J Sched (2013) 16:291–311
DOI 10.1007/s10951-012-0295-8

A new hyper-heuristic as a general problem solver: an implementation in HyFlex

Mustafa Mısır · Katja Verbeeck · Patrick De Causmaecker · Greet Vanden Berghe

Published online: 14 November 2012

Abstract This study provides a new hyper-heuristic design using a learning-based heuristic selection mechanism together with an adaptive move acceptance criterion. The selection process was supported by an online heuristic subset selection strategy. In addition, a pairwise heuristic hybridization method was designed. The motivation behind building an intelligent selection hyper-heuristic using these adaptive hyper-heuristic sub-mechanisms is to facilitate generality. Therefore, the designed hyper-heuristic was tested on a number of problem domains defined in a high-level framework, i.e., HyFlex. The framework provides a set of problems with a number of instances as well as a group of low-level heuristics. Thus, it can be considered a good environment to measure the generality level of selection hyper-heuristics. The computational results demonstrated the generic performance of the proposed strategy in comparison with other tested hyper-heuristics composed of the sub-mechanisms from the literature. Moreover, the performance and behavior analysis conducted for the hyper-heuristic clearly showed its adaptive characteristics under different search conditions. The principles comprising the here presented algorithm were at the heart of the algorithm that won the first international cross-domain heuristic search competition.

M. Mısır (✉) · K. Verbeeck · G. Vanden Berghe
KAHO Sint-Lieven, CODeS, Gebroeders De Smetstraat 1, 9000 Gent, Belgium
e-mail: mustafa.misir@kahosl.be

K. Verbeeck
e-mail: katja.verbeeck@kahosl.be

G. Vanden Berghe
e-mail: greet.vandenberghe@kahosl.be

M. Mısır · K. Verbeeck · G. Vanden Berghe · P. De Causmaecker
KU Leuven Campus Kortrijk, CODeS, Etienne Sabbelaan 53, 8500 Kortrijk , Belgium
e-mail: patrick.decausmaecker@kuleuven-kortrijk.be

Keywords Hyper-heuristics · general problem solver · HyFlex

1 Introduction

New problem domains or variants of existing problems have progressively entered the literature. They draw the attention of researchers to develop effective solution strategies. The performance of the developed algorithms tends to vary over different problems or even over problem instances belonging to a particular problem. In order to alleviate this situation, algorithm selection strategies have been studied. The primary goal of these strategies is to determine the best algorithm for the target problem instances. Even if this is a useful approach, it misses the possible improvement opportunities due to varying algorithm performance in the course of the solution process. *Selection hyper-heuristics* follow a deeper selection approach by managing a number of given low-level search strategies during the search thereby relying on their strengths and weaknesses (Burke et al. 2003a).

In the literature, the number of studies concerning hyper-heuristics is exponentially growing. Burke et al. (2010a) classified these hyper-heuristics based on the type of the provided feedback mechanisms and the nature of the heuristic search space. Three categories of feedback mechanisms are considered: hyper-heuristics with *online learning*, *offline learning*, and *no learning*. In online learning, the learning process occurs during the search process. Choice function (Cowling et al. 2001), reinforcement learning (Nareyek 2003; Özcan et al. 2010), learning automata (Mısır et al. 2009) are some examples of online learning. In contrast, offline learning refers to learning before starting the search. In particular, case-based reasoning (Burke et al. 2006b) and learning classifier systems (Marín-Blázquez and Schulenburg 2007;

Ross et al. 2002) work in an offline manner. Some hyper-heuristic components without any learning device are also available. An example is the simple random heuristic selection mechanism (Cowling et al. 2001). Hyper-heuristics are categorised with respect to the nature of the given heuristics as *selection hyper-heuristics* and *generation hyper-heuristics*. The aforementioned studies interested in the selection hyper-heuristics executing over constructive or perturbative heuristics. This type of hyper-heuristics also contain various meta-heuristic components such as tabu search (Burke et al. 2003b), genetic algorithms (Han and Kendall 2003), simulated annealing (Dowsland et al. 2007; Burke et al. 2012), and ant colony optimisation (Burke et al. 2005). On the other hand, there exist some hyper-heuristics that aim to generate the low-level heuristics. Burke et al. (2006a), Burke et al. (2007), Fukunaga (2008), Bader-El-Den et al. (2009), and Burke et al. (2010b) utilised genetic programming to generate low-level heuristics designed for the problem instances.

According to the initial definition, hyper-heuristics have been designed to raise the level of generality (Burke et al. 2003a). Placing a domain barrier preventing any problem-dependent data transition from or to hyper-heuristics is the foremost rule for reaching generality. This basic principle leads hyper-heuristics to focus on managing low-level search strategies instead of directly solving a problem instance. Therefore, it is required to design a hyper-heuristic that has the ability to govern different heuristic sets while profiting maximally from their capabilities. HyFlex is a software framework that empowers hyper-heuristic developers to test their approaches across a range of problems proposed by Ochoa et al. (2012). In its current version, six problem domains are available, i.e., max SAT, bin packing, permutation flowshop scheduling, personnel scheduling, travelling salesman, and vehicle routing. Related to each problem domain, a number of perturbative heuristics have been implemented from four main heuristic types, namely mutational heuristics, crossover operators, ruin-recreate heuristics, and hill climbers. The detailed description of these problems, the characteristics and origins of instances as well as the definition of the heuristics were given in Curtois et al. (2010), Hyde et al. (2010b), Hyde et al. (2010a), and Vázquez-Rodríguez et al. (2010). In addition to these features, HyFlex provides an opportunity to change the effect of mutational heuristics and hill climbers. Moreover, it is possible to keep some of the solutions in memory for further use.

In this study, a selection hyper-heuristic is implemented on HyFlex and an experimental performance analysis on the available problems is conducted. The proposed hyper-heuristic has been developed as a general adaptive strategy for the six problem domains and the given heuristic sets. It consists of a dynamic heuristic selection mechanism and a move acceptance strategy evolving with the changing characteristics of the search environment. Experimental results confirm that the developed hyper-heuristic can provide significant performance improvement compared to other hyper-heuristics tested on the problem instances. It should be mentioned here that the proposed hyper-heuristic is the forerunner of the hyper-heuristic that won the first international cross-domain heuristic search challenge (CHeSC 2011).[1]

In the remainder of this paper, the generality requirements for hyper-heuristics are succinctly discussed. The underlying components of the proposed hyper-heuristic are presented in Sect. 3. Next, the experimental results are discussed in Sect. 4. In the last section, the paper is summarised and concluded with a discussion on future research.

2 Generality requirements

A generic selection hyper-heuristic should be capable of managing a diverse range of heuristic sets utilised for solving distinct problems. Although the capability of solving as many problems as possible is the primarily mentioned focus, the main concern should be the management of different heuristic sets. This objective implicitly embraces the aim of solving various problems anyhow. The characteristics of the existing low-level heuristics may require distinct management strategies because each heuristic may have various advantages and disadvantages. These features should be interpreted relying on the dynamic performance of the heuristics and experimental limits such as the given execution time. And the heuristic set as a whole should be used in synergy. For this purpose, a heuristic selection mechanism should consist of particular analysis components to facilitate the adaptivity of the selection process, Table 1.

2.1 Heuristic set features

An analysis tool or a learning component should be designed based on a set of characteristics that determine the behavior of the heuristics. The first characteristic is the heuristic *set size*. A heuristic set with many heuristics has a higher probability of finding satisfactory solutions regarding a problem instance. Conversely, such a set can be hard to manage due to the availability of many options to select. The quality of heuristics depends critically on the set size. If the heuristics have similar performance, then the set size can make no difference. Since this is uncommon, it is required to employ effective learning strategies. The second feature is the *speed* of the heuristics. This characteristic may affect the number of decision steps for the selection process. Hence, it is required to interpret this element in combination with the *improvement capabilities* of the heuristics. The heuristic *specialisation* is

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

Table 1 The distribution of heuristic types for the given problem domains (heuristic indexes are shown in *parentheses*)

Heuristic Type	Bin Packing	Max SAT	Perm. Flowshop	Pers. Scheduling	TSP	VRP
Mutation	3 (0, 3, 5)	6 (0, 1, 2, 3, 4, 5)	5 (0, 1, 2, 3, 4)	1 (11)	5 (0, 1, 2, 3, 4)	3 (0, 1, 7)
Crossover	1 (7)	2 (9, 10)	4 (11, 12, 13, 14)	3 (8, 9, 10)	4 (9, 10, 11, 12)	2 (5, 6)
Ruin-recreate	2 (1, 2)	1 (6)	2 (5, 6)	3 (5, 6, 7)	1 (5)	2 (2, 3)
HillClimber	2 (4, 6)	2 (7, 8)	4 (7, 8, 9, 10)	5 (0, 1, 2, 3, 4)	3 (6, 7, 8)	3 (4, 8, 9)
Total	8	11	15	12	13	10

another factor. A heuristic dedicated to solving a constraint or improving an objective can be considered in this category. In addition, heuristics generating only improving or equal quality solutions such as hill climbers or any other strict heuristic behavior fall in this category. This feature list can obviously be extended. The main purpose should be the usage of the most relevant and effective features (Alpaydin 2010, ch.6) in a collaborative way to predict the future performance of the heuristics.

2.2 Parameter and rule settings

Parameter-free strategies are interesting from a generality perspective. Even though such methodologies are called parameter-free, their behavior depends on some predetermined values or rules. For instance, simple random is a parameter-free heuristic selection mechanism that gives an equal selection chance to each heuristic. The simple random is parameter-free since its parameters are set from the beginning. The improving or equal move acceptance mechanism is also considered a parameter-free approach.

However, this method is based on a predetermined rule in which only better or equal quality solutions are accepted. This means that, the move acceptance method is parameter free, yet not rule-free. All similar methods show that providing a totally independent mechanism seems impossible. Instead, the proposed algorithms should be able to control their parameters or rules according to the search space and the environmental settings with the aim of decreasing the user effect on the algorithm's performance.

3 Hyper-heuristic

A traditional selection hyper-heuristic requires a selection mechanism to determine the best heuristic to apply at each decision step. In addition, it needs a move acceptance strategy to check whether the constructed/visited solution is accepted with regard to its quality and the current state of the search. These consecutive operations are performed until the stopping condition is met. In this study, a new heuristic selection mechanism and a move acceptance strategy including additional components are proposed (Fig. 1). In the following paragraphs, these sub-mechanisms are explained in detail.

3.1 Heuristic selection

3.1.1 Adaptive dynamic heuristic set

Mısır et al. (2010) and Bilgin et al. (2012) studied a dynamic heuristic set strategy for the heuristic selection process. The motivation behind this approach is determining elite heuristic subsets during specific iteration intervals. Similar approaches aiming to eliminate heuristics were studied in Burke et al. (2003b), Chakhlevitch and Cowling (2005), Kendall and Hussin (2005a), and Kendall and Hussin (2005b). The dynamic heuristic set strategy was carried out by eliminating heuristics that are expected to perform worse compared to the rest and keep the best ones with respect to some performance criteria.

$$\begin{aligned} p_i = w_1\Big[\Big(C_{p,\text{best}}(i)+1\Big)^2\Big(t_{\text{remain}}/t_{p,\text{spent}}(i)\Big)\Big] \\ \times b + w_2\Big(f_{p,\text{imp}}(i)/t_{p,\text{spent}}(i)\Big) \\ -w_3\Big(f_{p,wrs}(i)/t_{p,\text{spent}}(i)\Big) + w_4\Big(f_{\text{imp}}(i)/t_{\text{spent}}(i)\Big) \\ -w_5\Big(f_{wrs}(i)/t_{\text{spent}}(i)\Big) \qquad (1) \end{aligned}$$

$$b = \begin{cases} 1, & \sum_{i=0}^{n} C_{p,\text{best}}(i) > 0 \\ 0, & otw. \end{cases}$$

These criteria reflect the performance changes of the heuristics during the search. Performance changes are determined based on the number of new best solutions found, the total fitness improvement and disimprovement per unit execution time. These elements are used according to their importance for the search process. For instance, finding a new best solution is more crucial than improvement without a new best solution. That is, a heuristic that reaches a new best solution is considered a higher quality heuristic. This information was gathered during a phase composed of a predetermined number of iterations. In the new selection strategy, i.e., adaptive dynamic heuristic set (ADHS), an updated version of this performance metric is proposed.

A weighted sum of different performance elements was used to determine the quality of different search strategies.

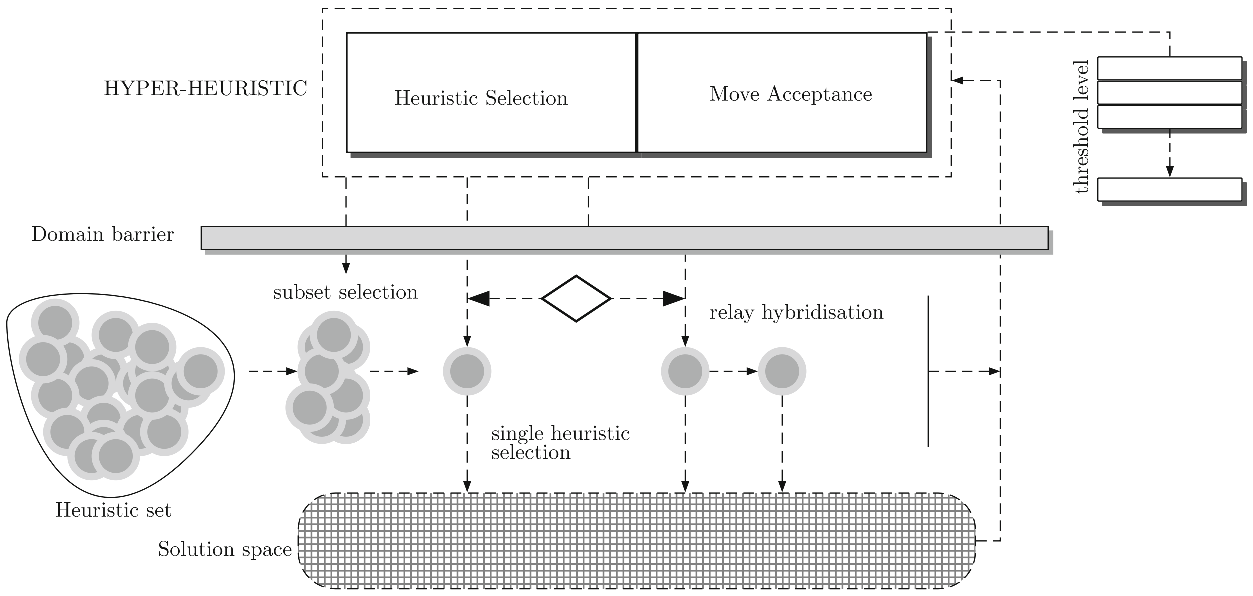

Fig. 1 A selection hyper-heuristic framework

Table 2 The performance metric variables for the adaptive dynamic heuristic set strategy

Variable	Definition
$C_{p,\text{best}}(i)$	The number of new best solutions found by heuristic i during a phase
$f_{\text{imp}}(i)$	The total fitness improvement by heuristic i
$f_{wrs}(i)$	The total fitness worsening by heuristic i
$f_{p,\text{imp}}(i)$	The total fitness improvement by heuristic i during a phase
$f_{p,wrs}(i)$	The total fitness worsening by heuristic i during a phase
$t_{\text{spent}}(i)$	The time spent by heuristic i until now
$t_{p,\text{spent}}(i)$	The time spent by heuristic i during a phase
t_{remain}	The remaining time to finish the whole search process
w_j	The weight of jth performance element ($j \in \{1 \rightarrow 5\}$)

Equation (1) presents the details of the performance measurement for each heuristic. The definition of the variables is given in Table 2. 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. Thus, for instance, if the first performance element of a heuristic has the highest value, then the rest of the performance elements have no effect on the overall p_i of the corresponding heuristic.

In the performance criterion, the first performance element is related to the heuristic's capability of finding a new best solution. In the aforementioned studies, this element was just a counter of new best solutions found by a heuristic. However, it may cause particular difficulties especially if certain heuristics can find new best solutions only after a long time. For that reason, it can be useful to apply a heuristic that may find more new best solutions during the given execution time. Otherwise, a relatively slow heuristic can stay in the heuristic set regardless of its speed because of finding new best solutions. In addition, it is important to use $C_{p,\text{best}}(i) + 1$ rather than $C_{p,\text{best}}(i)$ for preventing the same type of slow heuristics. The second element is used to select improving heuristics and the third element is employed to choose heuristics that deteriorate solutions less. The last two elements have a similar aim, but their values are independent from phases. They are calculated using the values collected until that moment.

The number of phases where a heuristic stays out of an elite heuristic set is denoted by the *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 $d = \sqrt{2n}$ where n shows the number of heuristics and the phase length (pl) is defined as the product of the tabu duration and a constant value ($ph_{\text{factor}} = 500$). These values are recalculated at the end of each phase. Since the calculated p_i values are a combination of different quantities and noisy, the performance values are converted into a quality index ($QI \in [1, n]$) value. A heuristic with the lowest p_i gets 1, the others get one unit more based on their order. The average (avg) of these QI values is calculated to determine the heuristics that will be excluded. Tabu heuristics also attend this calculation with $QI = 1$.

$$\text{avg} = \left\lfloor \left(\sum_{i}^{n} QI_i \right) \Big/ n \right\rfloor \tag{2}$$

Tabu duration adaptation The tabu duration of a specific heuristic is increased if it is successively excluded. In other words, the tabu duration is specifically determined for each heuristic using d as the base value. This is required since the proposed exclusion procedure gets a tabu heuristic back whenever its tabu status expires.

Phase length adaptation The number of iterations per phase is determined by $pl = d * ph_{\text{factor}}$. However, this value is updated based on the speed of the non-tabu heuristics for fairness purposes. For that purpose, at the end of each phase, the time spent per move concerning each non-tabu heuristic is calculated and the total of all (t_{subset}) is used to determine the next phase length. In the following equations, the simple formula for calculating the new phase length is given. A predefined constant value is assigned to $ph_{\text{requested}}$ that denotes the number of phases requested during the whole search process. For this study, $ph_{\text{requested}} = 100$. The duration of a phase (ph_{duration}) is calculated by dividing the total execution time by $ph_{\text{requested}}$. The resulting value is divided by t_{subset} to reach the new pl'. If the calculated value is smaller than pl, then it is utilised as the new phase length. This process is repeated at the end of each phase.

$$\begin{aligned} ph_{\text{duration}} &= t_{\text{total}}/ph_{\text{requested}} \\ pl' &= ph_{\text{duration}}/t_{\text{subset}} \end{aligned} \tag{3}$$

The utilised constant values as well as the provided rules for adaptation purposes regarding the ADHS are set based on the idea of giving chance to all the existing heuristics in the heuristic set to show their performance.

3.1.2 Learning automata

A learning automaton is a finite-state machine that aims at learning the optimal action out of a set of actions ($A = \{a_1, \ldots, a_n\}$) through interaction with an environment in a trial and error manner (Thathachar and Sastry 2004). During a learning process, the environmental response ($\beta(t) \in [0, 1]$) to the selected action is used to update a probability vector p consisting of the selection probabilities of the actions. The update operation is carried out using an update scheme (U). The update scheme of the learning automata is based on Eqs. (4) and (5). Equation (4) refers to the update operation for the applied action and Eq. (5) is used to update the probabilities of the remaining actions.

$$\begin{aligned} p_i(t+1) = {} & p_i(t) + \lambda_1\, \beta(t)(1 - p_i(t)) \\ & -\lambda_2(1-\beta(t))p_i(t) \\ & \text{if } a_i \text{ is the action taken at time step } t \end{aligned} \tag{4}$$

$$\begin{aligned} p_j(t+1) = {} & p_j(t) - \lambda_1\, \beta(t)p_j(t) \\ & + \lambda_2(1-\beta(t))[(r-1)^{-1} - p_j(t)] \\ & \text{if } a_j \neq a_i \end{aligned} \tag{5}$$

A learning automaton with a linear update scheme, *linear reward-inaction*, was employed as a heuristic selection mechanism in Mısır et al. (2009). During the learning process, actions were considered low-level heuristics and heuristics that found new best solutions were rewarded. The changing heuristic probabilities were used like roulette wheel selection. In this study, a *linear reward-punishment* scheme is used to update heuristic probabilities with respect to finding a new best solution, improving the current solution, worsening the current solution and finding a solution with equal quality as the current solution. Related learning rates were set in a decreasing manner in the given order as $\{\lambda_1 = 0.1, \lambda_1 = 0.01, \lambda_2 = 0.002, \lambda_2 = 0.001\}$ without extensive tuning. The first two learning rates are for rewarding due to finding new best solutions and improving solutions. The last two learning rates are used to punish heuristics due to finding worse solutions and solutions with the same quality. Differently from the above mentioned application of the learning automaton, it is just used to keep track of performance changes during different phases of the dynamic heuristic set process. In the beginning of each phase, the learning probabilities are reset. In addition, different learning rates are used for different heuristics. The underlying idea behind this approach is to project the speed of heuristics to the probability updates because it would be unfair to use the same reward/punishment for a very slow and a fast heuristic (Bowling and Veloso 2001). For instance, HyFlex contains local search heuristics that only return improving or equal solutions. These are generally expected to be more time consuming compared to other heuristic types. For that reason, the performance differences among heuristics should be related to their speed. The learning rate of the heuristics are determined using Eq. (6). $C_{\max}$ refers to the number of moves spent per unit time by the fastest heuristic. $C_{i,\text{move}}$ denotes the same value for heuristic i. The resulting value (λ_{mult}) is used as a multiplier for the initial learning rate. If this multiplier makes the related learning rate increase more than its predetermined limit, then the learning rate is set to a predetermined value.

$$\lambda_{\text{mult}} = C_{\max}/C_{i,\text{move}} \tag{6}$$

3.1.3 Relay hybridisation

The heuristics are divided into two types: mutational heuristics and hill climbers in Özcan et al. (2008). They were used within four different frameworks. One of the frameworks, F_C, offers selecting a mutational heuristic first and applying a pre-determined hill climber to the visited solution by the selected mutational heuristic. The experimental results showed that F_C is a very effective strategy to use. In this study, we proposed a relay hybridisation approach to determine a pair of heuristics that can be used consecutively, like F_C, to find superior results. For that purpose, a list of the best heuristics, when applied just after a specific heuristic, is determined. This list is updated in a *First-In-Last-Out* manner, by adding a new heuristic to the end of the list and removing the first one to keep the size of the heuristic set fixed if the list is full. After applying a relatively worse performing heuristic based on the provided performance metric, a heuristic is randomly chosen from its list and applied to the solution generated by this heuristic. Since the heuristic list can include different heuristics more than once, a random selection strategy chooses a heuristic among weighted heuristics. For instance, in a list size of 10, a specific heuristic may be present five times while the rest of the list may be occupied by the other heuristics. In such a list, the probability of selecting the five-times occurring heuristic is 50 %. That is, even if the selection mechanism is random, the end product is a weighted selection mechanism like roulette wheel.

Algorithm 1: Relay hybridisation

```
  Input: list size $l_{size} > 0$; $p, p' \in [0, 1]$
  if $p \leq (C_{phase}/pl)$ then
1     select $LLH_{tabu}$ and apply to $S \rightarrow S'$;
2     if $size(l_i) > 0$ and $p' <= 0.5$ then
3         select a $LLH$ from $l_i$ and apply to $S' \rightarrow S''$;
4     else
5         select a $LLH_{nonTabu}$ and apply to $S' \rightarrow S''$;
      end
  end
```

This strategy is applied based on C_{phase}/pl. C_{phase} denotes the number of iterations passed within a phase. A random probabilistic value $p \in [0, 1]$ is generated and the relay hybridisation is applied if $p \leq (C_{\text{phase}}/pl)$. The pseudocode of the method is depicted in Algorithm 1.

Figure 2 shows an example of the relay hybridisation. In this example, applying heuristic pairs involving $LLH_2 \rightsquigarrow LLH_5$ and $LLH_2 \rightsquigarrow LLH_3$ generated new best solutions before. Whenever LLH_2 is called and if the relay hybridisation is decided to be used, then it is determined whether a consecutive heuristic is randomly chosen from the heuristics that are currently available in the heuristic subset or from its pair list. The selected heuristic is consecutively applied to the solution at hand. If the resulting solution is a new best solution, then the selected heuristic is added to the end of the list.

For some problem domains with some heuristic sets, it may not be possible to find effective heuristic pairs. Therefore, a similar tabu strategy used for the heuristic exclusion process is employed. If no new best solution is found by the relay hybridisation after a phase, then this feature is disabled for one phase. Furthermore, if this consecutively happens, then the tabu duration is increased by 1. This value stays the

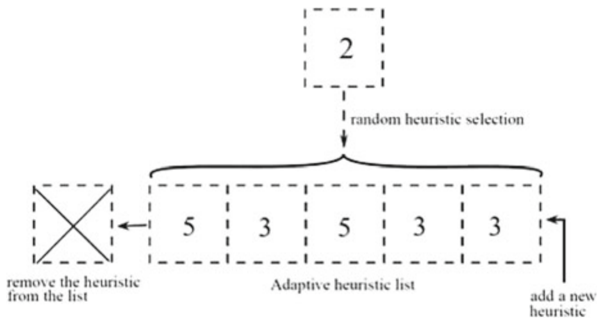
2
random heuristic selection
5
3
5
3
3
remove the heuristic from the list
Adaptive heuristic list
add a new heuristic

Fig. 2 Relay hybridisation

same whenever the tabu duration reaches its upper bound. If it can find a new best solution again, then the tabu duration is set to its initial value, 1.

Heuristic adaptation HyFlex provides an opportunity to modify some of the heuristics in an informed manner. It is possible to increase or decrease the perturbation effect of a mutational heuristic. In addition to that, it is also allowed to change the search depth of local search heuristics. In the proposed approach, the intensity of the mutational heuristics and the search depth of the hill climbers was updated only for the relay hybridisation. If a hill climber can find a new best solution more than fives times, then its search depth is set to 1.0. Otherwise, its value is set to $(0.1 + (ph_{\text{passed}}/pl) \times 0.4)$. ph_{passed} is the number of iterations passed during the current phase. The second part is used also for the mutation operators.

3.2 Move acceptance

Move acceptance mechanisms are very effective on the performance of hyper-heuristics (Özcan et al. 2008). They determine the way to traverse the search space. One of the main concerns of the move acceptance mechanisms is the acceptability of worsening solutions. Accepting a worsening solution is a widely accepted strategy to prevent from getting stuck around a solution. Mısır et al. (2009) proposed a move acceptance strategy, i.e., iteration limited threshold accepting (ILTA), which accepts worsening solutions in a controlled manner. ILTA immediately accepts improving or equal solutions. If the hyper-heuristic cannot find new best solutions during a pre-determined number of iterations, then a worsening solution is accepted. This operation is decided based on the value of the best solution found and a constant value determining a range. An adaptive version of ILTA (AILTA) was proposed by Mısır et al. (2010). In this move acceptance, a second and higher range value is determined. If the hyper-heuristic cannot find a new best solution using a given range value, then this range value is increased to enable accepting much worse solutions. This is required to get rid of local optima.

Algorithm 2: AILLA Move Acceptance

```
   Input: $i = 1$, $K \geq k \geq 0$, $l > 0$
   for $j$=0 to $l$-1 do $best_{list}(j) = f(S_{initial})$
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          $S_b \leftarrow S'$
9          $i = 1$
10         $adapt\_iterations = 0$
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
```

In the proposed hyper-heuristic, the underlying idea behind list-based threshold accepting (Lee et al. 2002) and late acceptance (Burke and Bykov 2008) was employed to decrease the parameter dependency of AILTA. The new move acceptance strategy, i.e., adaptive iteration limited list-based threshold accepting (AILLA), accepts worsening solutions using the fitness values of the previously visited best solutions. The best fitness encountered previously is used as the first threshold value. If it is not good enough to find a new best solution, then a higher fitness from the list ($\text{best}_{\text{list}}$ with l size) is used to decide about the acceptability of worsening solutions. The pseudocode of AILLA is presented in Algorithm 2. Other versions of AILLA were studied in Mısır et al. (2011b,a). Besides, the iteration limit (k) is updated whenever a new best solution is found. Equation (7) shows the details of this update process. $\text{iter}_{\text{elapsed}}$ is the number of iterations elapsed and t_{exec} refers to the total execution time.

$$k = \begin{cases} ((l-1) \times k + \text{iter}_{\text{elapsed}})/l, & cw = 0 \\ ((l-1) \times k + \sum_{i=0}^{cw} k \times 0.5^i \times tf)/l, & \text{otw.} \end{cases}$$
$$tf = (t_{\text{exec}} - t_{\text{elapsed}})/t_{\text{exec}} \tag{7}$$
$$cw = \lfloor \text{iter}_{\text{elapsed}}/k \rfloor$$

4 Experiments

Experiments were carried out on the six problems within HyFlex, namely max SAT (Hyde et al. 2010a), bin packing (Hyde et al. 2010b), permutation flow shop (Vázquez-Rodríguez et al. 2010), personnel scheduling (Curtois et al. 2010), travelling salesman, and vehicle routing problems (Walker et al. 2012). HyFlex provides 12 instances for the first four problems and ten instances for the last two Hyde and Ochoa (2011). Each instance was tested for ten runs. Each run is executed 10 min using Pentium Core 2 Duo 3 GHz PCs with 3.23 GB memory. In addition to the developed hyper-heuristic, nine additional hyper-heuristics were tested. They are combinations of simple random (SR) selection with AILLA, simulated annealing (SA) (Bilgin et al. 2010), late acceptance (LATE) (Demeester et al. 2010), great deluge (GD) (Kendall and Mohamad 2004), and improving or equal (IE) move acceptance mechanisms. Moreover, these acceptance mechanisms together with the adaptive dynamic heuristic set (ADHS) strategy are used as a part of the hyper-heuristic pool.

4.1 Computational results

Table 3 presents the score of the tested hyper-heuristics across six problem domains based on the scoring system used for CHeSC'2011. The detailed experimental results for each

Table 3 Comparison of all the tested hyper-heuristics based on CHeSC'2011 scoring system (higher values are better)

	Bin Packing	Max SAT	Perm. Flowshop	Pers. Scheduling	TSP	VRP	OverAll
ADHS–AILLA	**108**	54	**96**	92	**76**	**82**	**508**
ADHS–LATE	77	73	90	54	68	66	428
ADHS–SA	37	91.5	75.5	**99**	73	75	451
ADHS–GD	40	3	16	11	43	15	128
ADHS–IE	71	22.5	80.5	77	69	42	362
SR–AILLA	56	51.5	47	24	22	35	235.5
SR–LATE	30	59	37	4	15	32	177
SR–SA	2	**100**	9	36	0	22	169
SR–GD	2	0	9	4	22	0	37
SR–IE	45	13.5	8	40	2	21	129.5

Bold values show the highest (best) values for each problem and for the overall performance

Table 4 The results of the hyper-heuristics on the bin packing problem

Inst.	ADHS–AILLA			ADHS–LATE			ADHS–SA		
	Min	Avg	Std	Min	Avg	Std	Min	Avg	Std
1	0.00605	0.00688	0.00049	0.00648	0.00679	0.00022	0.01151	0.01668	0.00354
2	0.00667	0.00714	0.00036	0.00664	0.00709	0.00040	0.01227	0.01614	0.00453
3	0.02044	0.02107	0.00047	0.02078	0.02204	0.00080	0.02119	0.02288	0.00110
4	0.01970	0.02100	0.00113	0.02225	0.02354	0.00063	0.02118	0.02393	0.00144
5	0.00034	0.00119	0.00178	0.00457	0.00494	0.00025	0.00721	0.01135	0.00389
6	0.00306	0.00324	0.00012	0.00364	0.00383	0.00014	0.00887	0.01175	0.00347
7	0.01076	0.01396	0.00219	0.00971	0.01321	0.00273	0.03783	0.07749	0.01890
8	0.01930	0.02243	0.00166	0.02213	0.03240	0.00608	0.04623	0.08209	0.02556
9	0.04775	0.04886	0.00131	0.04643	0.05196	0.00418	0.04593	0.04875	0.00211
10	0.01234	0.01427	0.00137	0.01385	0.01596	0.00154	0.01165	0.01499	0.00185
11	0.10843	0.10851	0.00004	0.10875	0.10890	0.00014	0.11040	0.11386	0.00288
12	0.02102	0.02455	0.00218	0.02881	0.03313	0.00348	0.02680	0.02984	0.00199

Inst.	ADHS–GD			ADHS–IE			SR–AILLA		
	Min	Avg	Std	Min	Avg	Std	Min	Avg	Std
1	0.01191	0.01628	0.00288	0.00657	0.00803	0.00173	0.00697	0.00763	0.00113
2	0.01191	0.01529	0.00301	0.00678	0.00748	0.00036	0.00704	0.00786	0.00040
3	0.02111	0.02327	0.00168	0.02167	0.02280	0.00068	0.02095	0.02266	0.00086
4	0.02192	0.02423	0.00150	0.02157	0.02418	0.00159	0.01981	0.02271	0.00166
5	0.00717	0.01321	0.00315	0.00034	0.00203	0.00218	0.00034	0.00297	0.00226
6	0.00888	0.01213	0.00312	0.00312	0.00335	0.00015	0.00309	0.00326	0.00015
7	0.04795	0.06845	0.01448	0.01095	0.01471	0.00200	0.01108	0.01512	0.00148
8	0.05609	0.07818	0.01673	0.02127	0.02462	0.00246	0.03019	0.03311	0.00141
9	0.04455	0.04869	0.00265	0.04934	0.05121	0.00271	0.08321	0.08769	0.00395
10	0.01274	0.01453	0.00125	0.01560	0.01854	0.00219	0.02653	0.02911	0.00211
11	0.11082	0.11362	0.00274	0.10853	0.10857	0.00004	0.10961	0.10996	0.00049
12	0.02611	0.03010	0.00181	0.02379	0.02876	0.00313	0.04767	0.05566	0.00926

Inst.	SR–LATE			SR–SA			SR–GD		
	Min	Avg	Std	Min	Avg	Std	Min	Avg	Std
1	0.00758	0.01035	0.00144	0.03504	0.03605	0.00047	0.03581	0.03722	0.00162
2	0.00754	0.00841	0.00112	0.03513	0.03582	0.00057	0.03509	0.03633	0.00134
3	0.02232	0.02387	0.00092	0.02424	0.02530	0.00064	0.02557	0.02673	0.00075
4	0.02220	0.02383	0.00120	0.02578	0.02712	0.00090	0.02820	0.03018	0.00103
5	0.00542	0.00593	0.00028	0.01968	0.02023	0.00051	0.01967	0.02027	0.00040
6	0.00403	0.00411	0.00005	0.01749	0.01903	0.00141	y 0.01737	0.01894	0.00145
7	0.01657	0.01867	0.00230	0.09120	0.09990	0.00469	0.09939	0.10265	0.00365
8	0.07247	0.07468	0.00183	0.10301	0.10749	0.00255	0.10513	0.10810	0.00194
9	0.09462	0.09663	0.00160	0.09067	0.09512	0.00327	0.09456	0.09682	0.00180
10	0.02885	0.03024	0.00116	0.02868	0.03017	0.00101	0.02904	0.03051	0.00072
11	0.11243	0.11357	0.00064	0.11996	0.12060	0.00061	0.11993	0.12041	0.00053
12	0.05269	0.05424	0.00197	0.05072	0.05497	0.00223	0.05140	0.05340	0.00147

Table 4 continued

Inst.	SR–IE		
	Min	Avg	Std
1	0.00740	0.01009	0.00174
2	0.00756	0.00874	0.00155
3	0.02170	0.02412	0.00195
4	0.02286	0.02518	0.00152
5	0.00034	0.00385	0.00186
6	0.00325	0.00346	0.00015
7	0.00995	0.01335	0.00264
8	0.03810	0.04285	0.00248
9	0.07949	0.08659	0.00458
10	0.02561	0.02858	0.00186
11	0.10922	0.10934	0.00011
12	0.04581	0.05251	0.00595

Table 5 The results of the hyper-heuristics on the max SAT problem

Inst.	ADHS–AILLA			ADHS–LATE			ADHS–SA		
	Min	Avg	Std	Min	Avg	Std	Min	Avg	Std
1	7	8.8	2.39	3	7.6	3.31	4	6.3	1.25
2	21	24.8	2.15	19	22.9	2.47	19	21.9	2.23
3	15	19.3	2.71	16	18.6	2.01	15	18.5	2.01
4	5	13.7	4.67	8	11.2	2.57	4	9.2	3.82
5	9	25.3	11.82	3	9.6	9.19	7	15.3	10.51
6	14	41.1	14.35	8	20.8	15.89	7	15.4	9.82
7	5	7	1.56	7	8.2	1.23	6	7.3	1.25
8	6	6.7	0.95	5	8.5	1.51	5	6.6	0.84
9	8	10.4	1.78	9	11.7	2.58	6	8.7	1.16
10	213	215.9	1.52	211	213.9	3.60	209	213.6	2.91
11	7	16.6	4.62	6	10.1	3.07	2	7.1	2.77
12	8	9.9	1.79	7	9.9	2.02	8	10	1.94
Inst.	ADHS–GD			ADHS–IE			SR–AILLA		
	Min	Avg	Std	Min	Avg	Std	Min	Avg	Std
1	18	35.4	20.04	7	11.9	3.81	6	11	4.52
2	34	51.3	18.33	23	26.2	2.49	19	25.8	4.98
3	28	48.5	25.87	17	20.9	2.64	18	21.6	2.63
4	14	26.1	16.98	8	14.7	4.00	7	11.6	3.31
5	10	31.4	21.43	9	22.6	11.47	5	16.8	11.38
6	44	72.8	30.75	43	51	4.45	10	27.5	15.71
7	9	12.4	1.78	8	9.5	1.78	6	8.3	1.06
8	9	13.2	5.51	8	9.6	1.17	7	7.7	0.67
9	11	18.1	7.58	11	13.8	2.44	8	9.6	0.97
10	219	249.9	33.84	213	218.9	4.53	211	213.2	1.75
11	17	29.4	11.95	17	22.7	3.86	9	14.4	4.01
12	10	14.9	3.54	10	12.2	1.55	8	10	1.15

Table 5 continued

Inst.	SR–LATE			SR–SA			SR–GD		
	Min	Avg	Std	Min	Avg	Std	Min	Avg	Std
1	6	9.4	3.20	5	8.2	2.15	235	254.4	12.34
2	21	24	2.16	19	22.3	2.45	256	275.2	12.25
3	14	19.4	2.46	17	19.7	2.58	262	280.8	12.46
4	7	10.2	1.99	3	8.4	3.17	146	154.7	5.79
5	4	13.4	11.62	5	11.5	8.29	135	155.5	10.63
6	6	11.2	4.10	14	32.7	11.75	244	255.6	11.93
7	5	8.6	1.58	5	6.6	0.97	22	33.6	6.96
8	6	8.6	1.78	5	6.4	0.84	24	35.6	6.95
9	8	9.9	1.20	7	8.4	0.70	34	53.6	8.37
10	211	215.6	2.46	211	212.8	1.75	295	339.9	21.76
11	9	12.2	2.04	1	5.2	2.49	197	232	22.04
12	9	11.1	1.10	7	9.2	1.03	40	55.4	7.26

Inst.	SR–IE		
	Min	Avg	Std
1	11	16	3.30
2	23	26.6	2.50
3	17	20.3	2.21
4	26	29.9	3.18
5	22	37.9	7.25
6	39	54.2	6.66
7	6	10.4	2.32
8	7	9.6	1.96
9	10	12.2	1.81
10	211	216.4	3.98
11	19	31.3	5.98
12	11	13.1	1.60

problem domain are presented in Tables 4, 5, 6, 7, 8, and 9. Among the tested hyper-heuristics, ADHS–AILLA gets the highest score, 508 (out of 680). ADHS–SA and ADHS–LATE follow it with the scores of 451 and 458, respectively. ADHS–AILLA performs the best for the bin packing, permutation flowshop scheduling, travelling salesman, and vehicle routing problems. It comes second after ADHS–SA for the personnel scheduling domain. Its worst performance comes for the max SAT problem as the fifth hyper-heuristic.

For the hyper-heuristics with SR, AILLA provides the best results based on their overall performance. This indicates that the proposed move acceptance strategy is effective while dealing with different heuristic sets for different problem domains. Furthermore, its adaptive nature increases its robustness.

The performance difference between the hyper-heuristics with ADHS and SR demonstrate the effectiveness and contribution of the selection mechanism. The reason behind this result is the availability of different heuristics with respect to their speed and improvement capabilities. ADHS intelligently determines elite heuristic subsets during the run and explores effective heuristic pairs. The following subsections discuss the hyper-heuristics' performance for each problem domain separately.

4.1.1 Bin packing

For the bin packing problem, experimens showed that ADHS–AILLA performs significantly better than the other tested hyper-heuristics based on the Wilcoxon test with a 95 % confidence interval. Its CHeSC'2011 score is 108 which is the only score exceeding 100 throughout all the test problem domains among the tested approaches. There is clear performance difference between selection mechanisms, ADHS and SR. For all tested move acceptance mechanisms, ADHS outperforms SR. For the hyper-heuristics with SR, AILLA also outperforms the rest.

Table 6 The results of the hyper-heuristics on the permutation flowshop scheduling problem

Inst.	ADHS–AILLA			ADHS–LATE			ADHS–SA		
	Min	Avg	Std	Min	Avg	Std	Min	Avg	Std
1	6245	6282.1	19.17	6252	6278.1	17.46	6261	6288.3	21.79
2	6226	6245.9	12.91	6232	6246.5	8.68	6235	6250.9	15.33
3	6314	6333.1	14.09	6317	6327.2	10.63	6313	6335.2	13.88
4	6323	6341.9	18.82	6303	6336.8	20.12	6325	6349.2	16.59
5	6342	6372.3	17.67	6344	6369	17.38	6363	6386.5	19.68
6	10496	10499.2	3.97	10495	10499.7	4.08	10494	10502.9	7.61
7	10922	10922.8	0.42	10922	10922.6	0.52	10922	10922.9	0.32
8	26260	26305.5	32.07	26376	26436.1	37.57	26232	26310.2	40.04
9	26765	26801.6	22.44	26860	26909.1	26.53	26752	26785.3	28.11
10	26574	26657.3	38.11	26672	26735.6	34.50	26611	26644.7	33.05
11	11338	11371.6	24.14	11346	11366.8	15.53	11342	11379.8	17.67
12	26523	26620.2	38.91	26670	26715.4	31.44	26581	26625.9	28.18

Inst.	ADHS–GD			ADHS–IE			SR–AILLA		
	Min	Avg	Std	Min	Avg	Std	Min	Avg	Std
1	6335	6370.6	18.89	6272	6294.3	21.21	6283	6325.7	30.94
2	6296	6321.9	16.30	6222	6248.7	15.63	6254	6276.6	15.09
3	6382	6399.1	11.72	6307	6331.7	16.03	6324	6351.7	15.39
4	6366	6380.7	14.01	6323	6349.2	17.57	6323	6353.2	18.84
5	6447	6464.9	8.40	6344	6371.8	19.31	6380	6414.5	24.51
6	10520	10537.6	11.36	10490	10502	7.67	10509	10523.2	13.46
7	10956	10967.6	9.82	10922	10924.7	5.74	10923	10940.1	14.67
8	26403	26438.5	22.86	26251	26307.7	26.81	26361	26438	49.16
9	26883	26934.5	36.72	26758	26798.3	28.16	26805	26875.1	44.42
10	26736	26748.9	9.50	26585	26641.8	30.44	26622	26705	41.65
11	11486	11505.8	13.40	11325	11386.8	33.67	11408	11457.2	35.10
12	26679	26724.3	32.01	26585	26646	41.98	26651	26726.6	39.23

Inst.	SR–LATE			SR–SA			SR–GD		
	Min	Avg	Std	Min	Avg	Std	Min	Avg	Std
1	6332	6358.5	24.31	6340	6366.5	20.65	6366	6386.8	9.82
2	6277	6313.6	21.79	6281	6337	28.97	6322	6333.7	8.37
3	6350	6376.6	21.97	6374	6391.2	11.76	6399	6412.8	7.42
4	6323	6361.5	16.67	6326	6362.6	17.70	6388	6396.2	5.05
5	6374	6421.4	23.64	6402	6426.6	19.98	6460	6475.6	10.89
6	10517	10545.6	12.02	10518	10550.8	23.77	10520	10533	10.59
7	10923	10968	28.95	10923	10955	25.80	10965	10983.4	12.55
8	26336	26399.7	32.68	26436	26531.6	51.20	26379	26444.4	38.30
9	26812	26880.4	33.14	26897	27002.1	65.20	26893	26941.1	34.48
10	26694	26727.7	20.09	26750	26799.9	38.95	26687	26754.3	32.92
11	11454	11493.8	20.10	11391	11504.8	54.89	11453	11503.6	31.43
12	26638	26679.4	34.21	26732	26807.5	57.22	26714	26747.7	18.84

Table 6 continued

Inst.	SR–IE		
	Min	Avg	Std
1	6356	6385.2	23.69
2	6289	6331.3	23.84
3	6357	6390	18.87
4	6363	6377.2	12.98
5	6393	6424.3	18.51
6	10512	10545.3	26.96
7	10923	10972.1	40.72
8	26460	26532.8	67.01
9	26857	27015	88.09
10	26759	26805.9	35.16
11	11458	11504.4	28.50
12	26703	26792.9	57.02

Table 7 The results of the hyper-heuristics on the personnel scheduling problem

Inst.	ADHS–AILLA			ADHS–LATE			ADHS–SA		
	Min	Avg	Std	Min	Avg	Std	Min	Avg	Std
1	3294	3311.6	15.66	3315	3342.6	15.98	3296	3323.9	23.23
2	1994	2300.4	170.74	2220	2434.1	136.56	2020	2300.9	171.73
3	305	327	21.76	355	386.5	15.64	290	328.5	20.01
4	14	20.6	3.20	15	22	3.71	15	20	3.68
5	20	25.2	4.98	20	25.3	4.42	16	23.3	4.16
6	16	23	3.30	17	24.3	5.08	13	23.3	5.81
7	1109	1212.2	108.80	1121	1207.9	125.77	1102	1177.1	104.91
8	2163	2247.2	67.82	2302	2708.9	456.80	2161	2206.7	58.53
9	3224	3291.8	74.12	3373	4210.5	761.72	3147	3258.7	75.74
10	9295	9615.6	228.06	10191	26773.5	15568.81	9394	9534.5	128.97
11	1515	1732.6	186.11	1638	1823.7	81.09	1605	1843.6	106.54
12	300	319	20.11	310	334.6	22.86	295	310	11.06
Inst.	ADHS–GD			ADHS–IE			SR–AILLA		
	Min	Avg	Std	Min	Avg	Std	Min	Avg	Std
1	3339	3368.6	21.24	3291	3320.5	18.43	3339	3349.5	12.91
2	2328	2469.9	121.30	2040	2259.3	115.55	2105	2631.7	592.65
3	370	424.5	32.01	305	332	17.67	410	1726.1	1035.17
4	26	30.5	5.21	17	21.2	3.43	22	29.1	3.84
5	25	35.4	6.00	18	24.6	4.30	31	33.4	2.55
6	29	35.9	5.24	19	26.2	5.25	25	31.6	4.81
7	1322	1414.8	155.88	1105	1250.4	133.74	1109	1243	109.91
8	2493	3353.7	621.16	2177	2228.4	68.98	2262	2314.4	61.39
9	3430	4626	958.91	3135	3278.9	88.91	3145	3326.3	128.82
10	10298	32015.5	15145.38	9515	9585.1	72.26	9529	9641.5	94.39
11	1753	1911.3	95.93	1508	1857.4	332.47	1605	1977.7	387.68
12	345	368.8	18.87	305	324	15.95	365	713	559.05

Table 7 continued

Inst.	SR–LATE			SR–SA			SR–GD		
	Min	Avg	Std	Min	Avg	Std	Min	Avg	Std
1	3333	3352.7	10.33	3313	3346.7	22.02	3328	3366.4	20.26
2	2383	2466.6	59.77	2105	2459.7	349.77	2382	2615.3	152.66
3	365	388.5	19.73	455	1969	1306.25	386	435.1	31.71
4	18	26.4	3.60	19	29.3	5.33	28	34.4	4.43
5	21	26.7	3.86	21	30.5	5.28	28	38.3	4.42
6	23	29.7	2.87	28	31.2	2.39	33	39.8	4.64
7	1114	1290.5	165.08	1111	1166.1	67.01	1156	1396.8	146.64
8	2264	2723.2	438.59	2188	2291.6	84.49	2297	2843.8	424.04
9	3311	4210.8	520.24	3229	3330.4	86.26	3644	4302.6	611.83
10	13938	28415.2	13696.88	9426	9651	190.70	14718	29798.1	10935.77
11	1755	1935.9	133.64	1460	2009.8	365.78	1809	1986.8	134.91
12	340	361.5	15.28	365	701.5	595.24	350	381.6	21.11

Inst.	SR–IE		
	Min	Avg	Std
1	3314	3355.3	22.73
2	2265	2513.5	180.08
3	510	1453.5	894.67
4	20	27.1	6.77
5	25	30.4	5.08
6	25	31.1	4.70
7	1107	1247.2	104.70
8	2180	2291.8	75.20
9	3159	3247.9	55.94
10	9508	9668.8	121.74
11	1565	1734.8	161.37
12	380	824.5	625.86

4.1.2 Max SAT

SR–SA is the best performing approach for the max SAT problem. Despite this performance difference, ADHS–SA and ADHS–LATE perform significantly similar. ADHS performs better than SR with all the acceptance mechanisms except SA. Performance decrease caused by ADHS against SR with SA occurs only for this problem domain.

4.1.3 Permutation flowshop scheduling

The experimental results on the permutation flowshop scheduling instances indicate that ADHS–AILLA achieves better quality results in comparison with the other competing algorithms. However, there is no significant performance difference between ADHS–AILLA and ADHS–LATE, ADHS–IE, ADHS–SA. For the SR-version of these hyper-heuristics, SR–AILLA is significantly better than SR–IE and SR–SA despite their similar performance with ADHS. Therefore, it can be concluded that the effect of the selection mechanism on the performance is relatively high for the corresponding problem instances.

4.1.4 Personnel scheduling

ADHS–SA is the winning algorithm for the personnel scheduling domain. Its performance is significantly similar to ADHS–AILLA. The performance difference between ADHS and SR is again extremely high. In particular, SR–IE gets a score of 8 but the score of ADHS–IE is 80.5 which is the improvement provided just by changing the selection mechanism.

4.1.5 Travelling salesman

The travelling salesman problem is another domain where ADHS–AILLA performs best. However, no significant

Table 8 The results of the hyper-heuristics on the traveling salesman problem

Inst.	ADHS–AILLA			ADHS–LATE			ADHS–SA		
	Min	Avg	Std	Min	Avg	Std	Min	Avg	Std
1	48194.92	48340.97	319.54	48194.92	48272.71	92.24	48194.92	48263.53	95.19
2	108460.12	109924.07	1116.71	108815.00	110031.08	891.31	109394.92	110446.14	876.74
3	6832.09	6859.90	20.38	6844.23	6859.95	17.20	6837.72	6858.60	11.76
4	42155.38	42290.58	129.84	42117.42	42256.66	85.39	42158.63	42263.69	84.78
5	8911.98	8930.70	13.01	8921.66	8952.18	18.03	8921.10	8956.67	25.58
6	57747.49	58295.63	284.86	57772.26	58215.06	408.27	57718.46	58369.29	387.16
7	53454.48	54180.01	745.13	53692.78	54691.59	679.92	52571.13	54815.30	1335.36
8	66929.58	67587.25	380.69	67325.44	68115.59	916.13	66914.49	67236.46	242.14
9	20664651.02	20767860.09	69362.38	21044415.03	21195482.40	114493.15	20718695.42	20795493.49	98797.92
10	666525.47	668486.45	1281.89	673002.75	674226.75	949.74	667095.97	667836.72	629.64

Inst.	ADHS–GD			ADHS–IE			SR–AILLA		
	Min	Avg	Std	Min	Avg	Std	Min	Avg	Std
1	48194.92	48265.09	76.95	48194.92	48300.76	182.25	48303.92	49541.13	780.02
2	108887.53	111613.16	2094.32	108677.94	110246.18	1124.33	110331.68	112424.08	1357.37
3	6875.72	6931.98	43.00	6811.67	6854.27	19.56	6944.91	6974.78	26.43
4	42474.91	42734.27	222.98	42159.28	42319.19	103.03	42645.94	43234.94	382.24
5	9001.77	9064.31	44.95	8937.93	8958.83	17.15	9040.76	9092.15	50.16
6	58121.02	58968.29	549.97	58038.01	58359.04	260.22	59575.64	60076.20	443.18
7	53909.28	55717.75	1538.77	53610.45	54404.46	587.05	54675.31	55809.41	918.88
8	68322.99	69208.65	685.92	66716.29	67365.65	394.03	68763.98	69725.06	731.31
9	21115567.59	21258092.41	95417.79	20742658.99	20814378.40	70404.25	21139778.14	21440165.07	197350.82
10	675089.83	676856.22	1533.22	666445.85	667829.70	751.10	675287.99	677529.85	1449.93

Inst.	SR–LATE			SR–SA			SR–GD		
	Min	Avg	Std	Min	Avg	Std	Min	Avg	Std
1	49257.55	49857.53	499.53	49928.94	51227.53	751.27	48564.88	48956.81	237.87
2	110165.53	113993.35	1758.02	112753.01	115835.29	2347.66	110130.63	112009.43	1336.85
3	6959.17	7013.30	29.90	7017.76	7084.88	44.26	6953.66	6974.80	16.87
4	43221.69	43455.04	145.71	43556.02	43963.02	237.62	43007.56	43171.41	124.28
5	9127.84	9155.17	25.85	9223.23	9287.21	50.31	9072.50	9098.46	26.68
6	59755.48	60221.97	239.84	60020.14	60830.82	653.59	59559.65	59713.87	169.33
7	54350.43	55975.61	1146.34	54728.19	56111.13	986.03	55127.82	56394.91	1091.53
8	69186.32	70016.69	626.73	70159.76	70793.80	514.35	68858.43	69789.70	627.82
9	21105275.35	21252337.18	107060.34	21325127.76	21453051.13	84186.01	21191884.08	21417614.23	158583.77
10	675271.16	676961.30	1045.69	677866.45	679295.26	1291.11	675039.28	676994.72	1260.44

Inst.	SR–IE		
	Min	Avg	Std
1	49335.98	51209.43	775.76
2	110689.12	114848.24	2840.05
3	7017.76	7087.41	48.71
4	43564.38	43937.91	272.42
5	9151.04	9232.95	45.73

Table 8 continued

Inst.	SR–IE		
	Min	Avg	Std
6	60216.65	60792.07	369.95
7	54408.85	55888.61	1016.23
8	68843.26	70121.85	855.94
9	21238309.99	21469284.75	159054.93
10	675773.01	678128.75	1517.39

Table 9 The results of the hyper-heuristics on the vehicle routing problem

Inst.	ADHS–AILLA			ADHS–LATE			ADHS–SA		
	Min	Avg	Std	Min	Avg	Std	Min	Avg	Std
1	5065.97	5130.80	35.42	5091.90	5142.00	36.16	5082.05	5130.25	36.90
2	20653.84	21255.81	515.20	20652.47	21058.12	511.64	20654.90	21261.44	517.08
3	13301.96	13562.09	431.32	13301.54	13354.57	26.08	13311.63	13562.85	435.16
4	5317.40	5352.38	30.48	5307.30	5371.14	36.57	5326.31	5367.41	28.61
5	14270.54	14596.84	485.51	13354.67	14214.88	304.14	13296.24	14391.75	567.42
6	142500.38	145420.82	1362.31	150694.17	161808.71	15156.40	145320.93	147395.05	1876.41
7	59040.30	61270.50	1813.02	61632.17	64676.70	2280.38	59268.99	61837.04	1963.05
8	159472.84	160617.37	501.14	159697.12	162177.45	1944.21	159767.41	161169.18	780.70
9	148746.63	152011.50	1836.34	147881.97	155277.03	7625.65	150609.69	152743.51	1577.64
10	146385.27	147212.71	840.82	144494.22	144998.00	384.90	146095.58	147157.47	769.96
Inst.	ADHS–GD			ADHS–IE			SR–AILLA		
	Min	Avg	Std	Min	Avg	Std	Min	Avg	Std
1	4511.27	5135.46	222.88	5096.24	5173.24	67.13	5123.24	5173.78	26.05
2	20679.93	21335.72	486.30	20654.06	21263.62	521.96	20656.16	21454.80	418.95
3	12504.42	13739.45	678.33	13331.98	13779.13	532.92	13330.84	14095.40	520.54
4	5397.69	5562.54	294.99	5328.44	5558.63	391.26	5342.35	5382.40	46.92
5	14305.11	14865.83	733.75	14274.97	14919.90	520.97	14260.22	14489.48	435.96
6	215100.96	241955.39	15084.82	145530.83	147629.64	1991.99	156259.67	159172.87	2686.16
7	70300.67	83592.47	11530.71	59269.98	62569.36	2623.83	66073.03	68260.60	1643.27
8	178413.45	200534.46	20384.98	160247.30	161219.64	783.82	162682.76	165091.97	1513.45
9	176062.16	202494.28	16272.58	151911.91	154670.64	1715.27	161802.83	163823.64	1615.82
10	160990.41	167725.95	4019.81	145170.22	147426.92	1193.93	148558.29	150138.59	832.58
Inst.	SR–LATE			SR–SA			SR–GD		
	Min	Avg	Std	Min	Avg	Std	Min	Avg	Std
1	5171.65	5210.47	21.28	5127.49	5202.80	44.15	5252.10	5351.11	67.29
2	20655.05	20863.81	422.00	20694.92	21573.50	309.10	20668.74	21975.00	682.05
3	13319.56	13564.92	438.38	13346.68	14178.73	429.32	14414.75	14764.23	512.01
4	5358.07	5415.56	33.22	5361.07	5512.25	333.17	5460.27	6002.23	448.96
5	13351.68	14312.51	465.00	14272.65	14706.85	538.57	15329.63	15783.38	523.59
6	192819.81	195770.10	1840.80	153277.28	161223.61	3418.89	317092.35	330347.00	9385.38
7	89495.74	92826.22	1587.41	63707.66	68124.84	2947.26	120249.12	125272.46	2728.87

Table 9 continued

Inst.	SR–LATE			SR–SA			SR–GD		
	Min	Avg	Std	Min	Avg	Std	Min	Avg	Std
8	170984.64	172793.31	913.07	164288.33	165712.33	1262.67	228655.29	236974.89	5993.82
9	188657.42	191291.93	2108.08	160565.73	166457.13	3017.37	286915.80	306871.39	11020.64
10	153557.25	154946.58	1043.47	148897.05	151023.94	1230.66	215957.29	229365.16	5785.46

Inst.	SR–IE		
	Min	Avg	Std
1	5185.88	5249.82	60.46
2	20661.09	21269.81	516.92
3	13391.09	14323.02	328.51
4	5406.36	5727.30	441.82
5	14274.39	14729.14	515.57
6	154430.93	159568.08	3731.54
7	65136.72	68772.11	2526.90
8	161927.48	164211.23	1622.87
9	162386.92	164244.56	1266.54
10	148520.30	150636.49	1132.61

performance difference can be noticed compared with ADHS–SA, ADHS–IE and ADHS–LATE. As mentioned for the personnel scheduling problem, there are large margins between the scores of the hyper-heuristics with ADHS and SR for this problem domain too. For instance, the score of SA increases from 0 to 73 and the score of IE changes from 2 to 69 using ADHS instead of SR. In addition, SA and IE move acceptance mechanisms perform better than LATE and GD with ADHS while they perform significantly worse than LATE and GD with SR.

4.1.6 Vehicle routing

For the vehicle routing problem, ADHS–AILLA comes first with the score of 82. ADHS–SA and ADHS–LATE follow it with no significant performance difference. With ADHS, the scores of tested hyper-heuristics are two or three times higher than their scores compared to theirs scores obtained utilising SR.

4.2 Heuristic selection

Figure 3 demonstrates the effect of the proposed selection mechanism, ADHS, on the tested heuristic sets. For the bin packing problem, there is a large gap between ADHS and SR based on the number of heuristic calls during the run, ADHS chooses effective and fast heuristics more frequently. However, in some part of the search space, it prefers slower heuristics due to the changing search requirements. The selection process is still faster for max SAT but the difference is relatively smaller. For the flowshop scheduling problem, ADHS starts with fast heuristics, but then prefers slower options for performance improvement. The number of heuristic calls by ADHS compared to SR are also smaller for the personnel scheduling and travelling salesman problems The reason behind this slowing down on the number of heuristic calls partially caused by the relay hybridisation. For the vehicle routing problem, the number of heuristic calls by ADHS is again larger.

Figure 4 illustrates the number of calls for each heuristic regarding each problem domain. It can be concluded that at certain time points, the behaviour of the heuristic selection changes. The heuristics available for solving the bin packing problem instances show similar performance or behaviour through the whole search. This can be deduced from the constantly increasing number of heuristic calls. Only one heuristic started to be called more frequently during the second half of the search time. A similar pattern of behaviour can be seen on the max SAT, travelling salesman and vehicle routing problems' heuristics. For the permutation flowshop scheduling problem, the number of calls for a group of heuristics follows different trends during different time intervals. For the personnel scheduling problem, some immediate changes on the number of calls for each heuristic during the whole running period can be seen. This is mainly related to the speed of the heuristics that allows only a limited number of iterations for the given execution time. In addition, it should be noted that the variations of the heuristic selection and exclusion processes depend on the environmental responses from the corresponding search spaces during learning. For instance, the graph for max SAT shows that there is no strict changes of the heuristic preferences in general. The reason

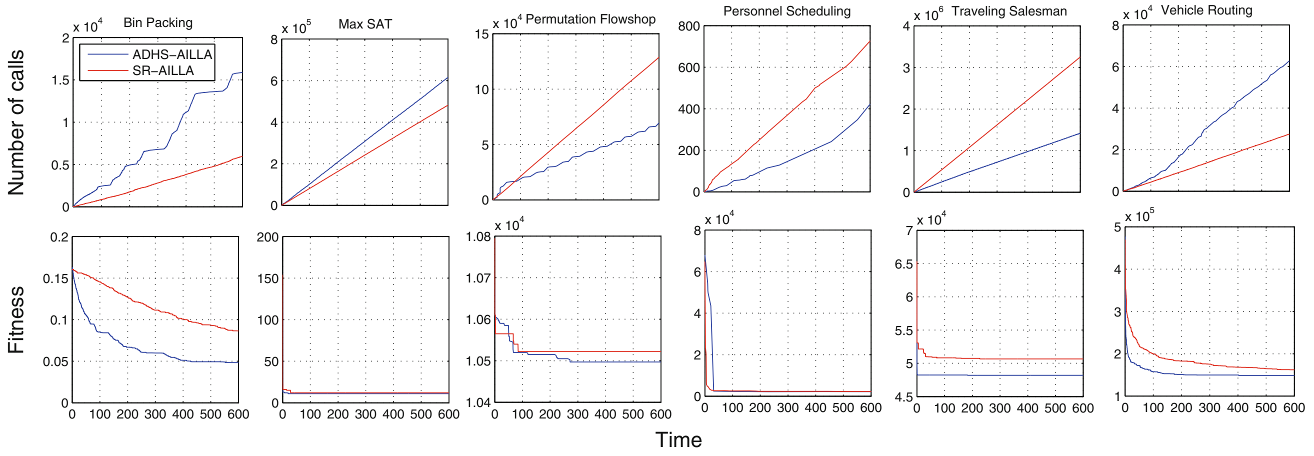

Fig. 3 Comparison of the performance and speed between the tested selection mechanisms (the chart pairs belong to the results obtained from a randomly selected run on one instance of the bin packing, max SAT, flowshop scheduling, and personnel scheduling problems from *top* to *bottom*). The data belongs to a run on an instance for each problem domain

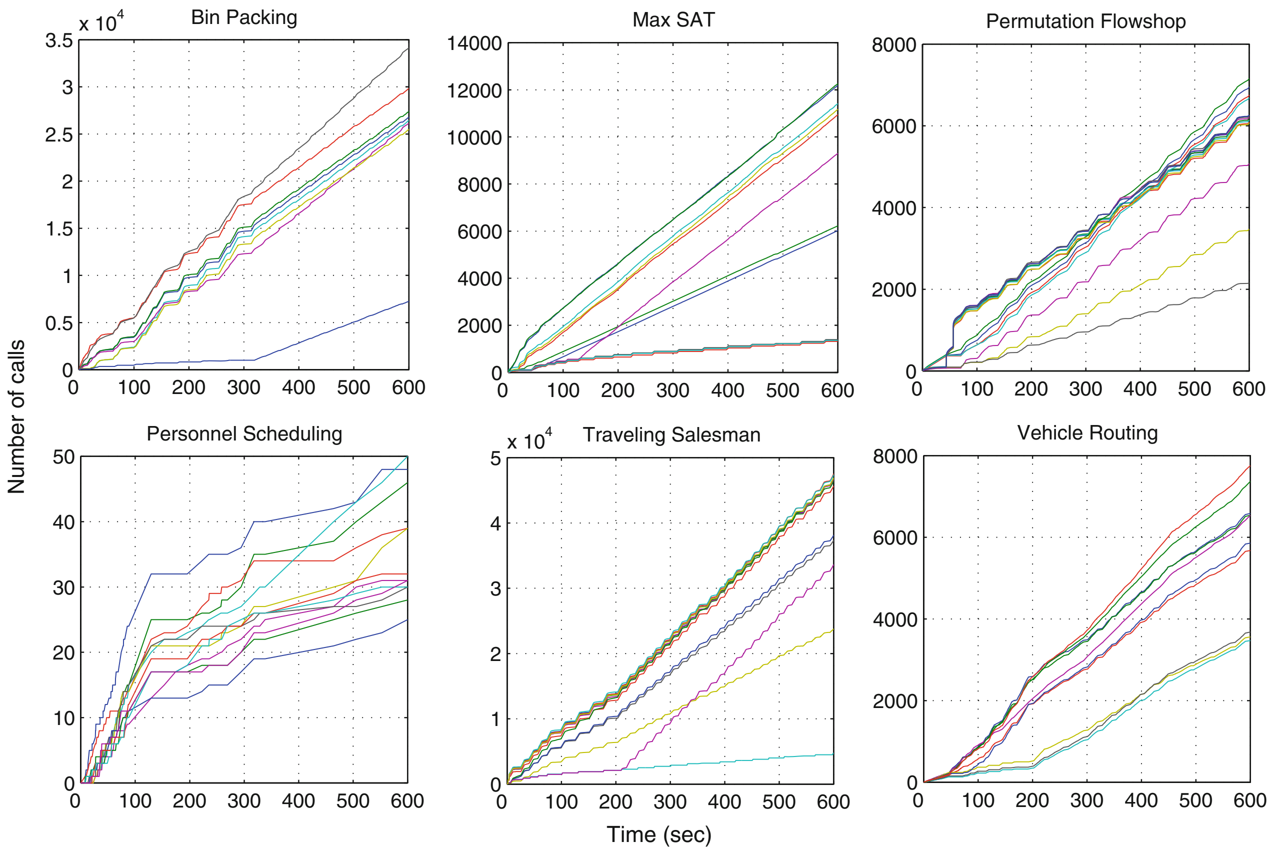

Fig. 4 The number of calls for each heuristic over the given problem domains by ADHS–AILLA (The data belongs to a run on an instance for each problem domain)

behind this behavior is related to the characteristic of the search space providing easy solution improvement opportunities. This means that it is easy to immediately find good quality solutions due to the characteristics of their search spaces shaped by their solution spaces and their corresponding low-level heuristics. After this quick search process, it gets harder to find improving solutions as the learning process regarding the heuristic selection operations slows down.

4.3 Relay hybridisation

Figure 5 demonstrates the effective heuristic pairs yielding new best solutions during the run on the tested problem domains. It shows the first heuristics as squares and the second heuristics as circles. Applying these two types of heuristics consecutively at an iteration is the hybridisation considered. The number of new best solutions found changes for different problem domains as well as different time intervals during a run. For the bin packing problem relay hybridisation is very effective for finding intensifying heuristic pairs for most of the execution time. The only hill climbers (LLH_4, LLH_6) perform as the most effective second heuristics. Two mutational heuristics (LLH_0, LLH_3), one ruin-recreate heuristic (LLH_2) and the crossover operator (LLH_7) help to diversify the solution for (LLH_6) as the first heuristics.

For the max SAT problem, the effect of hybridisation is very limited. The hill climbers (LLH_7, LLH_8) are used as the second heuristics and all the mutational heuristics (LLH_0, LLH_1, LLH_2, LLH_3, LLH_4, LLH_5) available in the corresponding heuristic set were utilised as the first heuristics.

The hill climbers (LLH_7, LLH_8, LLH_9, LLH_{10}) for the permutation flowshop scheduling problems were effectively used as second heuristics. Among them (LLH_7) and (LLH_8) were the most preferred options. The ruin-recreate heuristics (LLH_5, LLH_6) are used as the first heuristics generally together with two mutational heuristics (LLH_0, LLH_1).

For the personnel scheduling, due to the running time limit, the number of iterations were not as high as the other problem domains. Anyway, the relay hybridisation delivered some effective heuristic pairs composed of one mutational heuristic (LLH_{11}) two ruin-recreate heuristics (LLH_6, LLH_7) as the first heuristics and two hill climbers (LLH_3, LLH_4) as the second heuristics.

For the travelling salesman problem the effect of hybridisation is valid within certain time intervals. Two hill climbers

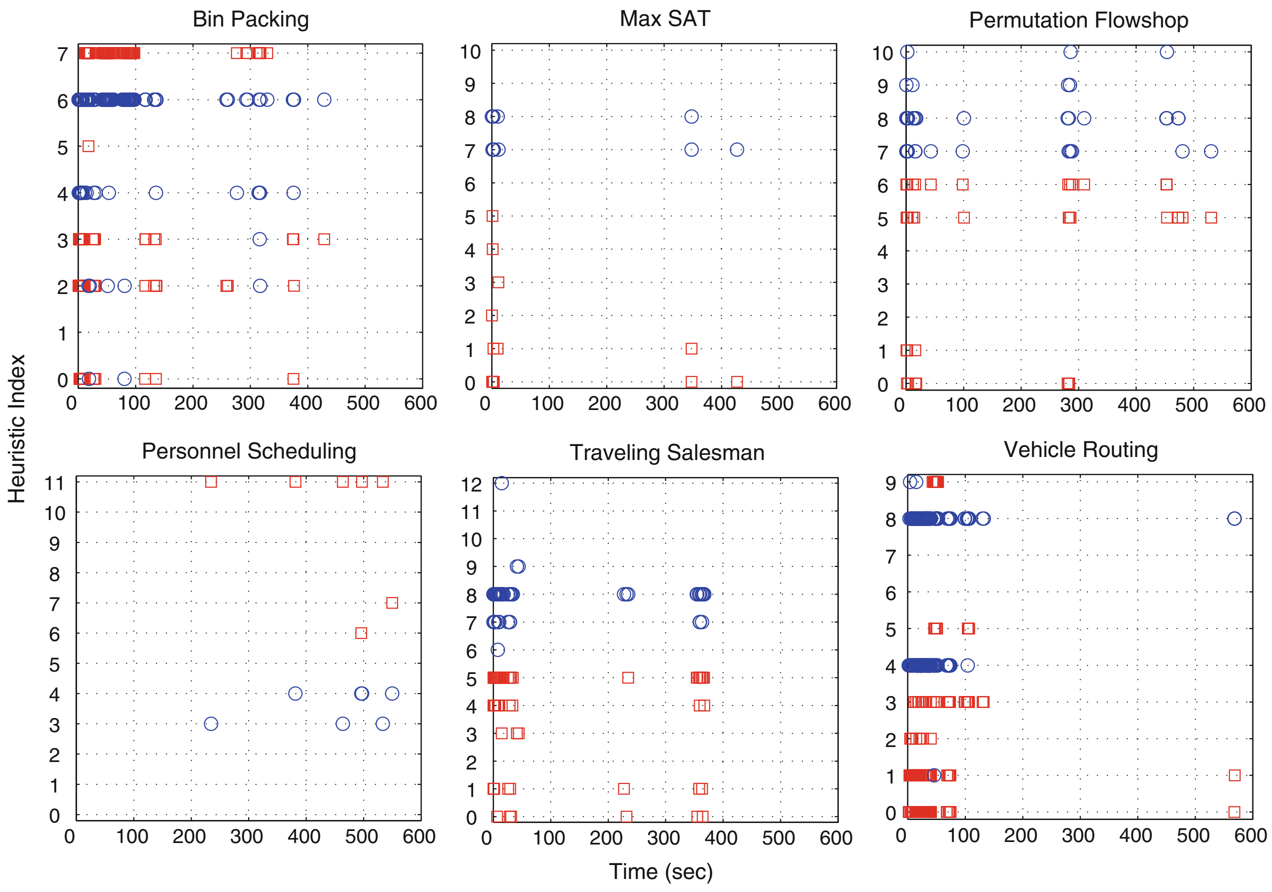

Fig. 5 Heuristic combinations resulting from relay hybridisation on different problem domains (*squares* represent the first heuristics and *circles* refer to the second heuristics in which the heuristics were consecutively applied). Each *graph* was drawn based on data belonging to a run on an instance for each problem domain

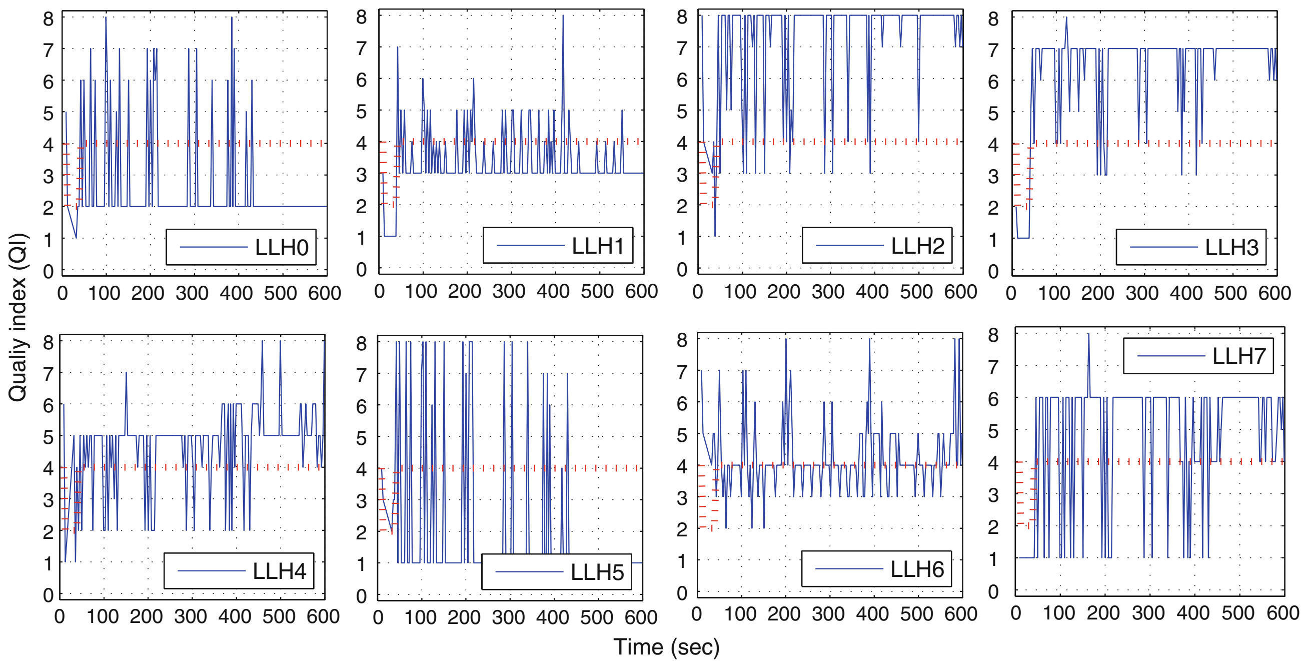

Fig. 6 QI-avg changes of the bin packing heuristics during a run for solving a bin packing problem instance (Quality index (QI) values showing the performance of the heuristics are depicted as *solid lines*. Average (avg) values of all the QI values through the search are indicated as *dotted lines*. The heuristics with a lower QI than "avg" are excluded)

(LLH_7, LLH_8) are the most frequently used second heuristics. Another hill climber (LLH_6) and two crossover operators (LLH_9, LLH_{12}) rarely yielded new best solutions when applied as second heuristics. The mutational heuristics (LLH_0, LLH_1, LLH_3, LLH_4) and the only ruin-recreate heuristic (LLH_5) effectively performed as first heuristics.

The last problem domain, vehicle routing, took advantage of the hybridisation mostly during the first quarter of the search. Two hill climbers (LLH_4, LLH_8) were effective second heuristics. It is also possible to conclude that the other hill climber (LLH_9) and a mutation operator (LLH_1) are seldomly used as second heuristics. Different heuristics performed as first heuristics, two mutational heuristics (LLH_0, LLH_1), the ruin-recreate heuristics (LLH_2, LLH_3), one crossover operator (LLH_5), and one hill climber (LLH_9).

The resulting heuristic pairs exhibit a behavior somewhat similar to memetic algorithms and iterated local search. However, for some problem domains, the effect of these pairs for finding new best solutions is limited. Employing a dynamic decision mechanism in order to determine whether applying single heuristics or heuristics in pairs may be effective to fasten the search process and provide further improvements.

4.4 Varying performance of the heuristics

Figure 6 illustrates the varying QI values of each heuristic during the search process on a bin packing instance. The fluctuating QI suggests that the performance of each heuristic is different for different regions of the search space regarding the utilised performance metric. Among these heuristics a ruin-recreate heuristic (LLH_2) with a mutational heuristic (LLH_3) are mostly available in the heuristic set. On the other hand, two mutational heuristics (LLH_0, LLH_5) are mostly excluded despite their very effective performance during certain phases. During the last quarter of the execution time, these heuristics are always excluded. One hill climber (LLH_4) is excluded from the heuristic set from time to time but during the last quarter of the time it was not excluded anymore. A ruin-recreate heuristic (LLH_1) is frequently excluded, but still provided better performance for some regions of the search space. The last two heuristics, one is hill climber (LLH_6) and one is a crossover (LLH_7), are mostly a part of the heuristic set. However, the QI fluctuations of LLH_6 are smaller compared to the QIs of the other heuristics except QI_1. These results show that the performance variation of different heuristics should be regularly monitored and analysed.

4.5 Set size

As mentioned in the computational results, ADHS elevates the performance of the tested move acceptance strategies significantly. One of the major reasons behind this effective performance improvement is the method determining elite heuristic subsets for different search regions. For the bin packing problem, the number of heuristics used is generally four or five out of eight heuristics. For the max SAT problem, five or six heuristics stay out of the heuristics sets with size 11. The number of heuristics used for solving the permutation flowshop scheduling instances is mostly eight out of 15 heuristics. The heuristic set for the personnel scheduling problem domain could not be processed for elimination due to the low speed of the heuristics and the maximum execution time. For the travelling salesman problem, ADHS keeps the heuristic set as involving mostly seven heuristics out of 13. This value is six for the vehicle routing problem with ten heuristics.

5 Conclusion

In this study, the design and implementation of a new selection hyper-heuristic on a hyper-heuristic software framework, i.e., HyFlex, was discussed. The developed approach consists of an adaptive dynamic heuristic set (ADHS) strategy determining the best heuristic subsets along a number of iterations. For enhancing the effectiveness of ADHS, a learning automaton-based heuristic selection strategy and a pairwise heuristic hybridisation mechanism were employed. In addition, a new threshold-based move acceptance strategy, adaptive iteration limited list-based threshold accepting (AILLA), was accommodated. A set of experiments was carried out with nine additional hyper-heuristics over six problem domains with their specific heuristic sets. The experimental results indicated that the designed hyper-heuristic is an effective strategy for solving the given problem instances using their corresponding heuristic sets. Although the hyper-heuristic performs well on average, there is still room form improvement. For the max SAT problem, SR–SA generates superior results compared to the proposed hyper-heuristic. Also, ADHS–SA performs slightly better than ADHS–AILLA for the personnel scheduling problem. Furthermore, the successor of the proposed hyper-heuristic won the first international cross-domain heuristic search challenge (CHeSC 2011).

In future research, the proposed hyper-heuristic will be improved to cope with its possible adaptation issues for different heuristic sets. For this purpose, a feedback mechanism will be setup between the selection process and the move acceptance part. In addition, a two-phase heuristic selection mechanism consisting of selecting a heuristic subset and choosing a heuristic from the selected heuristic set in connection with a feedback mechanism will be utilised. Furthermore, the adaptive characteristic of the move acceptance strategy will be boosted. Finally, a number of decision mechanisms to manage these sub-mechanisms in collaboration will be developed.

References

Alpaydin, E. (2010). *Introduction to machine learning* (2nd ed.). Cambridge, MA: MIT Press.

Bader-El-Den, M., Poli, R., & Fatima, S. (2009). Evolving timetabling heuristics using a grammar-based genetic programming hyper-heuristic framework. *Memetic Computing, 1*(3), 205–219.

Bilgin, B., Demeester, P., Mısır, M., & Vancroonenburg, W. (2012). One hyperheuristic approach to two timetabling problems in health care. *Journal of Heuristics, 18*(3), 401–434.

Bilgin, B., Demeester, P., Mısır, M., Vancroonenburg, W., Vanden Berghe, G., & Wauters, T. (2010). A hyper-heuristic combined with a greedy shuffle approach to the nurse rostering competition. In *The 8th international conference on the practice and theory of automated timetabling (PATAT'10)—the nurse rostering competition*, Belfast.

Bowling, M., & Veloso, M. (2001). Rational and convergent learning in stochastic games. In *International joint conference on artificial intelligence* (Vol. 17, pp. 1021–1026), Seattle, WA: Citeseer.

Burke, E., & Bykov, Y. (2008). A late acceptance strategy in hill-climbing for exam timetabling problems. In *Proceedings of the 7th international conference on the practice and theory of automated timetabling (PATAT'2008)*, Montreal.

Burke, E., Hart, E., Kendall, G., Newall, J., Ross, P., & Schulenburg, S. (2003a). chap. hyper-heuristics: An emerging direction in modern search technology (pp. 457–474). Handbook of meta-heuristics. Dordrecht: Kluwer.

Burke, E., Hyde, M., & Kendall, G. (2006a). Evolving bin packing heuristics with genetic programming. In T. Runarsson, H.-G. Beyer., E. Burke, J. Merelo-Guervos, L. Whitley, & X. Yao (Eds.), *Proceedings of the 9th international conference on parallel problem solving from nature (PPSN'06)*. LNCS (Vol. 4193 pp. 860–869). Reykjavik: Springer.

Burke, E., Hyde, M., Kendall, G., Ochoa, G., Özcan, E., & Woodward, J. (2010a). A classification of hyper-heuristic approaches. *Handbook of metaheuristics* (pp. 449–468). New York: Springer.

Burke, E., Hyde, M., Kendall, G., & Woodward, J. (2007). Automatic heuristic generation with genetic programming: Evolving a jack-of-all-trades or a master of one. In H. Lipson (Ed.), *Proceedings of genetic and evolutionary computation conference (GECCO'07)* (pp. 1559–1565), London.

Burke, E., Hyde, M., Kendall, G., & Woodward, J. (2010b). A genetic programming hyper-heuristic approach for evolving two dimensional strip packing heuristics. *IEEE Transactions on Evolutionary Computation, 14*(6), 942–958.

Burke, E., Kendall, G., Silva, D., O'Brien, R., & Soubeiga, E. (2005). An ant algorithm hyperheuristic for the project presentation scheduling problem. In *Proceedings of the IEEE congress on evolutionary computation (CEC'05)* (Vol. 3, pp. 2263–2270).

Burke, E., Kendall, G., Mısır, M., & Özcan, E. (2012). Monte carlo hyper-heuristics for examination timetabling. *Annals of Operations Research, 196*(1), 73–90.

Burke, E., Kendall, G., & Soubeiga, E. (2003b). A tabu-search hyper-heuristic for timetabling and rostering. *Journal of Heuristics, 9*(3), 451–470.

Burke, E., Petrovic, S., & Qu, R. (2006b). Case based heuristic selection for timetabling problems. *Journal of Scheduling, 9*(2), 115–132.

Chakhlevitch, K., & Cowling, P. (2005). Choosing the fittest subset of low level heuristics in a hyperheuristic framework. In G. Raidl & J. Gottlieb (Eds.), *Evolutionary computation in combinatorial optimization*. LNCS (Vol. 3448, pp. 23–33). Berlin/Heidelberg: Springer.

Cowling, P., Kendall, G., & Soubeiga, E. (2001). A hyperheuristic approach to scheduling a sales summit. In E. K. Burke & W. Erben (Eds.), *Selected papers from the 3rd international conference on practice and theory of automated timetabling (PATAT'00)*. LNCS (Vol. 2079, pp. 176–190). London: Springer.

Curtois, T., Ochoa, G., Hyde, M., & Vázquez-Rodríguez, J. A. (2010). *A hyflex module for the personnel scheduling problem*. CS Technical Report : University of Nottingham, Nottinghamshire.

Demeester, P., Causmaecker, P. D., & Vanden Berghe, G. (2010). A general approach for exam timetabling: A real-world and a benchmark case. In *Proceedings of the 8th international conference on the practice and theory of automated timetabling (PATAT'10)* (pp. 486–489), Belfast.

Dowsland, K., Soubeiga, E., & Burke, E. (2007). A simulated annealing hyper-heuristic for determining shipper sizes. *European Journal of Operational Research, 179*(3), 759–774.

Fukunaga, A. (2008). Automated discovery of local search heuristics for satisfiability testing. *Evolutionary Computation, 16*(1), 31–61.

Han, L., & Kendall, G. (2003). An investigation of a tabu assisted hyper-heuristic genetic algorithm. In *Proceedings of the IEEE congress on evolutionary computation (CEC'03)* (Vol. 3, pp. 2230–2237).

Hyde, M., & Ochoa, G. (2011). *HyFlex competition instance summary*. CS Technical Report : University of Nottingham, Nottinghamshire.

Hyde, M., Ochoa, G., Curtois, T., & Vázquez-Rodríguez, J. A. (2010a). *A hyflex module for the maximum satisfiability (MAX-SAT) problem*. CS Technical Report : University of Nottingham, Nottinghamshire.

Hyde, M., Ochoa, G., Curtois, T., & Vázquez-Rodríguez, J. A. (2010b). *A hyflex module for the one dimensional bin packing problem*. CS Technical Report : University of Nottingham, Nottinghamshire.

Kendall, G., & Hussin, N. (2005a). An investigation of a tabu-search-based hyper-heuristic for examination timetabling. *Selected papers from the 1st international conference on multidisciplinary scheduling: Theory and applications (MISTA'03)* (pp. 309–328). Nottingham: Springer.

Kendall, G., & Hussin, N. (2005b). A tabu search hyper-heuristic approach to the examination timetabling problem at the mara university of technology. In E. Burke & M. Trick (Eds.), *Proceedings of the 5th international conference on practice and theory of automated timetabling (PATAT'04)*. LNCS (Vol. 3616, pp. 270–293). Berlin: Springer.

Kendall, G., & Mohamad, M. (2004). Channel assignment in cellular communication using a great deluge hyper-heuristic. In *Proceedings of the 12th IEEE international conference on network (ICON'04)* (pp. 769–773).

Lee, D., Vassiliadis, V., & Park, J. (2002). List-based threshold-accepting algorithm for zero-wait scheduling of multiproduct batch plants. *Industrial & Engineering Chemistry Research, 41*(25), 6579–6588.

Marín-Blázquez, J., & Schulenburg, S. (2007). A hyper-heuristic framework with XCS: Learning to create novel problem-solving algorithms constructed from simpler algorithmic ingredients. In T. Kovacs, X. Llorà, K. Takadama, P. Lanzi, W. Stolzmann, & S. Wilson (Eds.), *Selected papers from the international workshops on learning classifier systems (IWLCS'03-05)*. LNCS (Vol. 4399, pp. 193–218). Berlin: Springer.

Mısır, M., Smet, P., Verbeeck, K., & Vanden Berghe, G. (2011a). Security personnel routing and rostering: A hyper-heuristic approach. In *Proceedings of the 3rd international conference on applied operational research (ICAOR'11)*. LNCS (Vol. 3, pp. 193–205), Istanbul.

Mısır, M., Vancroonenburg, W., & Vanden Berghe, G. (2011b). A selection hyper-heuristic for scheduling deliveries of ready-mixed concrete. In *Proceedings of the 9th metaheuristic international conference (MIC'11)*, Udine.

Mısır, M., Verbeeck, K., De Causmaecker, P., & Vanden Berghe, G. (2010). Hyper-heuristics with a dynamic heuristic set for the home care scheduling problem. In *Proceedings of the IEEE congress on evolutionary computation (CEC'10)* (pp. 2875–2882), Barcelona.

Mısır, M., Wauters, T., Verbeeck, K., & Vanden Berghe, G. (2009). A new learning hyper-heuristic for the traveling tournament problem. In *Proceedings of the 8th metaheuristic international conference (MIC'09)*, Hamburg.

Nareyek, A. (2003). Metaheuristics: Computer decision-making, chap. Choosing search heuristics by non-stationary reinforcement learning (pp. 523–544). Dordrecht: Kluwer.

Ochoa, G., Hyde, M., Curtois, T., Vázquez-Rodríguez, J., Walker, J., Gendreau, M., Kendall, G., McCollum, B., Parkes, A., Petrovic, S., & Burke, E. (2012). Hyflex: A benchmark framework for cross-domain heuristic search. In J.-K. Hao & M. Middendorf (Eds.), *Proceedings of the 12th European conference on evolutionary computation in combinatorial optimisation (EvoCOP'12)*. LNCS (Vol. 7245, pp. 136–147). Berlin: Springer.

Özcan, E., Bilgin, B., & Korkmaz, E. (2008). A comprehensive analysis of hyper-heuristics. *Intelligent Data Analysis*, *12*(1), 3–23.

Özcan, E., Mısır, M., Ochoa, G., & Burke, E. (2010). A reinforcement learning–Great-deluge hyper-heuristic for examination timetabling. *International Journal of Applied Metaheuristic Computing*, *1*(1), 39–59.

Ross, P., Schulenburg, S., Marín-Blázquez, J., & Hart, E. (2002). Hyper-heuristics: Learning to combine simple heuristics in bin-packing problems. In W. B. Langdon, E. Cantú-Paz, K. Mathias, R. Roy, D. Davis, R. Poli, K. Balakrishnan, V. Honavar, G. Rudolph, J. Wegener, L. Bull, M. A. Potter, A. C. Schultz, J. F. Miller, E. Burke, & N. Jonoska (Eds.), *Proceedings of the 4th annual conference on genetic and evolutionary computation (GECCO'02)* (pp. 942–948). New York: Morgan Kaufmann Publishers.

Thathachar, M., & Sastry, P. (2004). *Networks of learning automata: Techniques for online stochastic optimization*. Boston: Kluwer.

Vázquez-Rodríguez, J. A., Ochoa, G., Curtois, T., & Hyde, M. (2010). *A hyflex module for the permutation flow shop problem*. CS Technical Report : University of Nottingham, Nottinghamshire.

Walker, J., Ochoa, G., Gendreau, M., & Burke, E. (2012). Vehicle routing and adaptive iterated local search within the HyFlex hyper-heuristic framework. In Y. Hamadi & M. Schoenauer (Eds.), *Proceedings of the 6th learning and intelligent optimization conference (LION'12)*. LNCS (Vol. 7219). Paris: Springer.