# The Effect of the Set of Low-Level Heuristics on the Performance of Selection Hyper-heuristics

M. Mısır[1,2], K. Verbeeck[1,2], P. De Causmaecker[2], and G. Vanden Berghe[1,2]

[1] CODeS, KAHO Sint-Lieven
{mustafa.misir,katja.verbeeck,greet.vandenberghe}@kahosl.be
[2] CODeS, Department of Computer Science, KU Leuven Campus Kortrijk
patrick.decausmaecker@kuleuven-kortrijk.be

**Abstract.** The present study investigates the effect of heuristic sets on the performance of several selection hyper-heuristics. The performance of selection hyper-heuristics is strongly dependant on low-level heuristic sets employed for solving target problems. Therefore, the generality of hyper-heuristics should be examined across various heuristic sets. Unlike the majority of hyper-heuristics research, where the low-level heuristic set is considered given, the present study investigates the influence of the low-level heuristics on the hyper-heuristic's performance. To achieve this, a number of heuristic sets was generated for the patient admission scheduling problem by setting the parameters of a set of parametric heuristics with specific values. These values were set such that nine heuristic sets with different improvement capabilities, speed characteristics and size were generated. A group of hyper-heuristics with certain selection mechanisms and acceptance criteria having dissimilar intensification/diversification abilities were taken from the literature enabling a comprehensive analysis. The experimental results indicated that different hyper-heuristics perform superiorly on distinct heuristic sets. The results can be explained and hence result in hyper-heuristic design recommendations.

**Keywords:** Hyper-heuristics, Heuristic Set, Generality.

## 1 Introduction

*Selection hyper-heuristics* have been studied to effectively manage multiple algorithms, with the motivation behind their employment being to use the heuristics' strengths and eliminating their weaknesses, resulting in a better performance [1]. They take the search process to the heuristic level and perform without problem domain knowledge. Thus, hyper-heuristics are considered general algorithms capable of solving a diverse range of problems. Therefore, most of the hyper-heuristic studies in the literature deal with problem solving [2]. However, selection hyper-heuristics are not concerned with solving some problem instances, but with managing low-level heuristic sets for solving these instances as efficiently

C.A. Coello Coello et al. (Eds.): PPSN 2012, Part II, LNCS 7492, pp. 408–417, 2012.

as possible. There are a limited number of studies concentrating on the heuristic set part with heuristic set reduction or heuristic elimination strategies forming the basis of these studies. In [3], heuristics were made tabu for certain iterations based on their performance. A similar tabu idea was employed for a genetic algorithm based hyper-heuristic in [4]. A heuristic subset selection mechanism after a certain number of iterations and a heuristic set reduction strategy that excludes heuristics in time to determine a good heuristic subset from a large heuristic set were studied in [5]. Another heuristic subset selection approach which temporarily eliminates poor performing heuristics was introduced in [6]. Contrastingly, in [7], different heuristic sets using multiple heuristics from the low-level heuristics for solving the DNA sequencing problem were tested with a suite of hyper-heuristics.

Due to the search level and problem-independent nature of selection hyper-heuristics, *generality* is considered their most important trait. To show their generality level, work on different heuristic sets, with differing features, is required. To the best of our knowledge, there is no study focusing on the effect of different heuristic sets on the performance of hyper-heuristics. The present contribution aims at filling the void mentioned by trying to determine what features should be considered in the design phase of a hyper-heuristic from a generality perspective. For this purpose, 11 low-level heuristics designed to solve the patient admission scheduling problem were used to generate nine heuristic sets. These heuristic sets exhibit differences regarding their improvement capabilities and the speed of the residing heuristics as well as the number of utilised heuristics. Two heuristic selection mechanisms together with seven move acceptance criteria were adopted in building 14 hyper-heuristics with distinct characteristics relying on their selection strategies and intensification/diversification capabilities. The computational results clearly indicated that the nature of the heuristics, distribution of different heuristic types, size of the heuristic sets and runtime limitations have a remarkable impact on the performance of hyper-heuristics.

In the remainder of the paper, the low-level heuristics for the patient admission scheduling problem and heuristics sets generated based on these heuristics are argued in Section 2. Followingly, Section 3 elaborates the tested hyper-heuristics. Next, the computational results are presented and discussed in Section 4. In the last section, the paper is concluded and the requirements for generality and the future research opportunities are presented.

## 2 Patient Admission Scheduling Problem and Heuristics

The present study focuses on patient admission scheduling (PAS) due to its combinatorial complexity as well as the existence of a set of heuristics. The PAS problem concerns assigning patients to hospital rooms or beds based on the patients' requirements [8]. The basic components of the problem are: patients, rooms, wards and time slots. Each patient is characterised by his/her gender, age, pathology, room preference, admission date, and duration of the treatment. It is assumed that every pathology can be linked to one of the hospital's specialisms. Multiple wards of the hospital can have the same specialism, but some

wards are more specialized than others. We distinguish between major and minor specialisms. Every room is located on a ward of the hospital. The specialisms of the ward are inherited to a certain degree by the rooms of the ward. A room is characterized by its properties and its bed capacity. Depending on the patient's pathology, some of the room properties are mandatory or preferable.

### 2.1 Low-Level Heuristics

The following 11 simple low-level heuristics were used in the experiments. For this comparison study, it does not actually matter how this set of low-level heuristics is composed.

- $\boldsymbol{LLH_1}$: Swap all the bed assignments of a randomly selected patient with the beds of randomly selected patients
- $\boldsymbol{LLH_2}$: Transfer all the bed assignments of a randomly selected patient to randomly selected empty beds
- $\boldsymbol{LLH_3}$: Swap all the bed assignments between two randomly selected patients
- $\boldsymbol{LLH_4}$: Swap all the bed assignments of a randomly selected patient with randomly selected occupied beds. Transfer the remaining assignments to the randomly selected beds
- $\boldsymbol{LLH_5}$: Transfer all the bed assignments of a randomly selected patient to a randomly selected empty bed
- $\boldsymbol{LLH_6}$: Swap a randomly selected bed assignment with another bed while respecting room properties
- $\boldsymbol{LLH_7}$: Swap a randomly selected bed assignment with another bed while respecting room preferences of the corresponding patient
- $\boldsymbol{LLH_8}$: Swap a randomly selected bed assignment with another bed while respecting the room specialism
- $\boldsymbol{LLH_9}$: Swap a randomly selected bed assignment of a randomly selected patient with another bed while respecting room properties
- $\boldsymbol{LLH_{10}}$: Swap two randomly selected beds
- $\boldsymbol{LLH_{11}}$: Transfer all the patients in a randomly selected room to another randomly selected room

### 2.2 Differentiating Heuristic Sets

The motivation here is to generate a group of heuristic sets using the aforementioned parametric low-level heuristics for PAS. Nine heuristic sets in different sizes, with different speed and improvement capabilities were generated by setting their parameters. There exist studies concerning heuristics requiring their parameters be set when applying a number of atomic steps [8]. Similarly, in the present research, each heuristic has a parameter called *sampling factor*. This parameter constitutes the number of steps to apply the same operator for different neighbouring solutions. For instance, $LLH_3$ with sampling factor 4 means that it should perform the corresponding swap operation 4 times at each iteration.

**Heuristic Sets.** Nine heuristic sets under three group headings were derived based on the 11 parametric heuristics depicted in Table 1. The first group of heuristic sets was composed of 11 heuristics with a sampling factor of four. The aim of using this first group is to measure the performance of various hyper-heuristics in a default setting. The second set involves 22 heuristics with two versions of each heuristic with sampling factors 4 and 1000. The heuristics with sampling factor 1000 is 250 times slower than the ones with sampling factor 4. It enables investigating how a hyper-heuristic behaves when the speed difference among heuristics is extremely large. The last group employs 44 heuristic with four versions of each heuristic using sampling factor values 1,4,8,16. The reasoning behind this setting is to evaluate hyper-heuristics on larger heuristic sets with relatively small speed differences.

**Table 1.** Heuristic sets used for the experiments

| | **Set size** | **Sampling factors** | **Selection type** |
|---|---|---|---|
| $HS_1$ | 11 | 4 | BEST |
| $HS_2$ | 11 | 4 | FIRST_IMPROVING |
| $HS_3$ | 11 | 4 | HILL_CLIMBER |
| $HS_4$ | 22 | 4, 1000 | BEST |
| $HS_5$ | 22 | 4, 1000 | FIRST_IMPROVING |
| $HS_6$ | 22 | 4, 1000 | HILL_CLIMBER |
| $HS_7$ | 44 | 1,4,8,16 | BEST |
| $HS_8$ | 44 | 1,4,8,16 | FIRST_IMPROVING |
| $HS_9$ | 44 | 1,4,8,16 | HILL_CLIMBER |

Each of these heuristic set groups was tested under three different conditions. The first method, $BEST$, returns the best neighbouring solution after all the sample solutions were visited at each iteration. The second approach, $FIRST\_IMPROVING$, uses the first improving solution found after the sampling operations. The last technique, $HILL\_CLIMBER$, generates hill climbers based on the sampling factor value. Whenever a better or equal quality neighbouring solution is found during the sampling period, it is accepted.

Figure 1 depicts the average speed of performing one move on a PAS instance by each heuristic set. According to this metric, the heuristic sets with 22 heuristics, i.e. $HS_4$, $HS_5$, $HS_6$, are slower than the all others. This slowness is caused by utilising heuristics with a sampling factor of 1000. Of these, $HS_4$ and $HS_6$ are the slowest, as shown in the second graph. This severe speed difference occurs when a heuristic with the sampling factor of 1000 always checks 1000 neighbouring solutions, however, $HS_5$ stops looking for better neighbouring solutions whenever it finds an improving one.

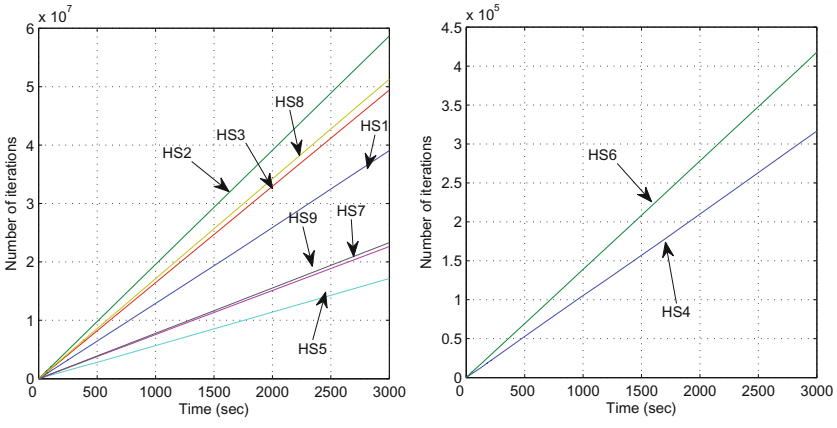


**Fig. 1.** Number of iterations spent over time when heuristics are randomly selected

## 3 Tested Hyper-heuristics

Fourteen selection hyper-heuristics (2 heuristic selection × 7 move acceptance) involving mechanisms from the literature were used for the experiments. Two selection criteria were employed for the heuristic selection process. The first approach is the simple random (SR) heuristic selection mechanism that chooses heuristics in a uniformly random manner. The second approach is the adaptive dynamic heuristic set (ADHS) strategy, determining effective heuristic subsets at runtime [9]. This strategy was also used in the winning hyper-heuristic [10] of the first international Cross-domain Heuristic Search Challenge (CHeSC 2011)[1]. The heuristic subset selection process is carried out using a performance metric involving the most relevant elements to evaluate the online behaviour of the heuristics. The details of the performance metric for heuristic $i$ are shown in Equation 1. $C_{p,best}(i)$ represents the number of new best solutions discovered during a phase. $f_{p,imp}(i)$ shows the total amount of improvement provided during a phase. $f_{p,wrs}(i)$ indicates the total worsening caused during a phase. $f_{imp}(i)$ and $f_{wrs}(i)$ both refer to the same measurements as the last two, but during the whole search rather than a single phase. The remaining elements were used to combine the improvement capabilities of the heuristics with their speed enabling better judgement. $t_{remain}$ denotes the remaining execution time to finalise the whole search process. $t_{p,spent}(i)$ and $t_{spent}(i)$, the former represents the spent execution time during a phase and the latter, from the start. The $w_j$ values are set as weights to differentiate the importance of each individual performance element. It is more important to have a higher value for an earlier element.

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

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

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

After a number of iterations, all the active heuristics are evaluated based on this performance metric. The length of these iterations is denoted as phase length, $pl$. The heuristics with comparatively poorer performance are excluded from the heuristic set for a number of phases. The duration of exclusion is referred to as tabu duration ($d$) and is set to $d = \sqrt{2n}$, where $n$ refers to the number of heuristics in the heuristic set. The tabu duration of the consecutively excluded heuristics is increased by one, until $2\sqrt{2n}$. If such a heuristic survives after a phase, its tabu duration is set to its initial value. In addition, the phase length is adapted during runtime with respect to the speed of the heuristics in the current heuristic set.

The heuristic selection operation from these subsets is handled using a learning automaton [9,11]. This method accommodates a vector of heuristic selection probabilities. These values are reset at the end of each phase given the synchronously performed update operation in determining which heuristics will be excluded.

The employed move acceptance strategies are as follows: adaptive iteration limited list-based threshold accepting (AILLA) [9], great deluge (GD) [8], late acceptance (LATE) [12], simulated annealing (SA) [8], improving or equal (IE), only improving (OI) and all moves (AM). All of these acceptance mechanisms immediately accept improving solutions. The first four acceptance methods, AILLA, GD, LATE and SA, provide diversification mechanisms by accepting worsening solutions with respect to certain dynamic threshold values. IE accepts equal quality solutions to diversify the search process. OI accepts only better quality solutions, hence it has no diversification strategy. The last acceptance criterion, AM, accepts all visited solutions.

The resulting hyper-heuristics using all these sub-mechanisms, ADHS-AILLA, ADHS-GD, ADHS-SA, ADHS-LATE, ADHS-IE, ADHS-OI, ADHS-AM, SR-AILLA, SR-GD, SR-SA, SR-LATE, SR-IE, SR-OI, SR-AM, have distinct characteristics for selecting heuristics and diversifying the search process such that a comprehensive performance analysis can be performed.

## 4 Computational Results

14 hyper-heuristics were run 10 times on 7 PAS instances, $dataset0 \rightarrow dataset6$, attainable at `http://allserv.kahosl.be/~peter/pas/` using Pentium Core 2 Duo 3 GHz PCs with 3.23 GB memory. Each hyper-heuristic was tested on 9 different heuristic sets. The time limits were taken as 10 and 50 minutes.

In Figure 2, the significantly best hyper-heuristics for each heuristic set are listed for the 10 minutes and 50 minutes experiments respectively. The significance of the performance difference is evaluated using the Wilcoxon test with a 95% confidence interval.

Regarding the 10 minutes experiments, different acceptance strategies deliver similar performances after 10 minutes of execution in most of the cases. For specific heuristic sets, even very simple acceptance mechanisms like OI and AM can find similar results to those in $HS_3$ and $HS_6$. The explanation behind this is as follows: the diversification characteristics of the selection mechanisms are no longer useful due to the heuristics' hill climbing behaviour. In addition, there is no heuristic set for which one hyper-heuristic outperforms the others except SR-LATE on $HS_9$. Moreover, the hyper-heuristics involving an acceptance mechanism with diversification and ADHS perform poorly on $HS_9$. For the majority of the test cases, AILLA and LATE perform better, yet there is no general statistically significant performance difference. For 50 minute experiments, the hyper-heuristics with GD perform best together with different hyper-heuristics on different heuristic sets. This can be considered as an effect of the execution time limit increase, from 10 minutes to 50 minutes. The hyper-heuristics using AILLA and LATE also show effective performance after running them for 50 minutes.

The hyper-heuristics generated the best results on $HS_1$, $HS_2$, $HS_7$ and $HS_8$. The heuristic sets involve heuristics using low sampling factors with a selection type of either $BEST$ or $FIRST\_IMPROVING$. This means that the fast heuristics with well balanced intensification-diversification behaviour resulted in better performance on the tested problem instances.

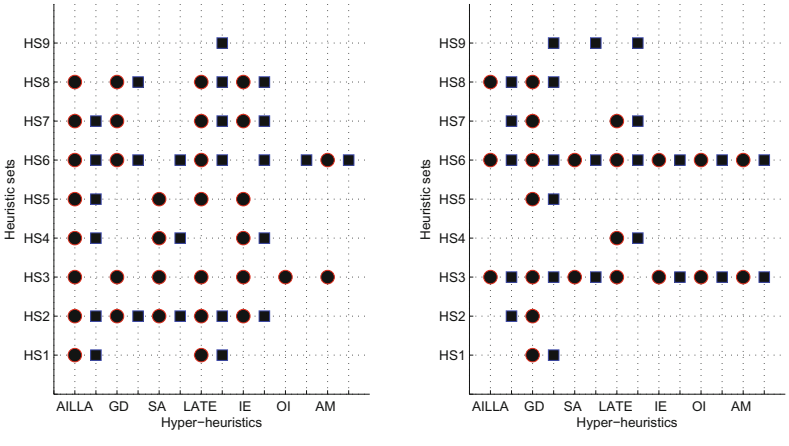


**Fig. 2.** The significantly best hyper-heuristics on each heuristic set after 10 minutes (left) and 50 minutes (right) (Circles refer to the hyper-heuristics with ADHS and squares refer to the hyper-heuristics with SR)

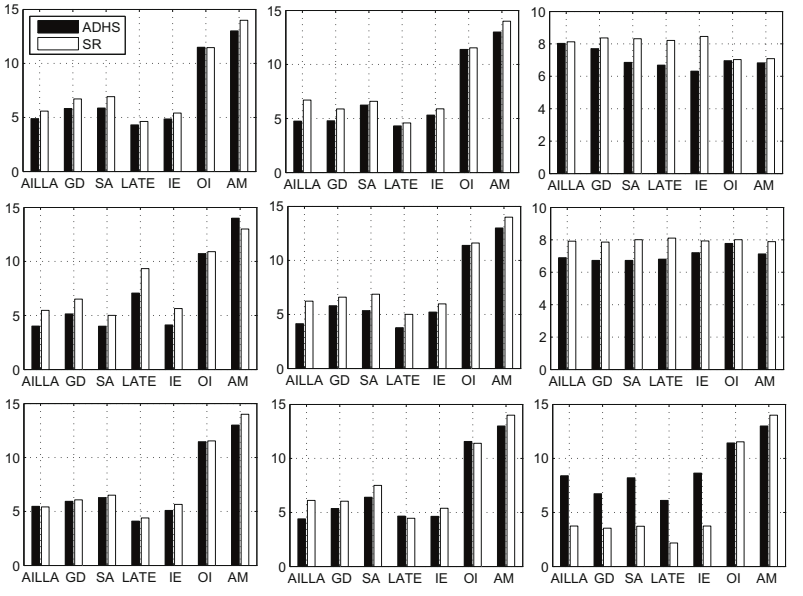


**Fig. 3.** Average ranking of the hyper-heuristics after 10 minutes (Each graph represents the results obtained on a heuristic set. They are ordered from left to right, top to bottom: $HS_1 \rightarrow HS_9$).

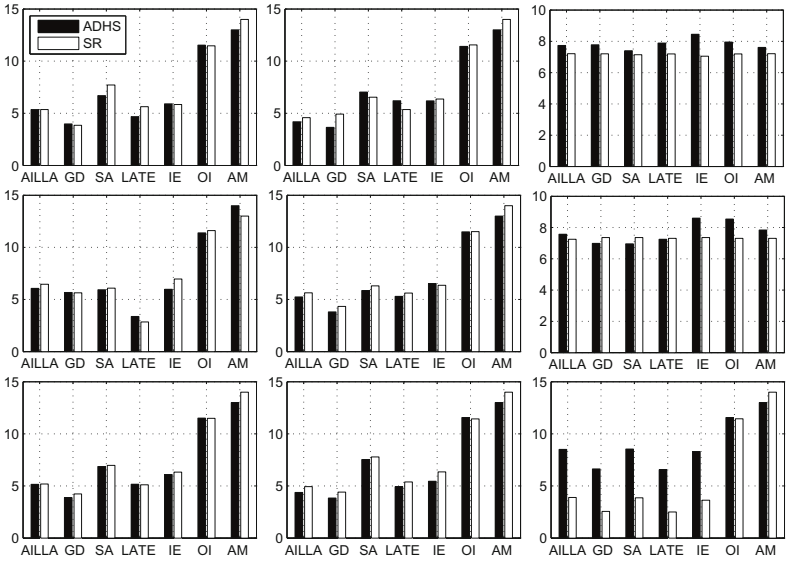


**Fig. 4.** Average ranking of the hyper-heuristics after 50 minutes (Each graph represents the results obtained on a heuristic set. They are ordered from left to right, top to bottom: $HS_1 \rightarrow HS_9$).

In Figure 3 and 4, the average ranking of the hyper-heuristics on each heuristic set with 10 and 50 minutes execution time limits are presented. For the 10 minute experiments, ADHS performs better than SR in the majority of the cases. However, SR provides better performance than ADHS for $HS_9$ that accommodates hill climbers, since ADHS mostly excludes the heuristics with a sampling factor of 1 that help to diversify the search. Consequently, the remaining heuristics may not be able to escape from certain local optima. For the 50 minute experiments, the performance difference between ADHS and SR degraded when compared to the 10 minute case. The two main reasons behind this empirical result are the low evolvability characteristic of the solution space and the longer running time. In this case, choosing wrong heuristics is not as influential when compared with the 10 minute execution time experiments.

## 5 Conclusion

The present study examined the performance changes of 14 selection hyper-heuristics due to different heuristic sets for the patient admission scheduling problem. The nature of the heuristic sets, size of the heuristic sets and other related limitations are all potential reasons why one hyper-heuristic delivers superior results. We reviewed these conditions in various experiments to identify the hyper-heuristics' generality levels. Nine heuristic sets were utilised in demonstrating the effect of the heuristic sets on the performance of selection hyper-heuristics. Each of these sets exhibits differences depending on the aforementioned experimental conditions. We tested, using these heuristics sets, 14 hyper-heuristics composed of an adaptive and a random selection mechanism combined with 7 move acceptance methods from the literature. The computational results on the tested heuristic sets showed that the best hyper-heuristic can change based on the heuristic set used and execution time limits employed. Particularly fast heuristic sets involving certain degree of intensification and diversification features showed better performance. These results also indicated that some of the hyper-heuristic components are more valuable than others under certain conditions. If the gap between speed and improvement capabilities of the low-level heuristics is large and the allowed execution time is short, then the heuristic selection is more important. However, if the heuristics are highly perturbative and destructive then a move acceptance strategy with effective diversification capabilities is a vital requirement. Also, a very naive acceptance mechanism like AM can deliver comparable results if the low-level heuristics have effective improvement capabilities. We have then demonstrated how by addressing different generality requirements.

In future work, the diversity of the application domains will be extended to show the performance changes with respect to the relation between heuristic search space and solution space. Additional mechanisms will be investigated to enable more general hyper-heuristics compared to traditional approaches composed of selection-acceptance pairs. Finally, a method will be devised to predict or measure the generality level of a hyper-heuristic.

## References


1. Burke, E., Hart, E., Kendall, G., Newall, J., Ross, P., Schulenburg, S.: Hyper-Heuristics: An Emerging Direction in Modern Search Technology. In: Handbook of Meta-Heuristics, pp. 457–474. Kluwer Academic Publishers (2003)
2. Burke, E., Gendreau, M., Hyde, M., Kendall, G., Ochoa, G., Ozcan, E., Qu, R.: Hyper-heuristics: A survey of the state of the art. Journal of the Operational Research Society (to appear)
3. Burke, E., Kendall, G., Soubeiga, E.: A tabu-search hyper-heuristic for timetabling and rostering. Journal of Heuristics 9(3), 451–470 (2003)
4. Han, L., Kendall, G.: An investigation of a tabu assisted hyper-heuristic genetic algorithm. In: Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2003, vol. 3, pp. 2230–2237 (2003)
5. Chakhlevitch, K., Cowling, P.I.: Choosing the Fittest Subset of Low Level Heuristics in a Hyperheuristic Framework. In: Raidl, G.R., Gottlieb, J. (eds.) EvoCOP 2005. LNCS, vol. 3448, pp. 23–33. Springer, Heidelberg (2005)
6. Misir, M., Verbeeck, K., De Causmaecker, P., Vanden Berghe, G.: Hyper-heuristics with a dynamic heuristic set for the home care scheduling problem. In: Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2010, Barcelona, Spain, July 18-23, pp. 2875–2882 (2010)
7. Blazewicz, J., Burke, E., Kendall, G., Mruczkiewicz, W., Oguz, C., Swiercz, A.: A hyper-heuristic approach to sequencing by hybridization of DNA sequences. Annals of Operations Research, 1–15 (2011)
8. Bilgin, B., Demeester, P., Misir, M., Vancroonenburg, W., Vanden Berghe, G.: One hyperheuristic approach to two timetabling problems in health care. Journal of Heuristics 18(3), 401–434 (2012)
9. Misir, M., Verbeeck, K., De Causmaecker, P., Vanden Berghe, G.: A New Hyper-heuristic Implementation in HyFlex: a Study on Generality. In: Fowler, J., Kendall, G., McCollum, B. (eds.) Proceedings of the 5th Multidisciplinary International Scheduling Conference: Theory & Applications, MISTA 2011, Phoenix/Arizona, USA, August 10-12, pp. 374–393 (2011)
10. Misir, M., Verbeeck, K., De Causmaecker, P., Vanden Berghe, G.: An intelligent hyper-heuristic framework for CHeSC 2011. In: The 6th Learning and Intelligent Optimization Conference (LION 6). LNCS (to appear, 2012)
11. Misir, M., Wauters, T., Verbeeck, K., Vanden Berghe, G.: A hyper-heuristic with learning automata for the traveling tournament problem. In: The 8th Metaheuristics International Conference on Metaheuristics: Intelligent Decision Making, Post Conference Volume (2011)
12. Demeester, P., De Causmaecker, P., Vanden Berghe, G.: 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 2010), Belfast, Northern Ireland, August 10-13 (2010)