Hyper-heuristics with a Dynamic Heuristic Set for the Home Care Scheduling Problem

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

Abstract— **A hyper-heuristic performs search over a set of other search mechanisms. During the search, it does not require any problem-dependent data. This structure makes hyper-heuristics problem-independent indirect search mechanisms. In this study, we propose a learning strategy to explore elite heuristic subsets for different phases of a search. For that purpose, we apply a number of hyper-heuristics with the proposed approach to a set of home care scheduling problem instances. The results show that the learning strategy increases the performance of the different hyper-heuristics by excluding some heuristics from the heuristic set over the tested problem instances.**

I. Introduction

Hyper-heuristics are effective high-level search mechanisms [1], [2], [3]. They perform search over a space of heuristics instead of solutions. This characteristic is supported with a domain barrier between hyper-heuristics and problem domains. The domain barrier is used to provide problem-independency by preventing any problem-dependent data flow. The duty of hyper-heuristics is to enable efficient management over a number of utilised heuristics in a problem independent manner. By using the strengths and preventing the weaknesses of multiple heuristics, it can be possible to provide improvement over applying a number of single heuristics to a problem instance.

Different heuristics can have different abilities and these abilities may change over a search space. During the search, different heuristics are selected with the aim of using the most appropriate one for the current region in the search space. There is often no heuristic that performs best everywhere. Hence, using the heuristics when they perform better may result in a more efficient solution strategy. Hyper-heuristics search for such a collaboration between the heuristics from a heuristic set.

There are two main types of hyper-heuristics. These hyper-heuristics are expressed as "*heuristics to choose heuristics*" and "*heuristics to generate heuristics*". The first category of hyper-heuristics includes a heuristic selection mechanism(s) and a move acceptance mechanism(s). Heuristic selection aims to choose the best possible heuristic(s) based on the current state of the search. A move acceptance mechanism is used to decide whether to accept or reject a visited/constructed solution(s) by the selected heuristic. Choice Function [4], Reinforcement Learning [5], Tabu Search [6], Case-based Reasoning [7] are some examples of heuristic selection. Also, population based approaches such as Genetic Algorithms [8] and Ant Colony Optimisation [9] can be used. Simulated Annealing [10], Monte Carlo [11], Great Deluge [12], Record-to-Record Travel [13], Late Acceptance [14] are the move acceptance mechanisms that have been used within hyper-heuristics.

Mustafa Mısır, Katja Verbeeck and Greet Vanden Berghe are with CODeS, KaHo Sint-Lieven, Gebroeders Desmeetstraat 1, 9000 Gent, Belgium (phone: +32-9-2658704; e-mail: {mustafa.misir, katja.verbeeck and greet.vandenberghe}@kahosl.be).

Patrick De Causmaecker is with CODeS, Department of Computer Science, Katholieke Universiteit Leuven, Campus Kortrijk, Etienne Sabbelaan 53, 8500 Kortrijk, Belgium (phone: +32-56-246002; e-mail: patrick.decausmaecker@kuleuven-kortrijk.be).

For the heuristic generation, in [15], [16], [17] Genetic Programming is employed to construct high quality heuristics dedicated to solving a target problem. In addition to that, it is being used as a construction mechanism for hyper-heuristic components. In [18], it was presented how a problem specific move acceptance mechanism can be generated. More about hyper-heuristics can be found in a recent survey paper [19] and a classification paper [20].

In this study, we propose a simple learning mechanism to exclude some heuristics during a given number of phases. We employed this strategy within a number of hyper-heuristics and tested it over a set of home care scheduling problem instances. The home care scheduling problem is a relatively new vehicle routing variant. It requires assigning a set of carers starting from different locations to a number of tasks requiring various skills, while respecting numerous constraints. Due to the routing characteristic of the problem, the total traveling time should be minimised. In addition to that, the total idle time of the carers should be optimised. The experimental results show that excluding worse performing heuristics is useful to find elite heuristic subsets in different phases of a search.

In the remainder of the paper, the proposed dynamic heuristic set strategy is explained (Section 2). Then, in Section 3, the details about the tested hyper-heuristics are provided. After that, the problem definition is given in Section 4. Next, the paper continues with experimental results in Section 5. Finally, the paper is concluded and future research directions are discussed.

II. Learning the Heuristic Set

The performance of a hyper-heuristic depends on the set of heuristics it can choose from. A heuristic set composed of specialised heuristics may be better than a heuristic set with weak/simple heuristics. Hyper-heuristics should have the ability to use the available potential of a heuristic set. That is, even if the heuristic set does not involve any specific heuristic or if the heuristic set is not highly specialised, hyper-heuristics should find a way to manage them in the

978-1-4244-8126-2/10/$26.00 ©2010 IEEE

best possible way. Also, due to the performance changes of a number of heuristics over a search space, it is not easy to find a heuristic that always produces the best decisions. For these reasons, different learning strategies have been employed to make better selections of heuristics. It may be useful to look for effective exclusion opportunities and determine elite heuristic sets during a search.

Using reduction or prohibition mechanisms in hyper-heuristics is not new. In [21], 95 low-level heuristics are automatically reduced to a small set to find the best possible heuristic set among them. In [6], tabu search with reinforcement learning was used for the heuristic selection part. Heuristics are ranked based on their improvement capabilities. The same data are used to make heuristics tabu or non-tabu. At each iteration, a non-tabu heuristic with the highest rank is selected. In [22], different fixed tabu durations are used to make heuristics tabu for at most 4 iterations. In [23], a similar idea was combined with different hyper-heuristics. In addition to the deterministic tabu durations, the experiments were repeated with a random dynamic tabu duration mechanism over a set of examination timetabling benchmarks. In the random version, tabu duration is selected randomly from a predetermined range. In [24], a genetic algorithm based hyper-heuristic was proposed. Each individual (chromosome) refers to a number of heuristics that are consecutively applied to a solution. The aim is to find the best heuristic combination, i.e. the best individual. Based on the performance of each heuristic located in a chromosome, some heuristics are prevented from being applied for a number of generations.

In this study, we developed a simple learning approach to exclude some heuristics for a number of phases. Phases are determined by a number of iterations. After n iterations, a snapshot of the heuristics' performance is taken. Based on that knowledge, some heuristics are excluded.

A. The Learning Strategy

In [25], heuristics are scored based on their performances. The average of all scores was used to divide the heuristic set into two. Heuristics with a higher or equal score than the average value are considered to be improving heuristics. Heuristics with a lower value than the average are considered as deteriorating heuristics. Instead of directly using such a scoring approach, we will learn a value so called quality index (QI) for each heuristic. This value expresses the quality of the heuristics during one phase and helps to determine performance differences. If the employed performance metric is different from zero for heuristic i, then its QI_i is set to a value based on its order among the other heuristics. Otherwise, QI_i is automatically set to 1. Better heuristics have higher QI values. The highest possible value is equal to the number of heuristics (n) and the lowest value is 1. At the end of each phase, QI values are set and the average QI (avg) is calculated. For the calculation of avg, tabu heuristics additionally contribute with $QI = 1$. This helps the learning strategy to keep a number of heuristics non-tabu under any case. The heuristics with a QI value smaller than avg are excluded from the set for a pre-determined number of phases (d: tabu duration).

$$avg = \left\lfloor \Big(\sum_{i}^{n} QI_i \Big) / n \right\rfloor \tag{1}$$

Although the QI values rank the heuristics based on given performance criteria, they do not reflect the exact performances of the heuristics. However, QI values are effective and accurate enough for the proposed learning strategy.

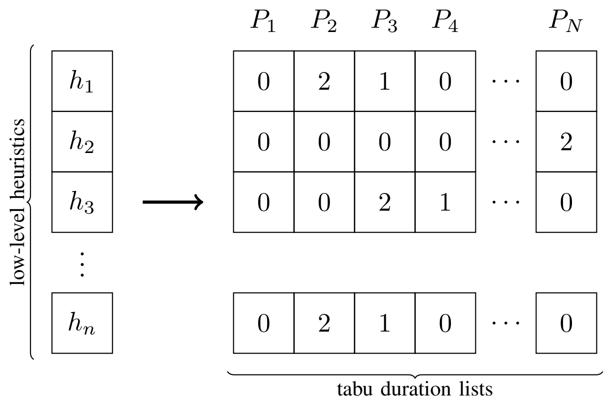

Fig. 1. An example of the prohibition mechanism with tabu duration of 2 ($d = 2$)

In Figure 1, an example of the learning mechanism is shown. The P_i denote the phases. At each phase, the tabu duration of each heuristic is updated if necessary. That is, if a non-tabu heuristic is determined to be tabu for the next phase, its tabu duration is set to a fixed value (d). If a heuristic is tabu, its tabu duration is decremented by 1 after every phase until it is zero. In the given example, during P_1, all the heuristics are non-tabu. In the second phase, h_1 and h_n have been set tabu. They are excluded from the heuristic set for the next two phases (tabu duration = 2). In the next phase, their tabu durations are decreased by 1 and another heuristic, h_3, is excluded due to its performance. This procedure continues until the termination criteria are met.

B. Performance Metrics Involved in Learning

There are different elements to measure the performance of a heuristic. These elements are mainly related to improvement capabilities of the employed heuristics. However, taking only the improvement/worsening behavior of the heuristics into account can be misleading. It can be useful to additionally consider the speed of the heuristics during generating a new solution.

We decided to use three dependent performance metrics for determining which heuristics are tabu. M_1) The first comparison is made by checking the capabilities of the heuristics for improving the current best solution. Heuristics with a higher number of improvements over the current best solution have higher QI values. M_2) In the case of not providing any new best solutions, the QI values of the heuristics are determined based on their improvement

Algorithm 1: The Learning Strategy

n: the number of heuristics; d: fixed tabu duration value; num_of_iter: current iteration number; ph_{iter}: the phase length in terms of the number of iterations during the current phase; $count(i)_{newBest}$: the number of new best solutions found by heuristic i during the current phase; imp_i: the total improvement provided by heuristic i during the current phase; wrs_i: the total disimprovement caused by heuristic i during the current phase; t_i: the total time spent by heuristic i during the current phase; $TDList(i)$: the tabu duration of heuristic i

Input: $d > 0 \wedge ph_{iter} > 0$

```
1  if $(num\_of\_iter \% ph_{iter}) = 0$ then
2      for $i \leftarrow 1$ to $n$ do
3          $M_1(i) = count(i)_{newBest}$;
4          if $t_i > 0$ then
5              $M_2(i) = imp_i / t_i$;
6              $M_3(i) = -(wrs_i / t_i)$;
7          else
8              $M_2(i) = M_3(i) = 0$;
           end
       end
9      for $i \leftarrow 1$ to $n$ do
10         for $j \leftarrow 1$ to 3 do
11             if $M_j(i) \neq 0$ then
12                 $QI_i = 1$;
13                 for $k \leftarrow 1$ to $n$ do
14                     if $i \neq k$ and $TDList(k) = 0$ and
                       $QI_k = 0$ then
15                         if $M_j(i) > M_j(k)$ then
16                             $QI_i = QI_i + 1$;
17                         else if $j = 1$ and
                           $M_j(i) = M_j(k)$ and
                           $M_2(i) > M_2(k)$ then
18                             $QI_i = QI_i + 1$;
                           end
                       end
                   end
19                 break;
               end
           end
20         if $QI_i = 0$ then
21             $QI_i = 1$;
           end
       end
22     for $i \leftarrow 1$ to $n$ do
23         if $TDList(i) = 0$ then
24             if $QI_i < avg(QI)$ then
25                 $TDList(i) = d$;
               end
26         else
27             $TDList(i) = TDList(i) - 1$;
           end
       end
   end
```

capabilities per unit execution time. M_3) If some of the heuristics do not provide any improvement but worsen the solution, then the QI values of these heuristics are determined based on their worsening characteristics per unit execution time.

- M_1 : Number of new best solutions found
- M_2 : Total fitness improvement / execution time
- M_3 : Total fitness disimprovement / execution time

Whenever num_of_iter reaches ph_{iter} and its multiples, the three performance metrics are checked for the non-tabu heuristics and based on these values, the QI value of each heuristic is determined. After that, some heuristics are made tabu for d phases and the tabu durations of the tabu heuristics are decreased by 1.

C. Size of the Heuristic Subsets

Based on the proposed learning strategy, the size of a heuristic subset or the number of non-tabu heuristics have limits. If none of the heuristics makes a change to a solution, then the QI value of each heuristic is set to 1. This means that avg is 1. Since $avg = 1$ is the lowest possible value, no heuristic will be excluded. On the other hand, if all the heuristics are non-tabu and all have a different performance, then $avg = (n+1)/2$. If $avg \geq 2$, then at least one heuristic is excluded. From this limit, we can derive the following formula (2).

$$\frac{(n-x) + x(x+1)/2}{n} \geq 2 \tag{2}$$

After simplification it turns into (3).

$$x(x-1) \geq 2n \tag{3}$$

In this formula x is the total number of non-tabu heuristics. If this formula is valid, then it is possible to exclude some heuristics. Whenever it is violated, the size of the heuristic subset reaches its lowest limit. For instance, for a large enough tabu duration ($d = \propto$), which prevent the tabu heuristics from being added back to the heuristic set before the number of tabu heuristics reaches its maximum limit, the minimum size of a heuristic subset is 3 for n=5, 4 for n=10 and 14 for n=100. As seen from the given examples, the minimum heuristic set for small n is some kind of exclusion mechanism. On the other hand, for a larger n, it behaves like a temporary heuristic reduction strategy. Of course, for larger heuristic sets, higher tabu duration values should be utilised. Otherwise, the number of non-tabu heuristics will be too large for an elite heuristic set. The phase length should also be determined based on the number of heuristics. For instance, if $n = 100$, using $ph_{iter} = 1000$ is not a reasonable choice. At best, each heuristic will be called only 10 times. Thus, making a meaningful comparison between the heuristics will not be possible, especially if most of the heuristics are non-tabu.

III. Hyper-heuristics

We employed 3 perturbative selection hyper-heuristics in different configurations from those in [26]. All the tested hyper-heuristics use the Simple Random (SR) heuristic selection mechanism. Since SR selects heuristics in a random uniform manner, all the heuristics get equal opportunity to be explored. For the move acceptance part, one hyper-heuristic involves Improving or Equal (IE). The other two use Iteration Limited Threshold Accepting (ILTA) [27] and Adaptive ILTA (AILTA) move acceptance strategies. The aim is to show performance changes over a number of hyper-heuristics.

Algorithm 2: ILTA Move Acceptance

```
Input: $k \geq 0 \wedge R \in (1.00 : \infty)$
1 if $f(S') < f(S)$ then
2     $S \leftarrow S'$;
3     $w\_iterations = 0$;
4 else if $f(S') = f(S)$ then
5     $S \leftarrow S'$;
6 else
7     $w\_iterations = w\_iterations + 1$;
8     if $w\_iterations \geq k$ and $f(S') < f(S_b) \times R$ then
9         $S \leftarrow S'$ and $w\_iterations = 0$;
      end
  end
```

The different move acceptance mechanisms considered here simply work as follows:

- ***IE*** : Accepts improving or equal quality solutions.
- ***ILTA*** : Accepts improving or equal quality solutions; in addition to that, accepts worsening solutions based on an iteration limit and a threshold value.
- ***AILTA*** : Adaptive version of ILTA. Adapts the threshold value if the current best solution is not improved.

In Algorithm 2, the pseudo-code of ILTA is given. k is the iteration limit for consecutive worsening moves to accept a worsening solution. R is a range value used to calculate a threshold level. S is the current solution, S' is the new solution found and S_b is the current best solution. $f(S)$, $f(S)$ and $f(S_b)$ denote the fitness values of these solutions. $w_{iterations}$ is the number of iterations denoting the consecutive worsening solutions (intermediate equal quality solutions are ignored).

ILTA has similarities with Record-to-Record Travel (RRT) [28] and Sequence Heuristic (SH) [29]. The similarity between ILTA and RRT is related to how the threshold values are calculated. In RRT, this value is calculated as $f(S_b) + Deviation$ where $Deviation$ is utilised for determining a threshold value for the acceptance process. ILTA also calculates the threshold by taking the fitness of the current best solution into account and employs a dynamic parameter of $Deviation$ using a fixed R. In [30], a similar dynamic approach was applied to the heterogeneous fleet vehicle routing problem. Concerning the iteration limit, ILTA resembles SH. It checks a fixed number of solutions before accepting a worsening move, just as SH does. Since each solution has a lot of neighboring solutions, accepting a worsening neighbour whenever encountering one can result in missing some good intermediate solutions. ILTA aims at handling this problem by employing reasonable iteration limits.

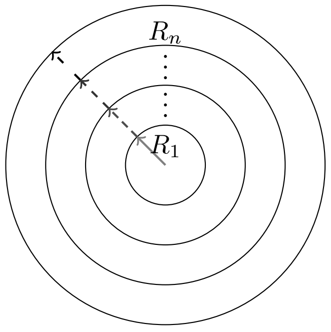

Fig. 2. Extending the acceptable search space region for worsening solutions by increasing the range value from $R_1 \mapsto R_n$

AILTA is an adaptive version of ILTA that increases the value of R in case it is hard to improve the current best solution. For that purpose, another iteration limit is added. This limit determines when R should be increased. To prevent the value of R of becoming very large, an upper bound for R is provided. Whenever the current best solution is improved, R is set to its initial value. The adaptation process is visualised in Figure 2.

IV. The Home Care Scheduling Problem

The home care scheduling problem considers assigning a number of carers to a number of patients who are located at different places requiring a number of tasks to be tackled. The problem is relatively new in the research community and it has not been widely studied. In the literature, the problem differs based on the type of the tasks. The home care scheduling problem considers, next to nursing/medical tasks, also general tasks such as cleaning. The more restrictive form, the home health care problem, only considers health related tasks. In [31], a spatial decision support system was developed to solve the home care problem over a five-day planning horizon. In [32], [33], [34], the daily problem is modeled like a multi depot vehicle routing problem with time windows, by considering each nurse's home as a separate depot. In [35], weekly home care schedules were constructed by assigning patients based on their care plans without fixed time windows constraints. In [36], particle swarm optimisation was used to determine exact visit times concerning assigned tasks for a home care problem. In [37], [38], a software system called LAPS CARE that focuses on home care by using different activity types such as cleaning for mostly elderly people was presented. In [39], a VRP software was applied to solve different scenarios of the home care problem. In [40], the home health care problem was modeled as a master schedule problem, which aims to decrease the number of routes, and an operational planning problem that is utilised to deal with immediate changes. In [41], the health care delivery problem for chemotherapy patients at their home was modeled based on a supply chain strategy involving drug production, distribution and administration phases whilst respecting the drugs' shelf life.

The problem resembles the Vehicle Routing Problem with Time Windows [42]. However, the home care scheduling problem includes more constraints and different skills to cover the given tasks. These characteristics are listed as;

For carers:

- ***Transportation***: Using different transportation means (car, bicycle, walking) to travel
- ***Starting Location***: The starting location concerning carers' working time can differ (a central depot, home or first visit)
- ***Working Time Window***: Each carer has a duty start time and a duty end time
- ***Skill Types***: Each carer has different skills
- ***Excluded Patients***: Some patients can be excluded for some nurses

For tasks:

- ***Locked Tasks***: Some tasks are locked to a specific carer
- ***Skill Requirement***: Each task has different skill requirements
- ***Time Window***: Each task has a time window
- ***Duration***: Each task has a duration
- ***Preferred Carers***: Some tasks have preferred carers
- ***Connected Tasks***: Some tasks can be related to each other. For instance, two related tasks may require to start at the same time

A. Fitness Function

The problem is tackled using a single fitness function that equals a weighted sum of objectives and constraint violations. The objectives that should be minimized are;

- Total traveling time
- Total idle time due to early arrival with respect to the task time windows and remaining time to the end of the working time window

Since feasible solutions are not guaranteed in case all the constraints are hard, we consider the following constraint violations;

- Task time window violation
- Working time window violation
- Skill requirement violation
- Assignment of unpreferred tasks
- Assignment of unpreferred carers
- Connected task violation

For more details about the fitness function, check [26].

B. Low-level Heuristics

The same low-level heuristics as in [26] are used. The details are as follows;

$\boldsymbol{LLH_1}$: Swap visits between two routes
$\boldsymbol{LLH_2}$: Swap visits within a route
$\boldsymbol{LLH_3}$: Move a visit to the best place within another route
$\boldsymbol{LLH_4}$: Select a route and find the visit that provides the best improvement when removed and move it to the best place within another route
$\boldsymbol{LLH_5}$: Select a route and find the visit that provides the best improvement when removed and move it to the best place within the best route that is determined based on an individual objective value of each route
$\boldsymbol{LLH_6}$: Select a route and find the visit that provides the best improvement when removed and move it to the best place within the most idle route

In fact, promising experiments with a larger set of low-level heuristics, i.e. 32 instead of 6, have been carried out. However, it turned out that many of these 32 low-level heuristics never led to any improvement.

V. Experiments

We tested the hyper-heuristics over 6 home care scheduling problem instances on Pentium Core 2 Duo 3 GHz PCs with 3.23 GB memory. The tests are repeated 10 times. The phase lengths are set to $\{1000, 2500, 5000, 10000\}$ iterations. For the tabu duration, we use 1. For ILTA, k is 100. For AILTA, the second iteration limit is set to 5000 and the range value is adapted ($R_1 \mapsto R_n$) by an incrementing factor 1.001. The parameters regarding the move acceptance part are the same as in [26]. These parameters were determined after some preliminary experiments. No special parameter tuning process was applied.

In Table I, the details about the home care scheduling instances are given. Among them, the *hh*, *ll2* and *ll3* instances are taken from [43]. The remaining instances, *hh1*, *ll11* and *ll21*, are the modified versions of the original *hh*, *ll1* and *ll2*.

TABLE I
THE HCSP INSTANCES

Bench.	Num. of Carers	Num. of Tasks	Num. of Patients
hh	15	154	74
hh1	15	150	74
ll11	9	104	59
ll2	7	61	30
ll21	7	60	30
ll3	7	61	30

A. Computational Results

In the following tables the average fitness (AF) values and the average ranks (AR) belonging to different phase lengths concerning each hyper-heuristic are provided. The experimental results with "N/A" denote the hyper-heuristics with no dynamic heuristic set.

In Table II(a), the results for SR-AILTA with R_1=1.003 and R_n=1.007 are given. The average rank values show that all the tested phase lengths improve the performance of the hyper-heuristic with an initially formed fixed heuristic set. Among them, 10000 looks like the best option. In Table II(b), the ranking of the same hyper-heuristic with R_n=1.01 is presented. For this hyper-heuristic configuration, 2500 is a useful phase length for determining a dynamic heuristic set. In Table II(c), again the same hyper-heuristic was tested

TABLE II

AVERAGE FITNESS VALUES WITH RANKS OF DIFFERENT HYPER-HEURISTICS OVER 6 HCSP BENCHMARK INSTANCES

(a) SR-AILTA with $R_1 = 1.003$ $R_n = 1.007$

AF	**PHASE LENGTH**				
	N/A	**1000**	**2500**	**5000**	**10000**
hh	75121,94	74049,25	74831,22	74772,85	74378,76
hh1	43645,46	46726,00	44719,40	43466,85	42706,44
ll11	50458,76	47392,77	46408,61	47675,16	44787,40
ll2	9873,51	9650,13	9332,27	9420,47	9352,76
ll21	8732,83	9233,70	9174,38	9432,25	9330,76
ll3	8775,59	8843,06	8724,14	8688,68	8815,87
AR	3,67	3,50	2,50	3,00	**2,33**

(b) SR-AILTA with $R_1 = 1.003$ $R_n = 1.01$

AF	**PHASE LENGTH**				
	N/A	**1000**	**2500**	**5000**	**10000**
hh	74213,60	73995,66	74654,60	74808,14	74220,70
hh1	42654,80	43415,05	43942,63	43157,50	43001,02
ll11	45264,19	44730,06	44677,35	46197,30	44996,57
ll2	9712,77	9491,63	9326,69	9369,95	9322,01
ll21	8739,54	9194,20	8708,62	8935,51	8829,87
ll3	8636,19	8699,11	8500,31	8628,37	8789,47
AR	2,83	3,33	**2,33**	3,67	2,83

(c) SR-AILTA with $R_1 = 1.004$ $R_n = 1.007$

AF	**PHASE LENGTH**				
	N/A	**1000**	**2500**	**5000**	**10000**
hh	75685,42	74241,68	74200,68	74634,61	74019,18
hh1	43837,95	43675,98	43155,15	43448,52	43301,90
ll11	47291,00	47212,63	42774,24	44854,42	49183,66
ll2	9776,71	9488,96	9296,00	9393,11	9478,91
ll21	9057,60	8919,95	8947,66	9014,98	9086,86
ll3	8645,29	8574,99	8659,42	8716,54	8819,17
AR	4,17	2,67	**1,67**	3,00	3,50

(d) SR-ILTA with R=1.003

AF	**PHASE LENGTH**				
	N/A	**1000**	**2500**	**5000**	**10000**
hh	75958,19	78077,78	74911,15	78822,38	75234,02
hh1	50103,33	45625,38	45637,32	45234,76	45057,79
ll11	53289,69	46101,79	50750,60	47830,82	49135,72
ll2	10187,80	9703,55	9661,95	9652,46	9611,41
ll21	9016,99	9462,47	9474,32	9686,57	9595,09
ll3	8995,84	8984,76	9368,15	8998,98	8995,41
AR	3,67	2,50	3,33	3,33	**2,17**

(e) SR-ILTA with R=1.004

AF	**PHASE LENGTH**				
	N/A	**1000**	**2500**	**5000**	**10000**
hh	75155,32	77448,89	74835,87	75125,33	74513,35
hh1	44794,36	44407,52	44508,57	44662,96	44457,79
ll11	51109,00	48684,60	45194,42	45532,99	46548,59
ll2	10047,85	9599,21	9439,91	9685,93	9609,40
ll21	9303,31	9387,10	9433,68	9765,26	9233,92
ll3	8752,16	8793,18	8943,88	8933,82	9103,44
AR	3,67	2,83	**2,50**	3,50	**2,50**

(f) SR-IE

AF	**PHASE LENGTH**				
	N/A	**1000**	**2500**	**5000**	**10000**
hh	89286,77	87083,58	89609,35	88606,49	87620,99
hh1	54778,81	55659,93	53259,56	50893,50	53326,32
ll11	104670,45	76356,56	81872,82	85122,85	80558,46
ll2	10339,69	9826,82	9909,80	15365,12	15353,77
ll21	9418,10	9512,55	9816,59	9527,36	9507,30
ll3	9395,24	9570,64	9770,43	9784,44	9765,54
AR	3,00	**2,17**	3,50	3,67	2,67

with R_1=1.004 and R_n=1.007. Learning heuristic subsets for different phases on average generates superior results. All the tested phase lengths result in improvements. Among them, 2500 shows the best performance based on average ranks.

In Table II(d), the average performance of SR-ILTA with R=1.003 is indicated. The ranking values based on the average fitness values show that all the phase length values are useful for determining elite heuristics subsets. Among them, 10000 performs best. In Table II(e), the performance of the same hyper-heuristic with R=1.004 was improved using different phase lengths. Among them, 2500 and 10000 have the same average rank as the best options.

The last hyper-heuristic, SR-IE, was also tested for different phase lengths. For this hyper-heuristic, 1000 provides the best improvement with respect to the average ranks.

The comparisons are made based on a fixed phase length value. In the case of taking different phase lengths into account for the same hyper-heuristic configuration, the performance improvement coming from the learning strategy appears to be higher.

In Table III, the average best results for the tested instances are shown. All the average best results after 10 trails excluding the *hh1* were improved. In addition, in Table IV, some new best results are provided. The current best result of the *hh1* could not be improved and for the *ll3* the previous best result is found again. For the rest, new best results were reached.

TABLE III

THE AVERAGE BEST RESULTS FOR THE HCSP INSTANCES

Bench.	HH	Phase Length	Fitness
hh	AILTA R_1=1.003 R_n=1.01	1000	73995,66
hh1	AILTA R_1=1.003 R_n=1.01	N/A	42654,80
ll11	AILTA R_1=1.004 R_n=1.007	2500	42774,24
ll2	AILTA R_1=1.004 R_n=1.007	2500	9296
ll21	AILTA R_1=1.003 R_n=1.01	2500	8708,62
ll3	AILTA R_1=1.003 R_n=1.01	2500	8500,31

TABLE IV

THE BEST RESULTS FOR THE HCSP INSTANCES

Bench.	HH	Phase Length	Fitness
hh	AILTA R_1=1.003 R_n=1.01	10000	71981,80
hh1	AILTA R_1=1.004 R_n=1.007	N/A	40644,60
ll11	AILTA R_1=1.003 R_n=1.007	2500	39046,80
ll2	AILTA R_1=1.004 R_n=1.007	1000	8926,25
ll21	AILTA R_1=1.003 R_n=1.01	2500	8372,40
ll3	ILTA R=1.003	N/A	8047,15
	AILTA R_1=1.003 R_n=1.01	2500	
	AILTA R_1=1.004 R_n=1.007	2500	

In Figure 3, sample QI values for each heuristic by SR-AILTA $R_1 = 1.003$ $R_n = 1.007$ with $ph_{iter} = 2500$ are presented. These values show how the performance of the heuristics changes in time. Especially, LLH_4 and LLH_5 perform worse than the other heuristics. This causes them to be frequently excluded from the heuristic set.

All these results show that the proposed learning strategy

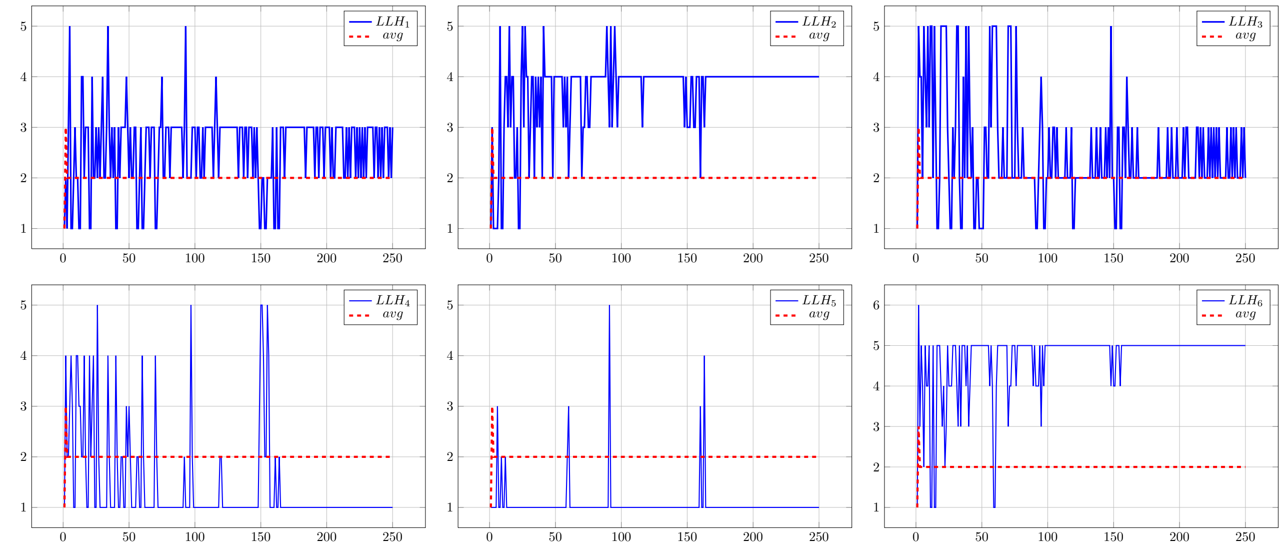

Fig. 3. Sample QI values during the first 250 phases over the hh by SR-AILTA $R_1 = 1.003$ $R_n = 1.007$ with $d = 1$ and $ph_{iter} = 2500$

can provide improvement over different hyper-heuristics. SR-IE without a dynamic heuristic set has been improved by the learning strategy. For the other hyper-heuristics that perform significantly better than SR-IE, it is still possible to provide improvement via the dynamic heuristic set. That is, regardless of the characteristics of the hyper-heuristics, the learning strategy can help them to fill the gap between their current performance and their potential on a given heuristic set.

TABLE V
SR-AILTA WITH $R_1 = 1.003$ $R_n = 1.01$ WITH $ph_{iter} = 2500$ AND $d = \{1, 2, 3, 4, 5, 6\}$

AF	TABU DURATION					
	1	2	3	4	5	6
hh	74654.60	74294.45	73848.02	74457.14	73771.55	74643.89
hh1	43942.63	43213.85	43096.84	43675.20	43747.53	44921.41
ll11	44677.35	44080.03	42614.54	43730.65	45522.86	44714.58
ll2	9326.69	9410.14	9406.52	9356.36	9358.18	9372.62
ll21	8708.62	8733.55	8780.35	9168.57	8906.54	9154.57
ll3	8500.31	8604.94	8717.98	8574.86	8668.40	8610.33
AR	**3,00**	3,17	**3,00**	3,17	3,83	4,83

The computational results were generated using a tabu duration of 1. This means that each excluded heuristic returns to the heuristic set just after one phase. For that reason more experiments were carried out to show the effect of different tabu durations $d = \{2, 3, 4, 5, 6\}$. However, we did not see a significant performance difference among the hyper-heuristics with $d = \{1, 2, 3, 4\}$. The performance decrease starts with $d = 5$. With $d = 6$, the quality of the solutions decreases in a significant way. This shows that there can be a range of tabu durations behaving similarly. Out of that range, the performance of the hyper-heuristic can decrease significantly.

VI. Conclusions

In this study, we propose a learning mechanism for excluding some heuristics during different phases of the search. The main aim is to find proper heuristic sets that can be used during different phases of the search. The proposed mechanism helps to determine elite heuristic subsets during each phase. This approach can be considered as a learning mechanism that is repeated at each phase. Any information learnt during a phase is forgotten in the next phase. To test the effectiveness of the idea, the hyper-heuristics are used with different phase lengths regarding the learning strategy over a set of home care scheduling problem instances.

The experimental results over 6 hyper-heuristic configurations show that this mechanism successfully determines the performance difference of the utilised heuristics during each phase. Simple rankings of the average fitness values for 10 trials indicate the performance improvement of the learning strategy.

In the future, we will work on an adaptation approach for determining the best phase lengths. The effects of different performance criteria will be investigated. We will combine it with a general learning mechanism for learning about the performance of heuristics over the entire search space in order to increase their effectiveness. In addition, a coupling strategy will be designed for better cooperation between the dynamic heuristic set approach and the move acceptance mechanisms. Furthermore, the behavior of the learning strategy will be investigated over larger heuristic sets.

References

[1] E.K. Burke, E. Hart, G. Kendall, J. Newall, P. Ross, and S. Schulenburg, *Handbook of Meta-Heuristics.* Kluwer Academic Publishers, 2003, ch. Hyper-Heuristics: An Emerging Direction in Modern Search Technology, pp. 457–474.
[2] P. Ross, *Search Methodologies.* Springer, 2005, ch. Hyper-Heuristics, pp. 529–556.
[3] E. Ozcan, B. Bilgin, and E. Korkmaz, "A comprehensive analysis of hyper-heuristics," *Intelligent Data Analysis*, vol. 12, no. 1, pp. 3–23, 2008.
[4] P. Cowling, G. Kendall, and E. Soubeiga, "A hyperheuristic approach to scheduling a sales summit," in *PATAT '00: Selected papers from the*

Third International Conference on Practice and Theory of Automated Timetabling III. London, UK: Springer-Verlag, 2001, pp. 176–190.

[5] A. Nareyek, "Choosing search heuristics by non-stationary reinforcement learning," in *Metaheuristics: Computer Decision-Making*. Kluwer Academic Publishers, 2003, pp. 523–544.

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

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

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

[9] E. Burke, G. Kendall, D. Silva, R. O'Brien, and E. Soubeiga, "An ant algorithm hyperheuristic for the project presentation scheduling problem," in *Proceedings of the Congress on Evolutionary Computation 2005 (CEC'05). Volume 3*, 2005, pp. 2263–2270.

[10] R. Bai and G. Kendall, "An investigation of automated planograms using a simulated annealing based hyper-heuristics," in *Meta-heuristics: Progress as Real Problem Solvers, Selected Papers from the 5th Metaheuristics International Conference (MIC'03)*, T. Ibaraki, K. Nonobe, and M. Yagiura, Eds. Springer, 2005, pp. 87–108.

[11] M. Ayob and G. Kendall, "A monte carlo hyper-heuristic to optimise component placement sequencing for multi head placement machine," in *Proceedings of the International Conference on Intelligent Technologies (InTech'03)*, Chiang Mai, Thailand, December 17–19 2003, pp. 132–141.

[12] G. Kendall and M. Mohamad, "Channel assignment in cellular communication using a great deluge hyper-heuristic," in *Proceedings of the 12th IEEE International Conference on Network (ICON'04)*, vol. 2, Singapore, November 16–19 2004, pp. 769–773.

[13] G. Kendall and M. Mohamad, "Channel assignment optimisation using a hyper-heuristic," in *Proceedings of the 2004 IEEE Conference on Cybernetics and Intelligent Systems (CIS'04)*, Singapore, December 1–3 2004, pp. 790–795.

[14] E. Ozcan, Y. Bykov, M.Birben and E.K. Burke, "Examination Timetabling Using Late Acceptance Hyper-heuristics," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC'09)*, Trondheim, Norway, May 18–21 2009, pp. 997–1004.

[15] E.K. Burke, M. Hyde, and G. Kendall, "Evolving bin packing heuristics with genetic programming," in *Proceedings of the 9th Parallel Problem Solving from Nature (PPSN'IX)*, ser. LNCS, T. Runarsson, H-G.B., E. Burke, J. Merelo-Guervos, L. Whitley, and X. Yao, Eds., vol. 4193. Reykjavik, Iceland: Springer-Verlag, September 9–13 2006, pp. 860–869.

[16] M. Bader-El-Den and R. Poli, "Generating sat local-search heuristics using a gp hyper-heuristic framework." in *Artificial Evolution*, ser. Lecture Notes in Computer Science, N. Monmarch, E.-G. Talbi, P. Collet, M. Schoenauer, and E. Lutton, Eds., vol. 4926. Springer, 2007, pp. 37–49.

[17] A.S. Fukunaga, "Automated discovery of local search heuristics for satisfiability testing," *Evolutionary Computation*, vol. 16, no. 1, pp. 31–61, 2008.

[18] M. Hyde, E. Ozcan, and E.K. Burke, "Multilevel search for evolving the acceptance criteria of a hyper-heuristic," in *Proceedings of the 4th Multidisciplinary International Scheduling Conference: Theory & Applications (MISTA'09)*, Dublin, Ireland, August 10–12 2009, pp. 798–801.

[19] E.K. Burke, M. Hyde, G. Kendall, G. Ochoa, E. Ozcan, and R. Qu, "A survey of hyper-heuristics," Univeristy of Nottingham, CS Technical Report No: NOTTCS-TR-SUB-0906241418-2747, 2009.

[20] E.K Burke, M. Hyde, G. Kendall, G. Ochoa, E. Ozcan, and J.R. Woodward, "A classification of hyper-heuristic approaches," Univeristy of Nottingham, CS Technical Report No: NOTTCS-TR-SUB-0907061259-5808, 2009.

[21] K. Chakhlevitch and P. Cowling, "Choosing the fittest subset of low level heuristics in a hyperheuristic framework," *LNCS*, vol. 3448, pp. 23–33, 2005.

[22] G. Kendall and N.M. Hussin, "An investigation of a tabu-search-based hyper-heuristic for examination timetabling," in *Multidisciplinary scheduling: theory and applications: 1st International Conference, MISTA'03: Nottingham, UK, 13-15 August 2003: selected papers*. Springer Verlag, 2005, p. 309.

[23] G. Kendall and N.M. Hussin, "A tabu search hyper-heuristic approach to the examination timetabling problem at the MARA University of Technology," in *Proceedings of the 5th Practice and Theory of Automated Timetabling (PATAT'04)*, ser. LNCS, vol. 3616. Springer, 2005, pp. 270–293.

[24] L. Han and G. Kendall, "An investigation of a tabu assisted hyper-heuristic genetic algorithm," in *Proceedings of Congress on Evolutionary Computation (CEC'03)*, vol. 3, 2003, pp. 2230–2237.

[25] E. Ozcan, M. Misir, and E.K. Burke, "A self-organising hyper-heuristic framework," in *Proceedings of the 4th Multidisciplinary International Scheduling Conference: Theory & Applications (MISTA'09)*, Dublin, Ireland, August 10–12 2009, pp. 784–787.

[26] M. Misir, K. Verbeeck, G. Vanden Berghe, and P. De Causmaecker, "A hyper-heuristic approach to the home care scheduling problem," KaHo Sint-Lieven, Tech. Rep., 2009.

[27] M. Misir, T. Wauters, K. Verbeeck, and G. Vanden Berghe, "A new learning hyper-heuristic for the traveling tournament problem." in *Proceedings of the 8th Metaheuristic International Conference (MIC'09)*, Hamburg, Germany, 2009.

[28] G. Dueck, "New optimization heuristics: The great deluge algorithm and the record-to-record travel," *Journal of Computational Physics*, vol. 104, no. 1, pp. 86–92, 1993.

[29] S. Nagar, S. Sahni, and E. Shragowitz, "Simulated annealing and combinatorial optimization," in *Proceedings of the 23rd ACM/IEEE Conference on Design Automation*, 1986, pp. 293–299.

[30] F. Li, B. Golden, and E. Wasil, "A record-to-record travel algorithm for solving the heterogeneous fleet vehicle routing problem," *Computers & Operations Research*, vol. 34, no. 9, pp. 2734–2742, 2007.

[31] S.V. Begur, D.M. Miller, and J.R. Weaver, "An integrated spatial DSS for scheduling and routing home-health-care nurses," *Interfaces*, vol. 27, pp. 35–48, 1997.

[32] C. Akjiratikarl, P. Yenradee, and P.R. Drake, "PSO-based algorithm for home care worker scheduling in the UK," *Computers & Industrial Engineering*, vol. 53, no. 4, pp. 559–583, 2007.

[33] S. Bertels and T. Fahle, "A hybrid setup for a hybrid scenario: combining heuristics for the home health care problem," *Computers & Operations Research*, vol. 33, no. 10, pp. 2866–2890, 2006.

[34] E. Cheng and J.L. Rich, "A home health care routing and scheduling problem," Rice University, Texas, Tech. Rep. TR98-04, 1998.

[35] V. Borsani, A. Matta, G. Beschi, and F. Sommaruga, "A home care scheduling model for human resources," in *Proceedings of the International Conference on Service Systems and Service Management (ICSSSM'06)*, Troyes, France, vol. 1, 2006, pp. 449–454.

[36] K. Martin and M. Wright, "Using particle swarm optimization to determine the visit times in community nurse timetabling," in *Proceedings of the 7th International Conference on the Practice and Theory of Automated Timetabling (PATAT'08)*, Montreal, Canada, August 18–22 2009.

[37] P. Eveborn, P. Flisberg, and M. Ronnqvist, "Laps Care–an operational system for staff planning of home care," *European Journal of Operational Research*, vol. 171, no. 3, pp. 962–976, 2006.

[38] P. Eveborn, M. Ronnqvist, H. Einarsdottir, M. Eklund, K. Liden, and M. Almroth, "Operations Research Improves Quality and Efficiency in Home Care," *Interfaces*, vol. 39, no. 1, p. 18, 2009.

[39] O. Braysy, W. Dullaert, and P. Nakari, "The potential of optimization in communal routing problems: case studies from Finland," *Journal of Transport Geography*, vol. 17, no. 6, pp. 484–490, 2009.

[40] S. Nickel, M. Schroder, and J. Steeg, "Planning for home health care services," Fraunhofer-Instituts fur Techno- und Wirtschaftsmathematik, Tech. Rep. 173, 2009.

[41] S. Chahed, E. Marcon, E. Sahin, D. Feillet, and Y. Dallery, "Exploring new operational research opportunities within the Home Care context: the chemotherapy at home," *Health Care Management Science*, vol. 12, no. 2, pp. 179–191, 2009.

[42] M.M. Solomon, "Algorithms for the vehicle routing and scheduling problems with time window constraints," *Operations Research*, vol. 35, no. 2, pp. 254–265, 1987.

[43] T. Justesen and M.S. Rasmussen, "The home care crew scheduling problem,", MSc Thesis, T.U. Denmark & U. Copenhagen, 2008.