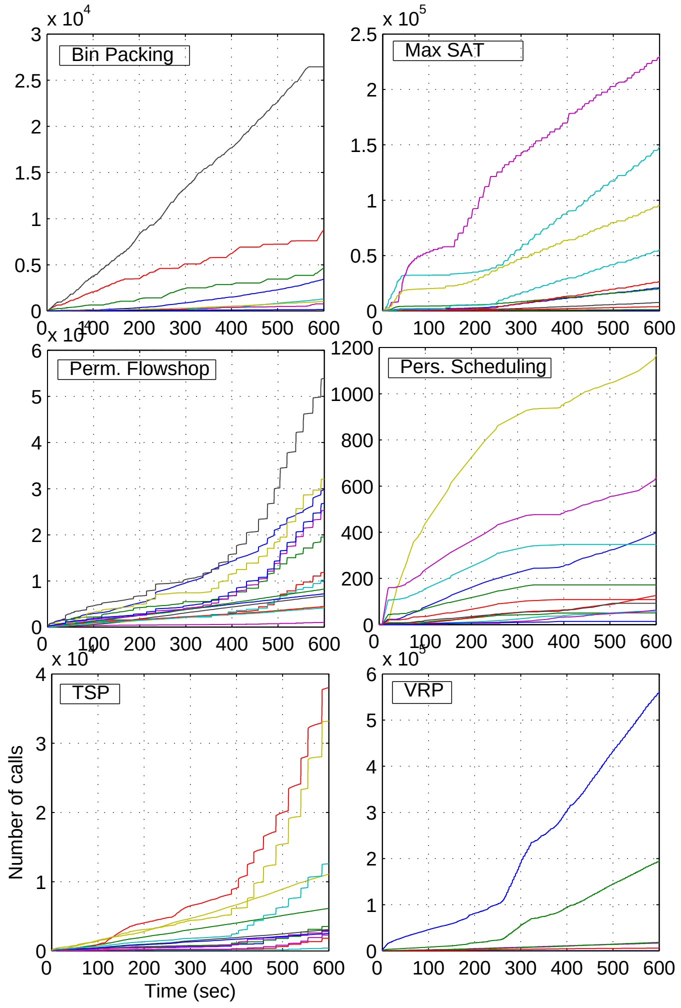

Fig. 4. The number of calls for each heuristic on each problem domain by ADHS-AILLA with 10 minutes of execution time (a run on one sample instance represents each domain)

selection or heuristic exclusion strategy effectively determines good heuristic subsets and this reduces the complexity of the heuristic selection operation. However, it should be noted that there is no one heuristic subset which always works well. Therefore, it is important to appoint different heuristic subsets for different parts of the search.

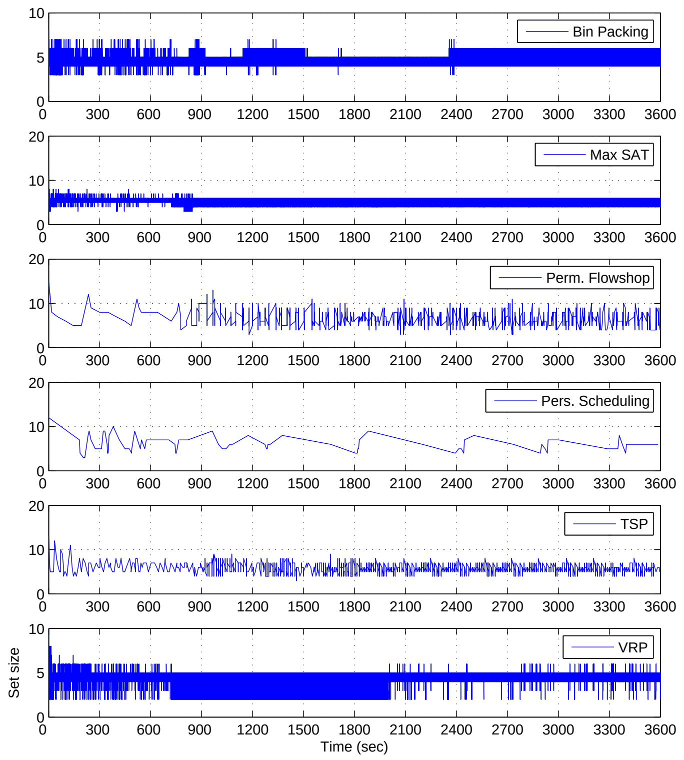

Fig. 5. The heuristic set size changes on each problem domain by ADHS-AILLA with 1 hour of execution time (a run on one sample instance representing each domain was used)

2) QI changes: Fig. 6 presents the QI values for each bin packing heuristic during a 10 minutes run. Among them, one ruin-recreate heuristic (LLH_2), one mutational heuristic (LLH_3) and only crossover operator (LLH_7) have high QI values during almost the whole run. One ruin-recreate heuristic (LLH_1) and two hill-climbers (LLH_4, LLH_6) have relatively worse QI values. Two mutational heuristics (LLH_0, LLH_5) generally get the lowest QI values. It is apparent that all the QI values tend to change, so high QI values may turn into low ones for some heuristics during the course of the search, and vice versa. This means that the hyper-heuristics' strength brings a performance change while solving a problem instance.

C. Relay Hybridisation

Fig. 7 presents heuristic pairs that explored new best solutions during 1 hour. In the light of the bin packing problem, such heuristic pairs were rarely identified during the first 10 minutes. After that time, almost no pair contributed to the optimisation. During the first half of the search, hybridisation frequently found new best solutions while solving the max SAT problem instances. From the given heuristic set, a mutational heuristic (LLH_6) was used as the first heuristic to apply. Two mutational heuristics (LLH_0, LLH_4), and one hill climber (LLH_8) were utilised as the second heuristics. All the hill climbers (LLH_7,LLH_8,LLH_9,LLH_{10}) given to solve the permutation flowshop scheduling problem discovered new best solutions as the second heuristics. Except for three mutational heuristics (LLH_2,LLH_3,LLH_4), all other heuristics were

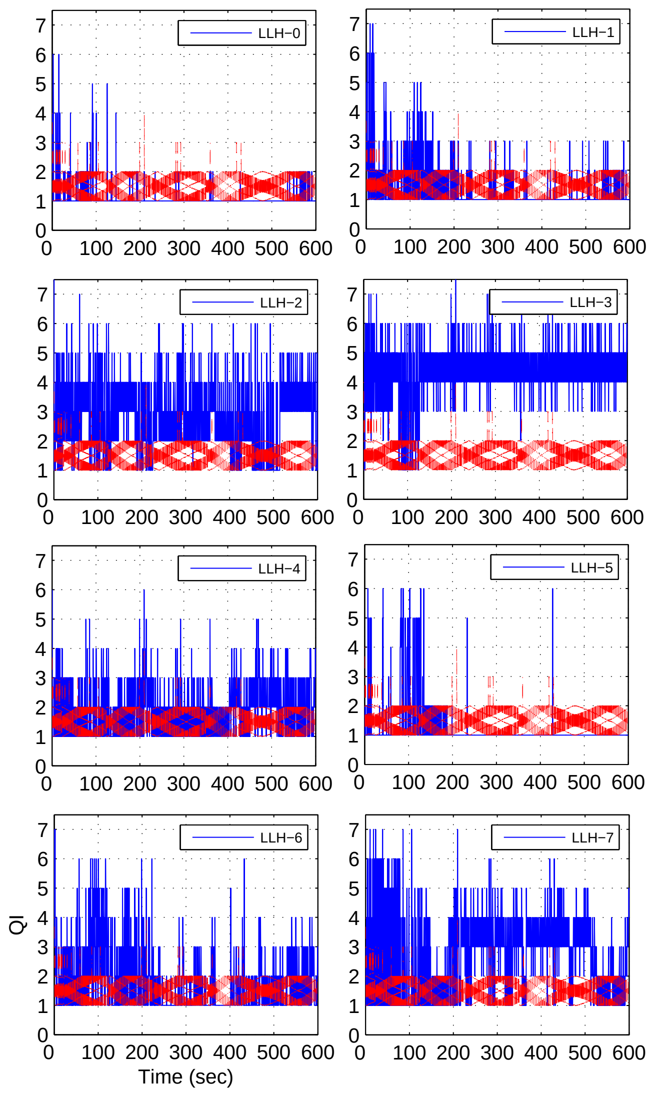

Fig. 6. QI changes of the bin packing heuristics by ADHS-AILLA with 10 minutes of execution time (the red dotted lines show avg)

used as the first ones of the pairs. For the personnel scheduling problem, two hill climbers (LLH_3,LLH_4) were used as the second heuristics. The rest of the heuristics, except a crossover operator (LLH_9), was used to change the solution before the second heuristics were applied. These heuristics found new best solutions only during the first half of the execution time. The only ruin-recreate heuristic (LLH_5) was mostly used as the first heuristic with regard to the travelling salesman problem. All the available hill climbers (LLH_6,LLH_7,LLH_8) were used as the second heuristics. Among them, especially LLH_8 successfully found many new best solutions via relay hybridisation. A crossover operator (LLH_6) was utilised as the first heuristic for most of the cases on the subject of the vehicle routing problem. A mutational heuristic (LLH_0) and two hill climbers (LLH_4, LLH_8) were generally applied as the second heuristics.

Fig. 8 demonstrates the ratio between the number of iterations spent by relay hybridisation compared to the single heuristic selection. This ratio changes over time for al; the

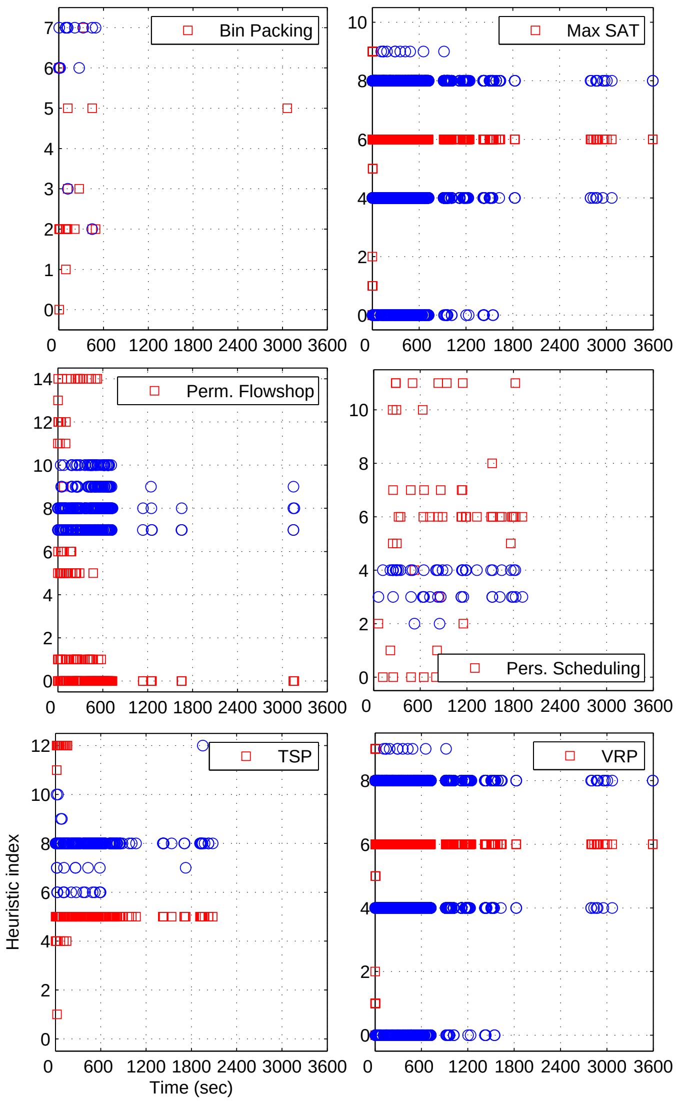

Fig. 7. Heuristic pairs yielding new best solutions by relay hybridisation using ADHS-AILLA for 1 hour of execution time (squares show the first applied heuristics, circles indicate the heuristics applied afterwards)

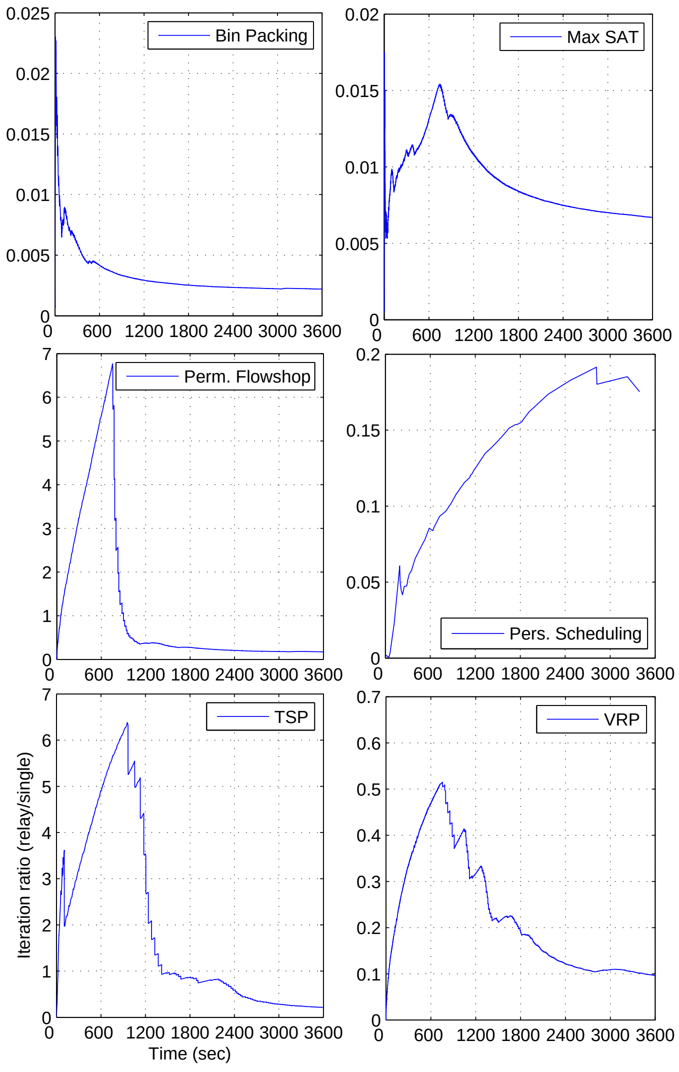

Fig. 8. Number of iterations ratio between relay hybridisation and single heuristic selection (relay/single)

problem domains. For the permutation flowshop scheduling problem, the ratio increases during the first 10 minutes and it is higher than 1. This shows that for this problem domain, relay hybridsation is preferred over single heuristic selection. Afterwards, it decreases and around 1000 seconds later, its value goes under 1 since it is more profitable to apply single heuristics. With the travelling salesman problem, a similar trend occurs. This ratio decreases to very low value due to the fast heuristics with very high improvement capabilities for the bin packing problem. For the max SAT problem, the ratio slightly increases, but decreases towards the end of the search. The amount of incrementation is relatively larger while solving the vehicle routing problem, but it also starts to decrease after almost 10 minutes. The ratio constantly increases for the personnel scheduling problem to about 0.2.

For all the tested problem domains, various heuristic pairs that can explore new best solutions were detected. Hence, this approach can be considered as a method to increase the quality of a heuristic for the mentoring task. However, adding new heuristics makes the heuristic selection operation more complex. The decision rule provided in the relay hybridisation solves this problem by deciding whether applying only one heuristic or heuristic pairs. For instance, ADHS-AILLA prefers heuristic pairs over single heuristic since using them is more profitable in comparison single heuristic for the permutation flowshop scheduling problem. Conversely, ADHS-AILLA applies single heuristics more since they are already better than heuristic pairs, even though it found very effective heuristic hybridisations for the bin packing problem. This means that heuristics should be chosen depending on both their improvement and speed traits.

D. AILLA

Fig. 9 shows the iteration limit (k) changes. k increases over time with some fluctuations in certain time limits while solving the bin packing problem instances. The hyper-heuristics wait about 500 iterations before diversifying the search during

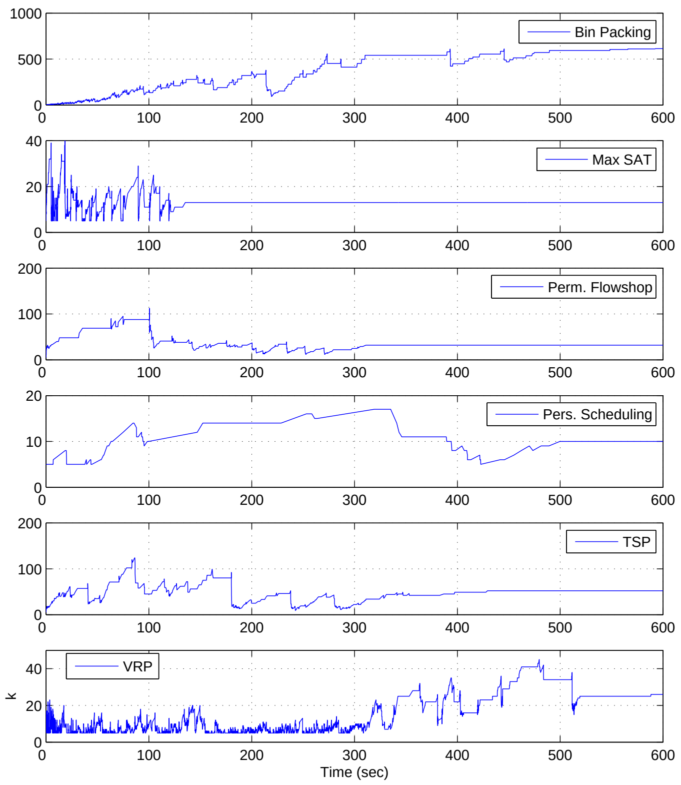

Fig. 9. Iteration limit (k) changes on each problem domain (a run on one sample instance representing each domain was used)

later iterations. For the max SAT problem, k constantly changes during the first 100 seconds, after that it is almost stable due to the fast improvement provided by the hyper-heuristic. In the case of the permutation flowshop problem, changes on k are limited to the first half of the running time. Due to the limited number of improvements and slow heuristics, the value of k changes over the whole period for the personnel scheduling problem. Considering the travelling salesman problem, k reaches 100 after several fluctuations and stabilizes after 400 seconds. Even though the variation of k is limited for the vehicle routing problem, changes are nevertheless frequent. This shows that improving solutions for 300 seconds is relatively easy and that this is mostly due to the re-initialisation procedure. After that time, the number of improvements decreases.

From the competition point of view, success of the competing methods combining mutational and ruin-recreate heuristics with hill climbers can be deduced from our relay hybridisation results. As expected, for most of the cases, the detected heuristic pairs that find new best solutions are in the form of perturbing a solution first then improving the new solution using a hill climber. However, there are certain cases such as the bin packing problem, hybridisation slows down the hyper-heuristic and causes missing fast and effective single heuristics. As discussed in Section 2, it is important to determine effective heuristics according to their improvement characteristics. However, it is even more important to consider these abilities along with their speed for better judgement.

E. Re-initialisation

When good quality solutions are detected in a fast manner and while there is still time to make improvements, ADHS-AILLA randomly re-inisialises the solution is randomly re-initialised. Fig. 10 presents the changes to the best fitness with re-initialisations by ADHS-AILLA. For the max SAT problem, the frequency of re-initialisation is very high. After almost 100 seconds, re-initialisation stops and the hyper-heuristic starts working to improve the overall best solution found after all performed re-initialisations. The vehicle routing problem was subject to several re-initialisations but fewer than the max SAT domain. Re-initialisation is much less frequent for the rest of the problems.

These results disclose that even a such a naive approach should be adaptive for different heuristics sets associated with distinct problems. There a few major points derived from these results. The first point is that how easy to find a good quality solutions for a particular problem instance. The other point is related to the continuity of improvement. For instance, the re-initialisation activities for the max SAT problem are very often. On the other hand, the number of generating new initial solutions are very limited in the case of the bin packing problem. ADHS-AILLA immediately finds good quality solutions due to the capabilities of the given heuristics and the structure of the solution space, so the fitness landscapes owned by each heuristic. Then, improving these quickly found good solutions is very hard. The fitness landscapes belonging to the bin packing problem domain is also appropriate for finding solutions in a fast manner. However, the improvement process is continuous. Therefore, there is no need to call the re-initialisation method as frequently as the max SAT case. This re-initialisation mechanism can deal with such differences in an efficient way.

F. CHeSC 2011

The cross-domain heuristic search challenge (CHeSC) 2011 is the first competition to show the level of generality of a high-level approach across multiple problem domains. CHeSC 2011 organisers provided four problem domains along with heuristic sets as a testbed for selection hyper-heuristics. The problem domains given prior to the competition were max SAT, bin packing, permutation flowshop scheduling and personnel scheduling problems. The competitors performed a set of experiments on these domains to examine the performance of their hyper-heuristics. For the competition, the testbed was extended with two hidden problem domains, namely travelling salesman and vehicle routing problems. For the problem domains announced before the competition, three available instances were randomly selected. In addition, two hidden instances were introduced. As a consequence, 5 instances from 6 problem domains were used as the test-bed. In this setting, the highest possible score for an algorithm is 300 based on the formula 1 scoring system.

In Table XV, the ranking and scores of the 20 competing algorithms are presented. Each algorithm runs 31 times for 10 minutes. These results reveal a clear performance difference between our approach, ADAPHH (ADHS-AILLA), and

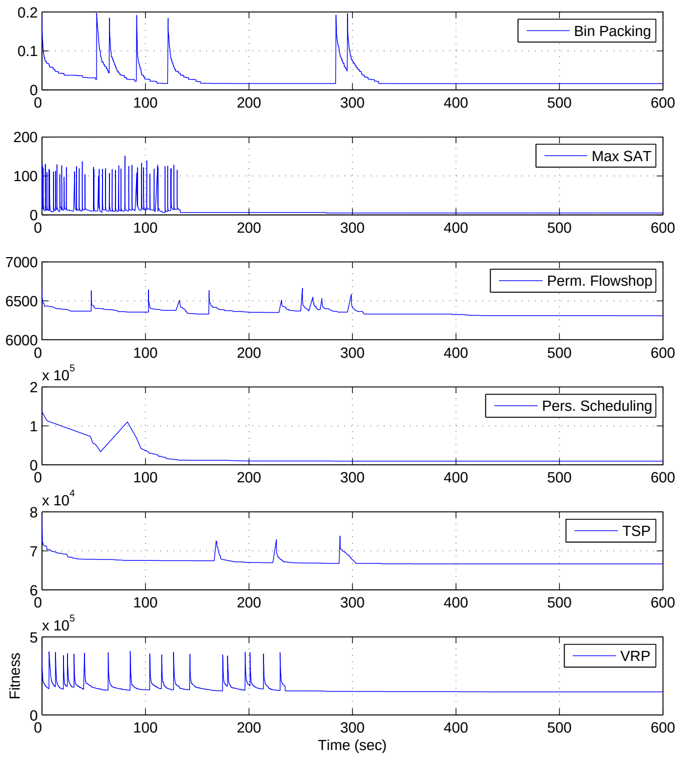

Fig. 10. Changes on the best fitness found by ADHS-AILLA on each problem domain

the other competing algorithms. This is a vital indicator in connection with the generality of the corresponding method. More detailed results are available at the competition website[1]. Problem-wise, the proposed approach outperforms the other algorithms for max SAT, 1D bin packing and travelling salesman. It comes second for permutation flowshop scheduling, 5th for vehicle routing and 10th for the personnel scheduling problem. Among the unseen problem domains, the travelling salesman problem is a good example to show the adaptive capabilities of this hyper-heuristic. On the other hand, the results on vehicle routing problems indicate that there is still room for improvement on its generality level. Its relatively worse performance for the personnel scheduling is caused by the limited time available to the learning process given the speed of the heuristics.

IX. Conclusion

A selection hyper-heuristic, ADHS-AILLA, accommodating various evolvable mechanisms has been developed in the present study. Each sub-mechanism focuses on a specific part of the hyper-heuristic. One mechanism, i.e. the adaptive dynamic heuristic set (ADHS) strategy, is dedicated to discover the best heuristic subsets during different search regions. In addition, a heuristic selection rule was determined for choosing heuristics from these subsets. Another algorithmic contribution is responsible for finding effective heuristic pairs yielding new best solutions. These methods were supported by

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

TABLE XV
CHeSC 2011 competition scores (Algorithms are sorted from the best to the worst based on their overall scores)

Algorithm	Score	MSAT	BP	FS	PS	TSP	VRP
ADAPHH	181	34.75	45.0	37.0	9.0	40.25	15.0
VNS-TW	134	34.25	3.0	34.0	39.5	17.25	6.0
ML	131.5	14.5	12.0	39.0	31.0	13.0	22.0
PHUNTER	93.25	10.5	3.0	9.0	11.5	26.25	33.0
EPH	89.75	0.0	10.0	21.0	10.5	36.25	12.0
HAHA	75.75	32.75	0.0	3.5	25.5	0.0	14.0
NAHH	75	14.0	19.0	22.0	2.0	12.0	6.0
ISEA	71	6.0	30.0	3.5	14.5	12.0	5.0
KSATS-HH	66.5	24.0	11.0	0.0	9.5	0.0	22.0
HAEA	53.5	0.5	3.0	10.0	2.0	11.0	27.0
ACO-HH	39	0.0	20.0	9.0	0.0	8.0	2.0
GenHive	36.5	0.0	14.0	7.0	6.5	3.0	6.0
DynILS	27	0.0	13.0	0.0	0.0	13.0	1.0
SA-ILS	24.25	0.75	0.0	0.0	19.5	0.0	4.0
XCJ	22.5	5.5	12.0	0.0	0.0	0.0	5.0
AVEG-Nep	21	12.0	0.0	0.0	0.0	0.0	9.0
GISS	16.75	0.75	0.0	0.0	10.0	0.0	6.0
SelfSearch	7	0.0	0.0	0.0	4.0	3.0	0.0
MCHH-S	4.75	4.75	0.0	0.0	0.0	0.0	0.0
Ant-Q	0	0.0	0.0	0.0	0.0	0.0	0.0

a move acceptance strategy (AILLA) for accepting or rejecting the visited solutions by the chosen heuristics and heuristic pairs. Furthermore, a re-initialisation mechanism performing depending on feedback from the move acceptance strategy was utilised. For improving the performance of the hyper-heuristic even more, a number of decision mechanisms were instantiated to activate or deactivate these sub-mechanisms if required. The resulting mechanisms were used in coordination within a hyper-heuristic framework. The performance of the hyper-heuristic across multiple domains, namely max SAT, bin packing, permutation flowshop scheduling, personnel scheduling, travelling salesman and vehicle routing, revealed that ADHS-AILLA is able to deliver high quality results for different instances of the problem domains, the corresponding heuristic sets and different execution time. In particular, the algorithm here presented won the first international Cross Domain Heuristic Search Challenge 2011.

A. *Level of generality*

For the generality of a search algorithm, it is required to eliminate problem-dependent elements as much as possible. The cross-domain search challenge 2011 competition provides the opportunity of solving a set of instances from different problems with no domain knowledge. In this competition, particular information on the subject of the heuristic types was released. Even if this information is withheld as is the case in many hyper-heuristic studies, it can nevertheless be used for the competition domains to deliver better quality solutions. Most of the competing algorithms used the information about heuristic types to devise memetic algorithm or iterated local search kind of approaches. The hyper-heuristic proposed here ignores this information and defines a set of heuristic types to generalise the hyper-heuristic by definition. These types were specified in regard to heuristics' improvement skills and they are valid for all possible heuristics. Hence, this approach helps to generalise the applicability of the hyper-heuristic across various heuristic sets involving different heuristic types.

Adaptiveness and coordination are the primary concepts that need to be considered for the purpose of generality in the hyper-heuristic research. These concepts have been materialised within the hyper-heuristic sub-mechanisms. In that sense, certain components to decide about the sub-mechanisms' lifetime, frequency of usage, status regarding whether they are active or passive are required. Furthermore, since there is no real parameter-free or rule-free algorithm, it is highly required to manage or adapt the corresponding parameters during the run. As a consequence, the hyper-heuristic developed should be able to properly change its behaviour for different problem sets and heuristic sets via various adaptation and decision mechanisms.

B. Future research

In the future, the user-dependency of the hyper-heuristic will be decreased by increasing its adaptive behaviour with respect to the changing requirements during the run. Then, the additional sub-components as well as more effective decision mechanisms will be implemented. Furthermore, the performance of the hyper-heuristic will be investigated over other problem domains, especially real world problems.

References

[1] 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 *Proc. 12th Conf. on Evol. Comput. in Comb. Opt.*, ser. LNCS, 2012.

[2] D. Ouelhadj and S. Petrovic, "A cooperative hyper-heuristic search framework," *J. Heur.*, vol. 16, no. 6, pp. 835–857, 2009.

[3] J. Gibbs, G. Kendall, and E. Ozcan, "Scheduling english football fixtures over the holiday period using hyper-heuristics," in *Proc. 9th Paral. Prob. Solv. Nat.*, ser. LNCS, vol. 6238. Krakow, Poland, 2010, pp. 496–505.

[4] E. Burke, G. Kendall, and E. Soubeiga, "A tabu-search hyper-heuristic for timetabling and rostering," *J. Heur.*, vol. 9, no. 3, pp. 451–470, 2003.

[5] R. Bai, E. Burke, G. Kendall, J. Li, and B. McCollum, "A hybrid evolutionary approach to the nurse Rostering problem," *IEEE Trans. Evol. Comput.*, vol. 14, no. 4, pp. 580–590, 2010.

[6] R. Qu and E. Burke, "Hybridizations within a graph-based hyper-heuristic framework for university timetabling problems," *J. Oper. Res. Soc.*, vol. 60, no. 9, pp. 1273–1285, 2009.

[7] E. Burke, G. Kendall, M. Misir, and E. Ozcan, "Monte carlo hyper-heuristics for examination timetabling," *Ann. Oper. Res.*, pp. 1–18, 2010, 10.1007/s10479-010-0782-2.

[8] B. Bilgin, P. Demeester, M. Misir, W. Vancroonenburg, and G. Vanden Berghe, "One hyperheuristic approach to two timetabling problems in health care," *J. Heur.*, 2012, 10.1007/s10732-011-9192-0.

[9] P. Garrido and M. Riff, "DVRP: a hard dynamic combinatorial optimisation problem tackled byanevolutionary hyper-heuristic," *J. Heur.*, vol. 16, no. 6, pp. 795–834, 2010.

[10] D. Meignan, A. Koukam, and J.-C. Creput, "Coalition-based metaheuristic: a self-adaptive metaheuristic using reinforcement learning and mimetism," *J. Heur.*, vol. 16, pp. 859–879, 2010.

[11] M. Misir, P. Smet, K. Verbeeck, and G. Vanden Berghe, "Security personnel routing and rostering: a hyper-heuristic approach," in *Proc. 3rd Int. Conf. Appl. Oper. Res.*, ser. LNMS, vol. 3, Istanbul, Turkey, 2011, pp. 193–205.

[12] H. Terashima-Marin, C. Zarate, R. P, and M. Valenzuela-Rendon, "A GA-based method to produce generalized hyper-heuristics for the 2d-regular cutting stock problem," in *Proc. Genetic Evol. Comput. Conf.* New York, USA, 2007, pp. 591–598.

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

[14] P. Cowling, G. Kendall, and L. Han, "An investigation of a hyperheuristic genetic algorithm applied to a trainer scheduling problem," in *Prof. IEEE Congr. Evol. Comput.*, 2002, pp. 1185–1190.

[15] P. Cowling, G. Kendall, and L. Han, "An adaptive length chromosome hyperheuristic genetic algorithm for a trainer scheduling problem," in *Proc. 4th Asia-Pac. Conf. Simul. Evol. Learn.*, Orchid Country Club, Singapore, 2002, pp. 267–271.

[16] L. Han and G. Kendall, "An investigation of a tabu assisted hyper-heuristic genetic algorithm," in *Proc. IEEE Congr. Evol. Comp.*, vol. 3, 2003, pp. 2230–2237.

[17] P. Ross, E. Hart, J. Marin-Blazquez, and S. Schulenberg, "Learning a procedure that can solve hard bin-packing problems: a new ga-based approach to hyperheuristics," in *Prof. Genetic Evol. Comput. Conf.*, ser. LNCS, vol. 2724, Chicago, Illinois, USA, 2003, pp. 1295–1306.

[18] H. Terashima-Marin, A. Moran-Saavedra, and P. Ross, "Forming hyper-heuristics with GAs when solving 2D-regular cutting stock problems," in *Proc. IEEE Congr. Evol. Comput.*, vol. 2, 2005, pp. 1104–1110.

[19] H. Terashima-Marin, R. P, C. Farias-Zarate, E. Lopez-Camacho, and M. Valenzuela-Rendon, "Generalized hyper-heuristics for solving regular and irregular bin packing problems," *Ann. Oper. Res.*, vol. 179, no. 1, pp. 369–392, 2010.

[20] E. López-Camacho, H. Terashima-Marín, and P. Ross, "Defining a problem-state representation with data mining within a hyper-heuristic model which solves 2d irregular bin packing problems," *Proc. 12th Ibero-Am. Conf. Adv. Artif. Intell.*, pp. 204–213, 2010.

[21] E. López-Camacho, H. Terashima-Marín, and P. Ross, "A hyper-heuristic for solving one and two-dimensional bin packing problems," in *Proc. Genetic Evol. Comput. Conf.*, New York, NY, USA, 2011, pp. 257–258.

[22] E. Burke, G. Kendall, D. L. Silva, R. O'Brien, and E. Soubeiga, "An ant algorithm hyperheuristic for the project presentation scheduling problem," in *Proc. IEEE Congr. Evol. Comput.*, vol. 3, 2005, pp. 2263–2270.

[23] P.-C. Chen, G. Kendall, and G. Vanden Berghe, "An ant based hyper-heuristic for the travelling tournament problem," in *Proc. IEEE Symp. Comput. Intell. Sched.*, Hawaii, USA, 2007, pp. 19–26.

[24] A. Keleş, A. Yayimli, and A. Uyar, "Ant based hyper heuristic for physical impairment aware routing and wavelength assignment," in *Proc. IEEE Conf. Sarn.*, 2010, pp. 90–94.

[25] Z. Ren, H. Jiang, J. Xuan, and Z. Luo, "Ant based hyper heuristics with space reduction: a case study of the p-median problem," in *Proc. 11th Parallel Prob. Solv. Nature*, ser. LNCS, vol. 6238, 2011, pp. 546–555.

[26] J. Obit, "Developing novel meta-heuristic, hyper-heuristic and cooperative search for course timetabling problems," Ph.D. dissert., U. Nottingham, 2010.

[27] A. Ahmed, A. Mazhar, A. Sajid, and A. Bukhari, "Particle swarm based hyper-heuristic for tackling real world examinations scheduling problem," *Aust. J. Basic Appl. Sci.*, vol. 5, no. 10, pp. 1406–1413, 2011.

[28] J. Cano-Belman, R. Rios-Mercado, and J. Bautista, "A scatter search based hyper-heuristic for sequencing amixed-model assembly line," *J. Heur.*, vol. 16, no. 6, pp. 749–770, 2010.

[29] E. Burke, M. Hyde, and G. Kendall, "Evolving bin packing heuristics with genetic programming," in *Proc. 9th Parallel Prob. Solv. Nature*, ser. LNCS, vol. 4193, Reykjavik, Iceland, 2006, pp. 860–869.

[30] E. Burke, M. Hyde, G. Kendall, and J. Woodward, "Automatic heuristic generation with genetic programming: Evolving a jack-of-all-trades or a master of one," in *Proc. Genetic Evol. Comput. Conf.*, London, England, 2007, pp. 1559–1565.

[31] S. Allen, E. Burke, M. Hyde, and G. Kendall, "Evolving reusable 3D packing heuristics with genetic programming," in *Proc. Genetic Evol. Comput. Conf.*, 2009, pp. 931–938.

[32] L. Abednego and D. Hendratmo, "Genetic programming hyper-heuristic for solving dynamic production scheduling problem," in *Proc. Int. Conf. Elect. Eng. Inf.*, Bandung, Indonesia, 2011.

[33] M. Bader-El-Den and R. Poli, "Generating SAT local-search heuristics using a GP hyper-heuristic framework," in *Proc. 8th Int. Conf. Evol. Artif.*, ser. LNCS, vol. 4926, 2008, pp. 37–49.

[34] M. Bader-El-Den and R. Poli, "Inc*: an incremental approach for improving local search heuristics," in *Proc. 8th Eur. Conf. Evol. Comp. Comb. Opt.*, 2008, pp. 194–205.

[35] M. Bader-El-Den and R. Poli, "Evolving effective incremental solvers for SAT with a hyper-heuristic framework based on genetic programming," *Genetic Prog. Theor. Prac. VI*, pp. 163–178, 2009.

[36] M. Bader-El-Den and R. Poli, "Grammar-based genetic programming for timetabling," in *Proc. IEEE Congr. Evol. Comput.*, 2009, pp. 2532–2539.

[37] M. Bader-El-Den, R. Poli, and S. Fatima, "Evolving timetabling heuristics using a grammar-based genetic programming hyper-heuristic framework," *Mem. Comput.*, vol. 1, no. 3, pp. 205–219, 2009.

[38] S. Nguyen, M. Zhang, and M. Johnston, "A genetic programming based hyper-heuristic approach for combinatorial optimisation," in *Proc. Genetic Evol. Comput. Conf.*, New York, USA, 2011, pp. 1299–1306.

[39] P. Cowling, G. Kendall, and E. Soubeiga, "A hyperheuristic approach to scheduling a sales summit," in *Proc. 3rd Int. Conf. Pract. Theor. Auto. Timet.*, London, UK, 2001, pp. 176–190.

[40] A. Nareyek, "Choosing search heuristics by non-stationary reinforcement learning," in *Meta.: Comp. Dec.-Mak.*, 2003, pp. 523–544.

[41] E. Ozcan, M. Misir, G. Ochoa, and E. Burke, "A reinforcement learning - great-deluge hyper-heuristic for examination timetabling," *Int. J. Appl. Meta. Comput.*, vol. 1, no. 1, pp. 39–59, 2010.

[42] M. Misir, T. Wauters, K. Verbeeck, and G. Vanden Berghe, "A hyper-heuristic with learning automata for the traveling tournament problem," in *Meta.: Intell. Dec. Mak., 8th Meta. Int. Conf.*, 2011.

[43] M. Misir, K. Verbeeck, P. De Causmaecker, and G. Vanden Berghe, "A New Hyper-heuristic Implementation in HyFlex: a Study on Generality," in *Proc. 5th Multi. Int. Sched. Theor. Appl.*, Phoenix/Arizona, USA, 2011, pp. 374–393.

[44] M. Misir, K. Verbeeck, P. De Causmaecker, and G. Vanden Berghe, "Hyper-heuristics with a dynamic heuristic set for the home care scheduling problem," in *Proc. IEEE Congr. Evol. Comput.*, Barcelona, Spain, 2010, pp. 2875–2882.

[45] E. Burke, S. Petrovic, and R. Qu, "Case based heuristic selection for timetabling problems," *J. Sched.*, vol. 9, no. 2, pp. 115–132, 2006.

[46] H. Asmuni, E. Burke, J. Garibaldi, and B. McCollum, "Fuzzy multiple heuristic orderings for examination timetabling," in *Proc. 5th Int. Conf. Prac. Theor. Auto. Timet.*, ser. LNCS, vol. 3616, 2005, pp. 334–353.

[47] J. Li, E. Burke, and R. Qu, "Integrating neural networks and logistic regression to underpin hyper-heuristic search," *Knowl.-Based Syst.*, vol. 24, no. 2, pp. 322–330, 2010.

[48] E. Burke, B. McCollum, A. Meisels, S. Petrovic, and R. Qu, "A graph-based hyper heuristic for educational timetabling problems," *Eur. J. Oper. Res.*, vol. 176, pp. 177–192, 2007.

[49] J. Ortiz-Bayliss, H. Terashima-Marin, and S. Conant-Pablos, "Neural networks to guide the selection of heuristics within constraint satisfaction problems," in *Proc. 4rd Mex. Conf. Pattern Recogn.*, ser. LNCS, vol. 6718, 2011, pp. 250–259.

[50] E. 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.*, 2012.

[51] A. Cuesta-Cañada, L. Garrido, and H. Terashima-Marín, "Building hyper-heuristics through ant colony optimization for the 2D bin packing problem," in *Proc. 9th Int. Conf. Knowl-Based Intell. Inf. Eng. Syst.*, ser. LNCS, vol. 3684, 2005, pp. 654–660.

[52] E. Burke, N. Pham, and R. Qu, "An investigation of population-based hyper-heuristics for graph colouring," in *Proc. 7th Meta. Int. Conf.*, 2007.

[53] Z. Aziz and G. Kendall, "An investigation of an ant-based hyperheuristic for the capacitated vehicle routing problem," in *Proc. 4th Multi. Int. Sched. Conf. Theor. Appl.*, Dublin, Ireland, 2009, pp. 823–826.

[54] E. Sin, "Reinforcement learning with egd based hyper heuristic system for exam timetabling problem," in *Proc. IEEE Int. Conf. Clou. Comput. Intell. Syst.*, 2011, pp. 462–466.

[55] S. Schulenburg, P. Ross, J. Marn-Blzquez, and E. Hart, "A hyper-heuristic approach to single and multiple step environments in binpacking problems," in *Proc. 5th Int. Works. Learn. Class. Syst.*, ser. LNAI, 2003, pp. 193–218.

[56] H. Terashima-Marin, E. Flores-Alvarez, and P. Ross, "Hyper-heuristics and classifier systems for solving 2D-regular cutting stock problems," in *Proc. Genetic Evol. Comput. Conf.*, 2005, pp. 637–643.

[57] J. G. Marín-Blázquez and S. Schulenburg, "Multi-step environment learning classifier systems applied to hyper-heuristics," in *Proc. Genetic Evol. Comput. Conf.*, 2006, pp. 1521–1528.

[58] J. Marin-Blazquez and S. Schulenburg, "A hyper-heuristic framework with XCS: Learning to create novel problem-solving algorithms constructed from simpler algorithmic ingredients," in *Proc. Int. Conf. Lear. Classif. Syst.*, ser. LNCS, vol. 4399, 2007, pp. 193–218.

[59] M. Hall, "Correlation-based feature selection for machine learning," Ph.D. dissert., U. Waikato, 1999.

[60] M. Hyde and G. Ochoa, "Hyflex competition instance summary," U. Nottingham, Tech. Rep., 2011.

[61] M. Misir, K. Verbeeck, P. De Causmaecker, and G. Vanden Berghe, "A new hyper-heuristic as a general problem solver: an implementation in HyFlex," KAHO Sint-Lieven, Tech. Rep., 2011.