An Investigation of Selection Hyper-heuristics in Dynamic Environments

Berna Kiraz, A. ima Uyar, Ender Özcan

Abstract. Hyper-heuristics are high level methodologies that perform search over the space of heuristics rather than solutions for solving computationally difficult problems. A selection hyper-heuristic framework provides means to exploit the strength of multiple low level heuristics where each heuristic can be useful at different stages of the search. In this study, the behavior of a range of selection hyper-heuristics is investigated in dynamic environments. The results show that hyper-heuristics embedding learning heuristic selection methods are sufficiently adaptive and can respond to different types of changes in a dynamic environment.

1 Introduction

Many real world optimization problems change over time, i.e. they are dynamic. In a dynamic environment, a change may occur in the objective function, the constraints of the problem or the problem instance itself. Moreover, the characteristics of the change dynamics may be different, e.g. the environment may be changing quickly or slowly, the magnitude of the change may be very high or low, and/or there may be a pattern in the change. A *good* optimization method solving a problem in a dynamic environment should be capable of adapting itself to those changes, modifying the candidate solution(s) to track the changing optima as quickly and as closely as possible. This necessitates setting up a strategy that specifies how the method should react to the environmental changes.

A hyper-heuristic is a high-level methodology which *selects* or *generates* low-level heuristics to solve difficult problems [5, 7]. In a selection hyper-heuristic framework, a hyper-heuristic selects a low-level heuristic without using any problem domain specific information and applies it to the solution at hand [18]. The new solution is either accepted or rejected based on an acceptance criterion. This process, based on a single point search, continues iteratively until a stopping condition is met. The heuristic selection and the acceptance methods are the two key components of selection hyper-heuristics. The idea of choosing from heuristics (or neighbourhoods) dates back to the 1960s [10, 12]. Denzinger et al. [11] introduced the term *hyper-heuristic* for the first time. There is a growing interest in hyper-heuristic research.

Cowling et al [9] investigated the performance of a variety of heuristic selection methods over a timetabling problem. The simple heuristic selection methodologies include *Simple Random* (SR), which chooses a low-level heuristic at random and then applies it to the candidate solution and *Greedy* (GR), which applies all low level heuristics to the same solution separately and then selects the one producing the best result. A more sophisticated learning heuristic selection

method was also proposed in this study. The *Choice Function* (CF) heuristic selection mechanism scores each low level heuristic based on its individual performance, its collective performance considering the previous low level heuristic invocation and the elapsed time since it was last called. A low level heuristic with the maximum score is selected at each step and its statistics are updated.

Reinforcement learning (RI) can also be used as a heuristic selection mechanism [6, 20]. In Reinforcement learning, each heuristic has a score. Initial scores of each heuristic are the same. If the current heuristic produces an improved solution, its score is increased; otherwise it is decreased. The scores are allowed to vary within predetermined lower and upper bounds.

Move acceptance strategies can be deterministic or non-deterministic. *All Moves* (AM) accepted, *Only Improving* (OI) accepted, *Improving and Equal* (IE) accepted are some examples for the deterministic acceptance criteria in literature [6, 9]. There are other more sophisticated acceptance mechanisms that were investigated as part of hyper-heuristics, such as Monte Carlo, Simulated Annealing and Great Deluge acceptance methods [1, 16, 4]. More on hyper-heuristics can be found in [6, 3, 21].

A preliminary study on the applicability of hyper-heuristics in a dynamic environment was conducted by Ozcan et al. [19]. A Greedy hyper-heuristic was used in the experiments. The results show that hyper-heuristics are indeed appropriate for solving dynamic environment problems. This is not surprising considering the adaptive nature of hyper-heuristics. This study extends the previous one with the goal of comparing the performances of hyper-heuristics using different heuristic selection mechanisms controlling a set of mutational low level heuristics in a dynamic environment. The Moving Peaks Benchmark, which allows full control over all change dynamics, is used in the experiments. The remainder of this paper is organized as follows. The next section provides background on dynamic environments. Section 3 gives the experimental design and results of the computational experiments for comparing the performance of hyper-heuristics to solve dynamic environment problems based using the Moving Peaks Benchmark, and Section 4 concludes the paper.

2 Dynamic Environments

In a dynamic environment different problem components, such as the objectives or the constraints, may change in time, restructuring the search landscape of a given problem. The problem solving methodologies should adaptively react to these changes and track the moving optima quickly and closely. Different change characteristics generate different dynamic environments with different requirements. These characteristics can be categorized as follows [2]:

- *Frequency of change* defines how often the environment changes.
- *Severity of change* defines the magnitude of the change in the environment.
- *Predictability of change* is a measure of the correlation between changes.
- *Cycle length/cycle accuracy* is a property that defines whether the optima return to previous locations.

Different techniques should be used for environments exhibiting different change characteristics. The simplest approach is to restart the search algorithm each time a change occurs. However, usually the change in the environment is not too drastic and information gained during the previous environments can be used to locate the new optima much quicker. The main problem in this case is the issue of convergence. Many evolutionary approaches, which take these into consideration, have been developed for dynamic environments. These are grouped into four main categories in [15] as follows: (i) Approaches which increase diversity after a change, (ii) approaches that maintain diversity throughout the running, (iii) memory based approaches, (iv) multipopulation approaches.

In the first group, during the stationary period, the evolutionary algorithm is applied normally. However, whenever a change occurs in the environment, a mechanism is used to increase diversity. Hypermutation [8] is the most common approach that belongs to this category, where the mutation rate is increased drastically for a number of generations when a change in the environment is detected. In [24], the mutation rate is increased gradually after a change occurs. In these approaches, the main problem is to determine the amount of required diversity. This amount depends highly on the magnitude of the change in the environment. Generating too much diversity will disrupt the search process, while too little will not be sufficient to solve the problem of convergence. The approaches in this category are generally more useful when the magnitude of the change is not too high.

In the second group of approaches, convergence is always prevented by maintaining diversity all throughout the generations. One of the well known methods in this category is the random immigrants approach [13]. In this method, a number of randomly generated individuals are inserted into the current generation. Maintaining a high level of diversity all the time may effect the search process negatively during the stationary periods by preventing convergence. The approaches in this category are generally more useful when the magnitude of the change is relatively high but the change frequency is relatively low.

In the third group of approaches, the evolutionary algorithm uses a memory to remember solutions which have been successful in previous environments. This memory is implemented explicitly or implicitly. In approaches which use an explicit memory, e.g. as in [27, 25], useful individuals are stored separately to be used later on. In approaches which use an implicit memory, e.g. as in [14, 23], memory is implemented by using redundant representations. The approaches in this category are generally more useful when previous environments are encountered later during the search.

In the fourth group of approaches, the population is divided into subpopulations, each of which conduct search in different regions of the search space. Among the well known approaches in this group are the self-organizing scouts [2] and the multi-national GA [22]. The approaches in this category aim to track several optima in different parts of the search space.

Detailed information on dynamic environments can be found in [2, 17] and more recently in [26].

3 Computational Experiments

3.1 Experimental Design

In this study, we use a parametrized Gaussian mutation to create different low-level heuristics. In Gaussian mutation, random values drawn from a Gaussian distribution with given mean and standard deviations, are added to each element of a candidate solution to generate a new one. Mean value for the Gaussian distribution in all low-level heuristics is taken as zero. The standard deviations are chosen as: 0.5, 2, 7, 15, 20, 25, and 30. These are determined experimentally.

As heuristic selection methods, Simple Random (SR), Greedy heuristic selection (GR), Choice Function (CF), and Reinforcement Learning (RI) are chosen. These selection heuristics are all representatives of different approaches. The first one is random and uses no information. The second one is greedy, i.e. tries to select the best option at each step. The last two incorporate some form of adaptation and learning. With each heuristic selection mechanism, Improving and Equal move (IE) acceptance scheme is used.

To observe the effectiveness of our hyper-heuristic, we include a hypermutation based single point search method (HM) in our experiments. In this method, a Gaussian mutation with zero mean and a predetermined standard deviation is applied during the stationary periods. Whenever the environment changes, the standard deviation is increased for a number of consecutive iterations. The parameter settings are again determined experimentally. The standard deviation of the Gaussian mutation during the stationary periods is 2. When change occurs, this is increased to 7 for 70 consecutive fitness evaluations, after which it is reset to 2.

For the experiments, we use the Moving Peaks Benchmark (MPB), which is a multidimensional dynamic landscape generator. In this benchmark, the height, width and location of the peaks in the landscape can be altered in a controlled fashion [2], where each peak has time varying height, width and location parameters. The height, the width and the location of each peak are randomly initialized.

Table 1 lists the parameters of the MPB used in the experiments. These are also experimentally determined.

Table 1. Parameter settings for the MPB

Parameter	Setting	Parameter	Setting
Number of peaks p	5	Number of dimensions d	5
Peak heights	$\in [30, 70]$	Correlation coefficient λ	0
Peak widths	$\in [0.8, 7.0]$	Basis function	none
Change severity $vlength$	1.0/5.0/10.0	Peak function	cone
Height_severity	0.0	Minimum coordinates	0.0
Width_severity	0.0	Maximum coordinates	100.0

In this study, we focus on exploring the effects of two change characteristics on the performance of the chosen hyper-heuristics: frequency of the changes and severity of the changes. To determine the duration of the stationary periods between the changes for various change frequency settings, we take the SR heuristic selection as basis. We let the hyper-heuristic using SR and IE run for long periods without any changes in the environment. By looking at the average convergence plots for these runs, we determine the change periods[1] as 6006 fitness evaluations for low frequency (LF), 1001 for medium frequency (MF) and 126 for high frequency (HF). 6006 fitness evaluations correspond to a stage in the plot where the algorithm has been converged for some time, 1001 corresponds to a time where the approach has not yet fully converged and 126 is very early on in the search. In this study, we only change the positions of the peaks and keep their width and length parameters fixed. The step size parameter determines how far the peaks can move at each change, thus it determines the change severity in the environment. In the experiments, the values of this parameter are determined empirically as 1.0, 5.0, and 10.0, for low severity (LS), medium severity (MS),and high severity (HS), respectively.

In order to compare the performance of the algorithms, the results are reported in terms of offline error [2], which is calculated as the cumulative average of the differences between the best values found so far and the optimum value at each time step, as given below.

$$\frac{1}{T}\sum_{t=1}^{T}(opt_t - e_t^*) \tag{1}$$

$$e_t^* = \max\{e_\tau, e_{\tau+1}, \ldots, e_t\} \tag{2}$$

where T is the total number of evaluations and τ is the last time step ($\tau < t$) when change occurred.

For RI, the initial scores of each heuristic are set to 15. Their lower and upper bounds are set to 0 and 30, respectively [20]. If the current heuristic produces a better solution than the previous one, its score is increased by 1, otherwise it is decreased by 1. For CF, α, β, and δ are set to 0.5 and updated by ± 0.01 at each iteration. Whenever the environment is changed, the parameters of CF and RI are reset to their initial values.

3.2 Results and Discussion

All results are presented as the average offline error of 100 runs. For each run of the algorithms, 20 changes occur after the initial environment. Each hyper-heuristic is denoted by their heuristic selection and move acceptance components as in Greedy-IE, where this hyper-heuristic is the greedy heuristic selection method combined with improving and equal as the move acceptance method.

[1] Since we have 7 low level heuristics and the greedy heuristic selection method evaluates all at each step, these values are determined as multiples of 7 to give each method an equal number of evaluations during each stationary period.

The total number of evaluations between each consecutive change is kept the same for all approaches for a fair comparison.

Table 2. Offline error of each approach for each dynamic environment type which is determined by a given frequency and severity of change

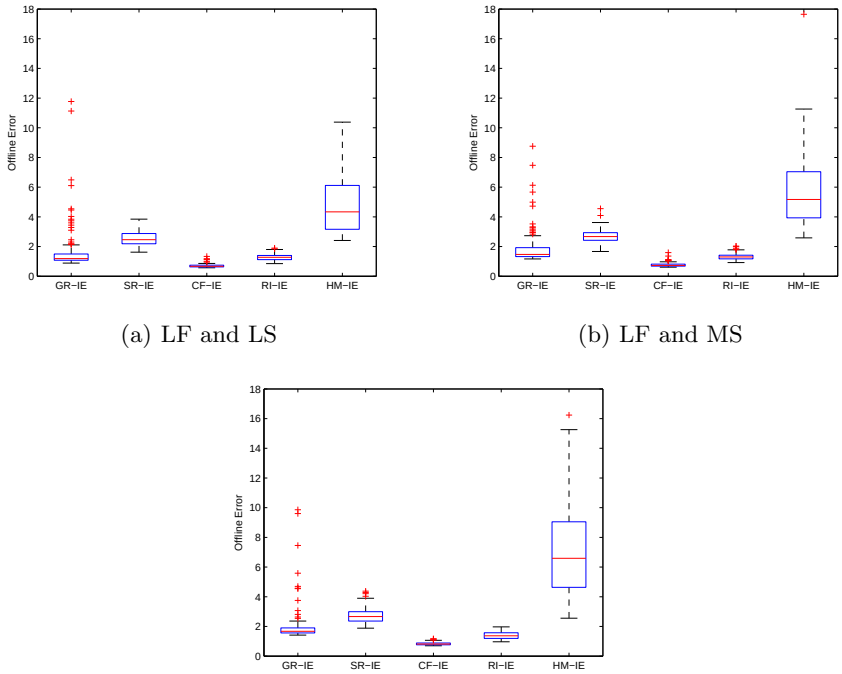
Algorithm	LF			MF			HF		
	LS	MS	HS	LS	MS	HS	LS	MS	HS
GR-IE	1.793	1.964	2.099	5.707	7.733	8.834	21.764	24.887	31.981
SR-IE	2.539	2.7	2.727	11.477	11.963	12.57	44.959	46.065	46.701
CF-IE	**0.706**	**0.767**	**0.838**	**1.669**	**2.062**	**2.399**	**8.793**	**10.091**	**13.894**
RL-IE	1.279	1.317	1.402	3.133	3.266	3.666	9.253	10.747	14.299
HM-IE	4.782	5.716	7.083	13.908	17.047	20.96	32.724	34.883	40.596

Table 2 summarizes the experimental results. The comparison of different approaches in nine different dynamic environments generated by combinations of three change periods and three change severities are given based on the offline error box plots in Figure 1(a)–(c), 2(a)–(c), and 3(a)–(c). In general, while the change frequency is increased, the performance of all methods degrade. Moreover, when the change severity is increased, the offline error is also increased, particularly even more for the high change frequency. It is observed that the choice function-IE hyper-heuristic outperforms the other approaches.

One-way ANOVA and Tukey HSD tests are performed to observe whether the pairwise performance variations between the approaches are statistically significant or not. The corresponding results are provided in Table 3. Greedy-IE performs significantly better than Simple random-IE and Hypermutation-IE for all different change frequencies and change severities. Choice function-IE and Reinforcement learning-IE are always significantly better than the other hyper-heuristics for all cases. Moreover, Hypermutation-IE is almost always significantly worse than the other hyper-heuristics. For the dynamic environment problems where the change frequency is high, Simple random-IE performs significantly better than Hypermutation-IE. Even if this is the case, Simple random-IE does not require any parameter tuning.

4 Conclusion and Future Work

In this study, we compare the performance of five hyper-heuristics combining four previously proposed heuristic selection mechanisms and a hyper-mutation based method with IE managing a set of mutational heuristics. The results show that the heuristic selection mechanism with learning, namely, choice function and reinforcement learning, outperform all other methods based on the offline error, when used within hyper-heuristics in dynamic environments generated using the Moving Peaks Benchmark.



(a) LF and LS

(b) LF and MS

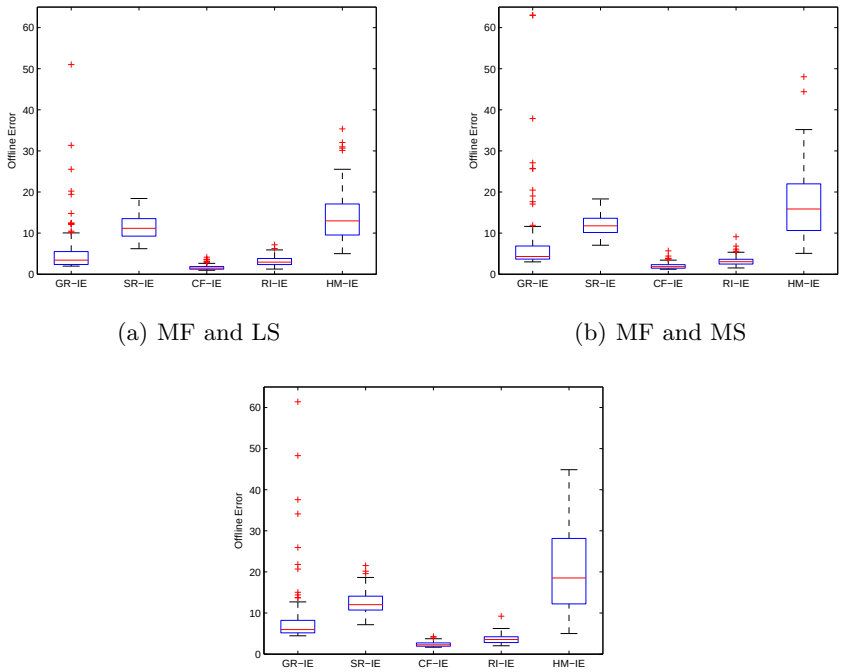
(c) LF and HS

Fig. 1. Comparison of hyper-heuristics combining greedy, simple random, choice function, reinforcement learning and hyper-mutation heuristic selection with IE for different change severity and low frequency settings based on the offline error box plots.

We will further our experiments to include more heuristic selection and acceptance methods, and so we will analyse the behavior of different hyper-heuristics in dynamic environments. Additionally, the performance of hyper-heuristics will be compared to the previously proposed methods and state-of-the-art algorithms for dynamic environments. In this study, it is assumed that the hyper-heuristics are aware of the time when the environment change occurs and acts on this. We are planning to work on a novel approach that does not require this.

References

1. Ayob, M., Kendall, G.: A monte carlo hyper-heuristic to optimise component placement sequencing for multi head placement machine. In: Proceedings of the Int. Conf. on Intelligent Technologies. pp. 132–141 (2003)
2. Branke, J.: Evolutionary optimization in dynamic environments. Kluwer (2002)
3. Burke, E., Hart, E., Kendall, G., Newall, J., Ross, P., Schulenburg, S.: Hyper-heuristics: An emerging direction in modern search technology. In: Glover, F., Kochenberger, G. (eds.) Handbook of Metaheuristics, pp. 457–474. Kluwer (2003)



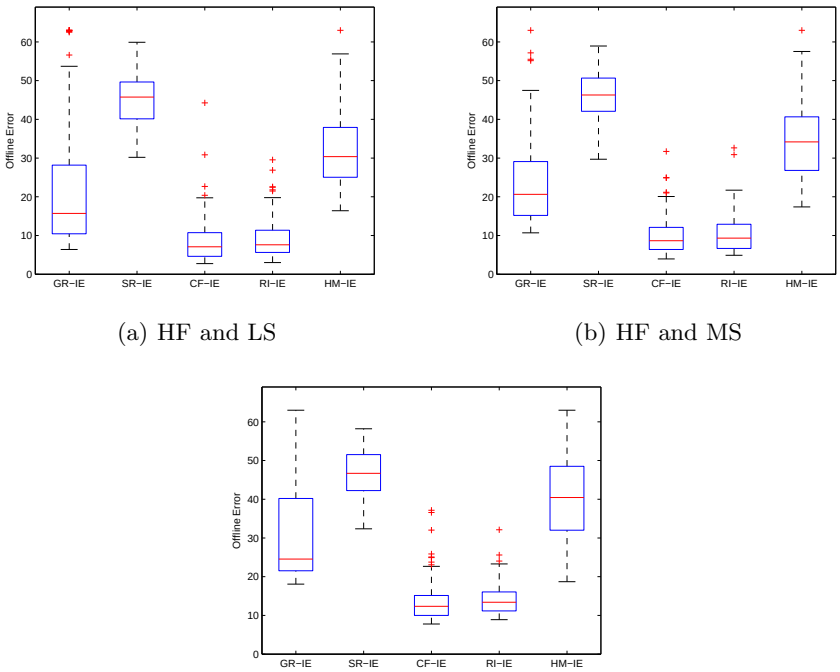
(a) MF and LS

(b) MF and MS

(c) MF and HS

Fig. 2. Comparison of hyper-heuristics combining greedy, simple random, choice function, reinforcement learning and hyper-mutation heuristic selection with IE for different change severity and medium frequency settings based on the offline error box plots.

4. Burke, E., Kendall, G., Misir, M., Özcan, E.: Monte carlo hyper-heuristics for examination timetabling. Annals of Operations Research pp. 1–18 (2010)
5. Burke, E.K., Hyde, M.R., Kendall, G., Ochoa, G., Özcan, E., Woodward, J.R.: Exploring hyper-heuristic methodologies with genetic programming. In: Kacprzyk, J., Jain, L.C., Mumford, C.L., Jain, L.C. (eds.) Computational Intelligence, Intelligent Systems Reference Library, vol. 1, pp. 177–201. Springer (2009)
6. Burke, E.K., Hyde, M., Kendall, G., Ochoa, G., Ozcan, E., Qu, R.: A survey of hyper-heuristics. Tech. rep. (2009)
7. Burke, E.K., Hyde, M., Kendall, G., Ochoa, G., Özcan, E., Woodward, J.R.: A classification of hyper-heuristic approaches. In: Gendreau, M., Potvin, J.Y. (eds.) Handbook of Metaheuristics, International Series in Operations Research and Management Science, vol. 146, pp. 449–468. Springer (2010)
8. Cobb, H.G.: An investigation into the use of hypermutation as an adaptive operator in genetic algorithms having continuous, time-dependent nonstationary environments. Tech. Rep. AIC-90-001, Naval Research Lab., Washington, DC
9. Cowling, P., Kendall, G., Soubeiga, E.: A hyper-heuristic approach to scheduling a sales summit. In: Practice and Theory of Automated Timetabling III : Third International Conference, PATAT 2000. vol. LNCS 2079 (2000)



(a) HF and LS

(b) HF and MS

(c) HF and HS

Fig. 3. Comparison of hyper-heuristics combining greedy, simple random, choice function, reinforcement learning and hyper-mutation heuristic selection with IE for different change severity and high frequency settings based on the offline error box plots.

10. Crowston, W.B., Glover, F., Thompson, G.L., Trawick, J.D.: Probabilistic and parametric learning combinations of local job shop scheduling rules. ONR Research memorandum, GSIA, Carnegie Mellon University, Pittsburgh -(117) (1963)
11. Denzinger, J., Fuchs, M., Fuchs, M.: High performance ATP systems by combining several AI methods. In: 4th Asia-Pacific Conf. on SEAL. pp. 102–107 (1997)
12. Fisher, H., Thompson, G.L.: Probabilistic learning combinations of local job-shop scheduling rules. In: Muth, J.F., Thompson, G.L. (eds.) Industrial Scheduling. pp. 225–251. Prentice-Hall, New Jersey (1963)
13. Grefenstette, J.J.: Genetic algorithms for changing environments. In: Proceedings of Parallel Problem Solving from Nature. p. 1371446 (1992)
14. J. Lewis, E.H., Ritchie, G.: A comparison of dominance mechanisms and simple mutation on nonstationary problems. In: Proceedings of Parallel Problem Solving from Nature. p. 139148 (1998)
15. Jin, Y., Branke, J.: Evolutionary optimization in uncertain environments-a survey. IEEE Tran. on Evolutionary Comp. 9(3), 303 – 317 (2005)
16. Kendall, G., Mohamad, M.: Channel assignment in cellular communication using a great deluge hyperheuristic. In: IEEE Int. Conf. on Network. pp. 769–773 (2004)

Table 3. Pair-wise comparison of hyper-heuristics for each dynamic environment type determined by a given frequency and severity of change. Given A vs B, $s+$ ($s-$) denote that A (B) is performing statistically better than B (A), while $\approx$ denotes that there is no statistically significant performance variation between A and B.

Algorithms	LF			MF			HF		
	LS	MS	HS	LS	MS	HS	LS	MS	HS
GR-IE vs SR-IE	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$
GR-IE vs CF-IE	$s-$	$s-$	$s-$	$s-$	$s-$	$s-$	$s-$	$s-$	$s-$
GR-IE vs RI-IE	$s-$	$s-$	$s-$	$s-$	$s-$	$s-$	$s-$	$s-$	$s-$
GR-IE vs HM-IE	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$
SR-IE vs CF-IE	$s-$	$s-$	$s-$	$s-$	$s-$	$s-$	$s-$	$s-$	$s-$
SR-IE vs RI-IE	$s-$	$s-$	$s-$	$s-$	$s-$	$s-$	$s-$	$s-$	$s-$
SR-IE vs HM-IE	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s-$	$s-$	$s-$
CF-IE vs RI-IE	$s+$	$s+$	$\approx$	$\approx$	$\approx$	$\approx$	$\approx$	$\approx$	$\approx$
CF-IE vs HM-IE	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$
RI-IE vs HM-IE	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$

17. Morrison, R.W.: Designing evolutionary algorithms for dynamic environments. Springer (2004)
18. Özcan, E., Bilgin, B., Korkmaz, E.E.: A comprehensive analysis of hyper-heuristics. Intelligent Data Analysis 12, 3–23 (2008)
19. Özcan, E., Etaner-Uyar, S., Burke, E.: A greedy hyper-heuristic in dynamic environments. In: GECCO 2009 Workshop on Automated Heuristic Design: Crossing the Chasm for Search Methods. pp. 2201–2204 (2009)
20. Özcan, E., Misir, M., Ochoa, G., Burke, E.K.: A reinforcement learning - great-deluge hyper-heuristic for examination timetabling. International Journal of Applied Metaheuristic Computing 1(1), 39–59 (2010)
21. Ross, P.: Hyper-heuristics. In: Burke, E.K., Kendall, G. (eds.) Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques, chap. 17, pp. 529–556. Springer (2005)
22. Ursem, R.K.: Multinational GA optimization techniques in dynamic environments. In: Proceedings of the Genetic Evol. Comput. Conf. pp. 19–26 (2000)
23. Uyar, A.S., Harmanci, A.E.: A new population based adaptive domination change mechanism for diploid genetic algorithms in dynamic environments. Soft Computing 9, 803–814 (2005)
24. Vavak, F., Jukes, K., Fogarty, T.C.: Adaptive combustion balancing in multiple burner boiler using a genetic algorithm with variable range of local search. In: Proceedings of the Int. Conf. on Genetic Algorithms. pp. 719–726 (1997)
25. Yang, S.: Genetic algorithms with memory and elitism based immigrants in dynamic environments. Evolutionary Computation 16, 385–416 (2008)
26. Yang, S., Ong, Y.S., Jin, Y. (eds.): Evolutionary Computation in Dynamic and Uncertain Environments, Studies in Computational Int., vol. 51. Springer (2007)
27. Yang, S., Yao, X.: Population-based incremental learning with associative memory for dynamic environments. Trans. on Evolutionary Comp. 12, 542–561 (2008)