Performance of Selection Hyper-heuristics on the Extended HyFlex Domains

Alhanof Almutairi[1], Ender Özcan[1], Ahmed Kheiri[2] and Warren G. Jackson[1]

[1] University of Nottingham, ASAP Research Group, School of Computer Science, Wollaton Road, Nottingham, NG8 1BB, UK,
{psxaka,ender.ozcan,psxwgj}@nottingham.ac.uk
[2] Cardiff University, Operational Research Group, School of Mathematics, Senghennydd Road, Cardiff, CF24 4AG, UK,
KheiriA@cardiff.ac.uk

Abstract. Selection hyper-heuristics perform search over the space of heuristics by mixing and controlling a predefined set of low level heuristics for solving computationally hard combinatorial optimisation problems. Being reusable methods, they are expected to be applicable to multiple problem domains, hence performing well in cross-domain search. HyFlex is a general purpose heuristic search API which separates the high level search control from the domain details enabling rapid development and performance comparison of heuristic search methods, particularly hyper-heuristics. In this study, the performance of six previously proposed selection hyper-heuristics are evaluated on three recently introduced extended HyFlex problem domains, namely 0-1 Knapsack, Quadratic Assignment and Max-Cut. The empirical results indicate the strong generalising capability of two adaptive selection hyper-heuristics which perform well across the 'unseen' problems in addition to the six standard HyFlex problem domains.

Keywords: Metaheuristic, Parameter Control, Adaptation, Move Acceptance, Optimisation

1 Introduction

Many combinatorial optimisation problems are computationally difficult to solve and require methods that use sufficient knowledge of the problem domain. Such methods cannot however be reused for solving problems from other domains. On the other hand, researchers have been working on designing more general solution methods that aim to work well across different problem domains. Hyper-heuristics have emerged as such methodologies and can be broadly categorised into two categories; *generation* hyper-heuristics to generate heuristics from existing components, and *selection* hyper-heuristics to select the most appropriate heuristic from a set of low level heuristics [3]. This study focuses on selection hyper-heuristics.

A selection hyper-heuristic framework operates on a single solution and iteratively *selects* a heuristic from a set of low level heuristics and applies it to the candidate solution.Then a *move acceptance* method decides whether to accept or reject the newly generated solution. This process is iteratively repeated until a termination criterion is satisfied. In [5], a range of simple selection methods are introduced, including *Simple Random* (SR) that randomly selects a heuristic at each step, and *Random Descent* which works similarly to SR, but the selected low level heuristic is applied repeatedly until no additional improvement in the solution is observed. Most of the simple non-stochastic basic move acceptance methods are tested in [5]; including *All Moves* (AM), which accepts all moves, *Only Improving* (OI), which accepts only improving moves and *Improving or Equal* (IE), which accepts all non-worsening moves. *Late acceptance* [4] accepts an incumbent solution if its quality is better than a solution that was obtained a specific number of steps earlier. More on selection hyper-heuristics can be found in [3].

HyFlex [14] (**Hy**per-heuristics **Flex**ible framework) is a cross-domain heuristic search API and HyFlex v1.0 is a software framework written in Java, providing an easy-to-use interface for the development of selection hyper-heuristic search algorithms along with the implementation of several problem domains, each of which encapsulates problem-specific components, such as solution representation and low level heuristics. We will refer to HyFlex v1.0 as HyFlex from this point onward. HyFlex was initially developed to support the first Cross-domain Heuristic Search Challenge (CHeSC) in 2011[3]. Initially, there were six minimisation problem domains implemented within HyFlex [14]. The HyFlex problem domains have been extended to include three more of them, including 0-1 Knapsack Problem (KP), Quadratic Assignment Problem (QAP) and Max-Cut (MAC) [1]. In this study, we only consider the 'unseen' extended HyFlex problem domains to investigate the performance and the generality of some previously proposed well performing selection hyper-heuristics.

2 Selection Hyper-heuristics for the Extended HyFlex Problem Domains

In this section, we provide a description of the selection hyper-heuristic methods which are investigated in this study. These hyper-heuristics use different combinations of heuristic selection and move acceptance methods.

Sequence-based selection hyper-heuristic (SSHH) [10] is a relatively new method which aims to discover the best performing sequences of heuristics for improving upon an initially generated solution. The hidden Markov model (HMM) is employed to learn the optimum sequence lengths of heuristics. The hidden states in HMM are replaced by the low level heuristics and the observations in HMM are replaced by the sequence-based acceptance strategies (AS). A transition probabilities matrix is utilised to determine the movement between the hidden states; and an emission probabilities matrix is employed to determine whether a particular sequence of heuristics will be applied to the candidate solution or will be coupled with another LLH. The move acceptance method used in [10] accepts all improving moves and non-improving moves with an adaptive threshold. The SSHH showed excellent performance across CHeSC 2011 problem domains achieving better overall performance than Adap-HH which was the winner of the challenge.

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

Dominance-based and random descent hyper-heuristic (DRD) [16] is an iterated multi-stage hyper-heuristic that hybridises a dominance-based and random descent heuristic selection strategies, and uses a naïve move acceptance method which accepts improving moves and non-improving moves with a given probability. The dominance-based stage uses a greedy-like method aiming to identify a set of 'active' low level heuristics considering the trade-off between the delta change in the fitness and the number of iterations required to achieve that change. The random descent stage considers only the subset of low level heuristics recommended by the dominance-based stage. If the search stagnates, then the dominance-based stage may kick in again aiming to detect a new subset of active heuristics. The method has proven to perform relatively well in the MAX-SAT and 1D bin-packing problem domains as reported in [16].

Robinhood (*ro*und-ro*bin* neighbour*hood*) hyper-heuristic [11] is an iterated multi-stage hyper-heuristic. Robinhood contains three selection hyper-heuristics. They all share the same heuristic selection method but differ in the move acceptance. The Robinhood heuristic selection allocates equal time for each low level heuristic and applies them one at a time to the incumbent solution in a cyclic manner during that time. The three move acceptance criteria employed by Robinhood are only improving, improving or equal, and an adaptive move acceptance method. The latter method accepts all improving moves and non-improving moves are accepted with a probability that changes adaptively throughout the search process. This selection hyper-heuristic outperformed eight 'standard' hyper-heuristics across a set of instances from HyFlex problem domains. A detailed description of the Robinhood hyper-heuristic can be found in [11].

Modified choice function (MCF) [6] uses an improved version of the traditional choice function (CF) heuristic selection method used in [5] and has a better average performance than CF when compared across the CHeSC 2011 competition problems. The basic idea of a choice function hyper-heuristic is to choose the best low level heuristic at each iteration. Hence, move acceptance is not needed and all moves are accepted. In the traditional CF method, each low level heuristic is assigned a score based on three factors; the recent effectiveness of the given heuristic (f_1), the recent effectiveness of consecutive pairs of heuristics (f_2), and the amount of time since the given heuristic was used (f_3) where each factor within CF is associated with a weight; α, β, and δ respectively [5]. It was also stated in the CF study that the hyper-heuristic was insensitive to the parameter settings for solving Sales Summit Scheduling problems and are consequently fixed throughout the search. MCF extends upon CF by controlling the weights of each factor for improving its cross-domain performance [6]. In MCF, the weights for f_1 and f_2 are equal as defined by the parameter ϕ_t, and the weight for f_3 is set to $1-\phi_t$. ϕ_t is controlled using a simple mechanism. If an improving move is made, then $\phi_t = 0.99$. If a non-improving move is made, then $\phi_t = max\{\phi_{t-1} - 0.01, 0.01\}$.

Fuzzy late acceptance-based hyper-heuristic (F-LAHH) [8] was implemented for solving MAX-SAT problems and showed promising results. F-LAHH utilises a fitness proportionate selection mechanism (RUA1-F1FPS) [7] for the heuristic selection method and uses late acceptance, whose list length is adaptively controlled using a fuzzy control system, for its move acceptance method. In RUA1-F1FPS, the low level heuristics are assigned scores which are updated based on acceptance of the candidate solution as defined by the RUA1 scheme. A heuristic is chosen using a fitness proportionate (roulette wheel) selection mechanism utilising Formula 1 (F1) ranking scores (F1FPS). Each low level heuristic is ranked based on their current scores using F1 ranking and are assigned probabilities to be selected proportional to their F1 rank. The fuzzy control system, as defined in [8], adapts the list length of a late acceptance move acceptance method at the start of each phase each to promote intensification or diversification within the subsequent phase of the search based on the amount of improvement over the current phase. The F1FPS scoring mechanism used in this study is the RUA1 method as used in [7, 8]. The parameters of the fuzzy system are the same as those used in [8] with the universe of discourse of the list length fuzzy sets $U = [10000, 30000]$, the initial list length of late acceptance $L_0 = 10000$, and the number of phases equal to 50.

Simple Random-Great Deluge (SR-GD) is a single-parameter selection hyper-heuristic method. At each step, a random heuristic will be selected and applied to the current solution. Great deluge move acceptance method [9] accepts improving solutions by default. A non-improving solution is only accepted if its quality is better than a threshold level at each iteration. Initially, the threshold level is set to the cost of the initially constructed solution. The threshold level is then updated at each iteration with a linear rate given by the following formula:

$$T_t = c + \Delta C \times (1 - \frac{t}{N}) \tag{1}$$

where T_t is the value of the threshold level at time t, N is the time limit, ΔC is the expected range for the maximum change in the cost, and c is the final cost.

3 Empirical Results

The methods presented in Section 2 are applied to 10 instances from each of the recently introduced HyFlex problem domains. The experiments are conducted on an i7-3820 CPU at 3.60GHz with a memory of 16.00GB. Each run is repeated 31 times with a termination criteria of 415 seconds corresponding to 600 nominal seconds of the CHeSC 2011 challenge test machine[4]. The following performance indicators are used for ranking hyper-heuristics across all three domains:
- **rank:** rank of a hyper-heuristic with respect to μ_{norm}.
- μ_{rank}**:** each algorithm is ranked based on the median objective values that they produce over 31 runs for each instance. The top algorithm is assigned to rank 1, while the worst algorithm's rank equals to the number of algorithms being considered in ranking. In case of a tie, the ranks are shared by taking the average. The ranks are then accumulated and averaged over all instances producing μ_{rank}.
- μ_{norm}**:** the objective function values are normalised to values in the range [0,1] based on the following formula:

$$norm(o,i) = \frac{o(i) - o_{best}(i)}{o_{worst}(i) - o_{best}(i)} \tag{2}$$

where $o(i)$ is the objective function value on instance i, $o_{best(i)}$ is the best objective function value obtained by all methods on instance i, and $o_{worst(i)}$ is the worst objective

[4]http://www.asap.cs.nott.ac.uk/external/chesc2011/benchmarking.html

function value obtained by all methods on instance *i*. μ_{norm} is the average normalised objective function value.

- **best:** is the number of instances for which the hyper-heuristic achieves the best median objective function value.

- **worst:** the number of instances for which the hyper-heuristic delivers the worst median objective function value.

As a performance indicator, μ_{rank} focusses on median values and does not consider how far those values are from each other for the algorithms in consideration, while μ_{norm} considers the mean performance of algorithms by taking into account the relative performance of all algorithms over all runs across each problem instance.

Table 1 summarises the results. On KP, SSHH delivers the best median values for 8 instances including 4 ties. Robinhood achieves the best median results in 5 instances including a tie. SR-GD, F-LAHH and DRD show comparable performance. On the QAP problem domain, SR-GD performs the best in 6 instances and F-LAHH shows promising results in this particular problem domain. This gives an indication that simple selection methods are potentially the best for solving QAP problems. SSHH ranked as the third best based on the average rank on QAP problem. On MAC, SSHH clearly outperforms all other methods, followed by SR-GD and then Robinhood. The remaining hyper-heuristics have relatively poor performance, with MCF being the worst of the 6 hyper-heuristics. Overall, SSHH turns out to be the best with $\mu_{norm} = 0.16$ and $\mu_{rank} = 2.28$. SR-GD also shows promising performance, scoring the second best. MCF consistently delivers weak performance in all the instances of the three problem domains. Table 1 also provides the pairwise average performance comparison of SSHH versus (DRD, Robinhood, MCF, F-LAHH and SR-GD) based on the Mann-Whitney-Wilcoxon statistical test. SSHH performs significantly better than any hyper-heuristic on all MAC instances, except Robinhood which performs better than SSHH on four out of ten instances. On the majority of the KP instances, SSHH is the best performing hyper-heuristic. SSHH performs poorly on QAP when compared to F-LAHH and SR-GD and both hyper-heuristics produce significantly better results than SSHH on almost all instances. SSHH performs statistically significantly better than the remaining hyper-heuristics on QAP.

The performance of the best hyper-heuristic from Table 1, SSHH is compared to the methods whose performances are reported in [1], including Adap-HH, which is the winner of the CHeSC 2011 competition [13], an Evolutionary Programming Hyper-heuristic (EPH) [12], Fair-Share Iterated Local Search with (FS-ILS) and without restart (NS-FS-ILS), Simple Random-All Moves (SR-AM) (denoted as AA-HH previously) and Simple Random-Improving or Equal (SR-IE) (denoted as ANW-HH previously). Table 2 summarises the results based on μ_{rank}, μ_{norm}, best and worst counts. Adap-HH performs better than SSHH in KP and QAP while SSHH performs the best on MAC. Overall, SSHH is the best method based on μ_{norm} with a value of 0.113, however Adap-HH is the top ranking algorithm based on μ_{rank} with a value of 2.53 and SSHH is the second best with a value of 3.20.

Table 1. The performance comparison of SSHH, DRD, Robinhood, MCF, F-LAHH and SR-GD over 31 runs for each instance. The best median values per each instance are highlighted in bold. Based on the Mann-Whitney-Wilcoxon test, for each pair of algorithms; SSHH versus X; SSHH $>$ ($<$) X indicates that SSHH (X) is better than X (SSHH) and this performance variance is statistically significant with a confidence level of 95%, and SSHH $\geq$ ($\leq$) X indicates that there is no statistical significant between SSHH and X, but SSHH (X) is better than X (SSHH) on average.

		SSHH				DRD				Robinhood				MCF				F-LAHH				SR-GD		
Domain	Instance	med.	rank	min.	vs	med.	rank	min.	vs	med.	rank	min.	vs	med.	rank	min.	vs	med.	rank	min.	vs	med.	rank	min.
KP	Inst1	**-104046**	1	-104046	$>$	-104025	4.5	-104044	$>$	-104034	3	-104046	$>$	-103998	6	-104046	$>$	-104037	2	-104046	$>$	-104025	4.5	-104046
	Inst2	**-1247642**	1	-1261320	$>$	-1208666	6	-1208666	$\geq$	-1241628	2	-1253664	$>$	-1226625	3	-1244413	$>$	-1212253	5	-1220422	$>$	-1212829	4	1221623
	Inst3	**-241934**	1	-242963	$>$	-232525	6	-233066	$>$	-236420	5	-238447	$>$	-239323	2	-240023	$>$	-238397	4	-239848	$>$	-238664	3	-239192
	Inst4	**-431350**	1	-431362	$>$	-431333	2	-431349	$>$	-431320	4	-431338	$>$	-431325	3	-431341	$>$	-431314	6	-431331	$>$	-431316	5	-431329
	Inst5	**-396167**	3	-396167	$\leq$	**-396167**	3	-396167	$\leq$	**-396167**	3	-396167	$>$	-396127	6	-396167	$>$	**-396167**	3	-396167	$\leq$	**-396167**	3	-396167
	Inst6	-4251693	4	-4268665	$>$	-4248962	5.5	-4248962	$<$	**-4262735**	1	-4312111	$>$	-4248962	5.5	-4321660	$\leq$	-4251867	3	-4268839	$<$	-4253175	2	-4273295
	Inst7	-929052	2	-943136	$>$	-924303	5	-924357	$>$	-924346	4	-933892	$>$	-923904	6	-939879	$>$	-924937	3	-941397	$<$	**-935411**	1	-940485
	Inst8	**-1577175**	2.5	-1577175	$>$	-1577166	5	-1577175	$\leq$	**-1577175**	2.5	-1577175	$>$	-1572999	6	-1577175	$>$	**-1577175**	2.5	-1577175	$\leq$	**-1577175**	2.5	-1577175
	Inst9	**-1530477**	1.5	-1530511	$>$	-1530465	3.5	-1530485	$\geq$	**-1530477**	1.5	-1530494	$>$	-1530465	3.5	-1530498	$>$	-1530453	5.5	-1530484	$>$	-1530453	5.5	-1530463
	Inst10	**-1467357**	2	-1467362	$>$	**-1467357**	2	-1467357	$>$	**-1467357**	2	-1467362	$>$	-1457070	6	-1467353	$>$	-1467353	4.5	-1467361	$>$	-1467353	4.5	-1467362
	rank average	1.90				4.25				2.80				4.70				3.85				3.50		
	norm average	0.10				0.27				0.17				0.30				0.25				0.17		
QAP	Inst1	152572	4	152224	$\leq$	**152000**	1	152000	$\geq$	152686	5	152334	$>$	153398	6	152700	$<$	152372	3	152122	$<$	152258	2	152068
	Inst2	154492	3	154130	$>$	155000	5	154000	$\geq$	154616	4	154136	$>$	155300	6	154706	$<$	154178	2	153960	$<$	**154172**	1	154016
	Inst3	148374	3	147930	$\geq$	148604	5	147916	$\leq$	148462	4	148088	$>$	149584	6	148604	$<$	148140	2	148026	$<$	**148056**	1	147900
	Inst4	150366	4	149782	$\geq$	150336	3	149724	$\geq$	150380	5	150002	$>$	151016	6	150164	$<$	149978	2	149730	$<$	**149892**	1	149688
	Inst5	21419490	4	21325030	$\leq$	21400000	3	21300000	$<$	21383596	2	21325716	$>$	21598704	6	21414834	$>$	21495226	5	21351226	$<$	**21361794**	1	21207680
	Inst6	1190346287	4	1186663179	$\geq$	1190000000	3	1190000000	$>$	1199401744	5	1192546366	$>$	1249957271	6	1204968089	$<$	1188454126	2	1186678730	$<$	**1188111647**	1	1186811188
	Inst7	504406437	4	500015697	$\geq$	504000000	3	502000000	$>$	508225133	5	504102563	$>$	511240596	6	506396735	$<$	**501945504**	1	500096792	$<$	502027073	2	499922042
	Inst8	44892452	3	44855568	$>$	44900000	4	44800000	$>$	44933092	6	44875514	$\geq$	44903670	5	44869704	$<$	**44859724**	1	44841194	$<$	44863858	2	44842660
	Inst9	8179752	3	8151040	$>$	8200846	4	8165384	$>$	8202996	5	8177206	$>$	8254190	6	8213094	$<$	**8162896**	1	8157314	$<$	8163776	2	8150316
	Inst10	273622	3	273216	$>$	274000	5	273000	$>$	273908	4	273590	$>$	274404	6	273566	$<$	273460	2	273264	$<$	**273362**	1	273216
	rank average	3.50				3.60				4.50				5.90				2.10				1.40		
	norm average	0.24				0.29				0.32				0.58				0.16				0.12		
MAC	Inst1	**-41101646**	1	-41517765	$>$	-39393891	6	-40202568	$>$	-40471041	3	-40863976	$>$	-40157605	5	-40967725	$>$	-40419083	4	-41268393	$>$	-40756746	2	-41377263
	Inst2	**-273938900**	1	-277548425	$>$	-266329920	5	-268635140	$>$	-269502099	2	-271045451	$>$	-256423018	6	-261640131	$>$	-266773056	4	-274334343	$>$	-267482996	3	-269292120
	Inst3	**-3056**	1	-3062	$>$	-3014	6	-3030	$>$	-3043	4.5	-3051	$>$	-3046	3	-3056	$>$	-3043	4.5	-3053	$>$	-3053	2	-3057
	Inst4	**-3040**	1	-3050	$>$	-2991	6	-3012	$>$	-3027	4	-3032	$>$	-3027	4	-3033	$>$	-3027	4	-3037	$>$	-3035	2	-3047
	Inst5	**-3041**	1	-3051	$>$	-3000	6	-3016	$>$	-3028	4.5	-3034	$>$	-3029	3	-3042	$>$	-3028	4.5	-3042	$>$	-3038	2	-3045
	Inst6	**-13243**	1	-13300	$>$	-13047	6	-13106	$>$	-13204	3	-13246	$>$	-13176	5	-13241	$>$	-13186	4	-13247	$>$	-13216	2	-13284
	Inst7	-1352	2	-1358	$>$	-1246	6	-1278	$<$	**-1362**	1	-1368	$>$	-1316	5	-1330	$>$	-1322	4	-1342	$>$	-1334	3	-1346
	Inst8	-10074	2	-10125	$>$	-9819	6	-9872	$<$	**-10152**	1	-10190	$>$	-9964	5	-9996	$>$	-10004	4	-10101	$>$	-10046	3	-10078
	Inst9	-454	2.5	-458	$>$	-416	6	-430	$\leq$	-454	2.5	-456	$>$	-444	4	-454	$>$	-440	5	-450	$<$	**-456**	1	-456
	Inst10	-2912	2	-2960	$>$	-2676	6	-2704	$<$	**-2942**	1	-2952	$>$	-2810	5	-2842	$>$	-2848	4	-2906	$>$	-2884	3	-2926
	rank average	1.45				5.90				2.65				4.50				4.20				2.30		
	norm average	0.14				0.74				0.21				0.40				0.34				0.22		
	μ_{rank}	2.28				4.58				3.32				5.03				3.38				2.40		
	μ_{norm}	0.16				0.43				0.23				0.43				0.25				0.17		

Table 2. The performance comparison of SSHH, Adap-HH, FS-ILS, NR-FS-ILS, EPH, SR-AM and SR-IE

KP Problem Domain

rank	method	μ_{rank}	μ_{norm}	best	worst
1	Adap-HH	1.95	0.027	8	0
2	EPH	2.35	0.053	4	0
3	SSHH	2.45	0.059	5	0
4	SR-AM	4.40	0.148	2	0
5	SR-IE	5.55	0.328	0	4
6	NR-FS-ILS	5.60	0.361	1	6
7	FS-ILS	5.70	0.395	1	2

QAP Problem Domain

rank	method	μ_{rank}	μ_{norm}	best	worst
1	NR-FS-ILS	1.95	0.100	5	0
2	Adap-HH	2.50	0.103	2	0
3	FS-ILS	2.85	0.103	3	0
4	EPH	3.80	0.133	0	0
5	SR-AM	4.10	0.146	1	0
6	SSHH	5.80	0.189	0	0
7	SR-IE	7.00	0.634	0	10

MAC Problem Domain

rank	method	μ_{rank}	μ_{norm}	best	worst
1	SSHH	1.35	0.092	9	0
2	SR-AM	2.45	0.252	1	0
3	Adap-HH	3.15	0.275	0	0
4	NR-FS-ILS	4.00	0.374	0	0
5	FS-ILS	4.85	0.392	1	2
6	EPH	5.60	0.519	0	1
7	SR-IE	6.60	0.732	0	7

Overall

rank	method	μ_{rank}	μ_{norm}	best	worst
1	SSHH	3.20	0.113	14	0
2	Adap-HH	2.53	0.135	10	0
3	SR-AM	3.65	0.182	4	0
4	EPH	3.92	0.235	4	1
5	NR-FS-ILS	3.85	0.278	6	6
6	FS-ILS	4.47	0.297	5	4
7	SR-IE	6.38	0.565	0	21

4 Conclusion

A hyper-heuristic is a search methodology, designed with the aim of reducing the human effort in developing a solution method for multiple computationally difficult optimisation problems via automating the mixing and generation of heuristics. The goal of this study was to assess the level of generality of a set of selection hyper-heuristics across three recently introduced HyFlex problem domains. The empirical results show that both Adap-HH and SSHH perform better than the previously proposed algorithms across the problem domains included in the HyFlex extension set. Both adaptive algorithms embed different online learning mechanisms and indeed generalise well on the ‘unseen’ problems. It has also been observed that the choice of heuristic selection and move acceptance combination could lead to major performance differences across a diverse set of problem domains. This particular observation is aligned with previous findings in [2, 15].

References

1. Adriaensen, S., Ochoa, G., Nowé, A.: A benchmark set extension and comparative study for the HyFlex framework. In: IEEE Congress on Evolutionary Computation. pp. 784–791 (2015)
2. Bilgin, B., Özcan, E., Korkmaz, E.E.: An experimental study on hyper-heuristics and exam scheduling. In: Burke, E.K., Rudová, H. (eds.) Practice and Theory of Automated Timetabling VI. Lecture Notes in Computer Science, vol. 3867, pp. 394–412 (2007)

3. Burke, E.K., Gendreau, M., Hyde, M., Kendall, G., Ochoa, G., Özcan, E., Qu, R.: Hyper-heuristics: a survey of the state of the art. Journal of the Operational Research Society 64(12), 1695–1724 (2013)
4. Burke, E.K., Bykov, Y.: A late acceptance strategy in hill-climbing for exam timetabling problems. In: Proceedings of the 7th International Conference on the Practice and Theory of Automated Timetabling (PATAT '08) (2008)
5. Cowling, P., Kendall, G., Soubeiga, E.: A hyperheuristic approach to scheduling a sales summit. In: Burke, E., Erben, W. (eds.) Practice and Theory of Automated Timetabling III, Lecture Notes in Computer Science, vol. 2079, pp. 176–190. Springer Berlin Heidelberg (2001)
6. Drake, J.H., Özcan, E., Burke, E.K.: An improved choice function heuristic selection for cross domain heuristic search. In: Coello, C.A.C., Cutello, V., Deb, K., Forrest, S., Nicosia, G., Pavone, M. (eds.) Parallel Problem Solving From Nature (PPSN XII). Lecture Notes in Computer Science, vol. 7492, pp. 307–316. Springer Berlin Heidelberg (2012)
7. Jackson, W.G., Özcan, E., Drake, J.H.: Late acceptance-based selection hyper-heuristics for cross-domain heuristic search. In: 13th UK Workshop on Computational Intelligence. pp. 228–235 (2013)
8. Jackson, W., Özcan, E., John, R.I.: Fuzzy adaptive parameter control of a late acceptance hyper-heuristic. In: Computational Intelligence (UKCI), 14th UK Workshop on. pp. 1–8 (2014)
9. Kendall, G., Mohamad, M.: Channel assignment optimisation using a hyper-heuristic. In: Proceedings of the IEEE Conference on Cybernetic and Intelligent Systems. pp. 790–795 (2004)
10. Kheiri, A., Keedwell, E.: A sequence-based selection hyper-heuristic utilising a hidden Markov model. In: Proceedings of the 2015 on Genetic and Evolutionary Computation Conference. pp. 417–424. GECCO '15, ACM, New York, NY, USA (2015)
11. Kheiri, A., Özcan, E.: A hyper-heuristic with a round robin neighbourhood selection. In: Middendorf, M., Blum, C. (eds.) Evolutionary Computation in Combinatorial Optimization, Lecture Notes in Computer Science, vol. 7832, pp. 1–12. Springer Berlin Heidelberg (2013)
12. Meignan, D.: An evolutionary programming hyper-heuristic with co-evolution for CHeSC1́1. In: The 53rd Annual Conference of the UK Operational Research Society (OR53) (2011)
13. 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 and Application (MISTA2011). pp. 374–393 (2011)
14. Ochoa, G., Hyde, M., Curtois, T., Vazquez-Rodriguez, J.A., Walker, J., Gendreau, M., Kendall, G., McCollum, B., Parkes, A.J., Petrovic, S., Burke, E.K.: HyFlex: a benchmark framework for cross-domain heuristic search. In: Hao, J.K., Middendorf, M. (eds.) Evolutionary Computation in Combinatorial Optimization, LNCS, vol. 7245, pp. 136–147 (2012)
15. Özcan, E., Bilgin, B., Korkmaz, E.E.: A comprehensive analysis of hyper-heuristics. Intelligent Data Analysis 12(1), 3–23 (2008)
16. Özcan, E., Kheiri, A.: A hyper-heuristic based on random gradient, greedy and dominance. In: Gelenbe, E., Lent, R., Sakellari, G. (eds.) Computer and Information Sciences II, pp. 557–563. Springer London (2012)