Journal of Engineering and Applied Sciences 13 (12): 4535-4541, 2018
ISSN: 1816-949X

A Comparative Study on three Component Selection Mechanisms for Hyper-Heuristics in Expensive Optimization

Jia Hui Ong and Jason Teo
Evolutionary Computing Laboratory, Faculty of Computing and Informatics,
Universiti Malaysia Sabah, Kota Kinabalu, Malaysia

Abstract: Numerous studies in optimization problems often lead to tailoring a specific algorithm to adapt to the problem instances, especially in expensive optimization problems. The focus of these researches is often to challenge and outperform another algorithm in the specific problem instant. Once the problem instants changes, more tailoring of the algorithm has to be done in order for the algorithm to perform at an optimum level. Expensive optimization often requires a large amount of resources to run on such as computational power, high run-time budget and consumes a lot of time. As such, tailoring an algorithm to perform well in expensive optimization requires a lot of expertise and time. Hyper-heuristics is an approach that utilizes a set of Low-Level Heuristic (LLH) and a selection mechanism to solve expensive optimization problems. The main aim of using hyper-heuristics is to be able to apply a general yet efficient optimizationalgorithm to all expensive problem instances with very minor or minimal tweaks. In this study, three different selection mechanisms for Hyper-heuristics are introduced and compared against one of the top performing expensive optimization algorithms known asthe Mean-Variance Mapping Optimization (MVMO) as described in the CEC 2015 and 2016 expensive optimization competitions. Three variants of hyper-heuristics were used in this study, Simple Random All Moves Acceptance (SRAMA), Tabu-Search All Moves Acceptance (TSAMA) and Random Gradient Descent All Moves Acceptance (RGDAMA). The set of LLH will also include a simplified version of MVMO. The performance of hyper-heuristics is highly encouraging against a specifically tailored algorithm for CEC test set of expensive optimization problems.

Key words: Hyper-heuristics, expensive optimization, hyper-heuristic component selection, MVMO, minor, TSAMA

INTRODUCTION

Expensive optimization problems refer to optimization problems in the real world that requires a large amount of resources to run. Hence, it is crucial that optimum solution can be found in a short number of evaluations. A competition on expensive optimization problem in Congress on Evolutionary Computation (CEC) 2015 was held to address the need to focus on this area of research. Most of the researcher participated in this competition merely focus on tuning and tailoring a specific algorithm that can run well in the test suit provided. Mean Variance Mapping Optimization (MVMO) emerge as the best performing algorithm in the competition and again in CEC 2016. Tuning and tailoring an algorithm is time consuming and requires expertise on the algorithm. Since, the tailored algorithm was created to perform well towards a specific algorithm when the problem instances change more time and expertise are needed to tune the algorithm, hence, the reusability of tailored algorithm is relatively low. Hyper-heuristics is defined as high-level approaches that utilize a set of low level heuristic to any problem instantly at any decision point. It was first introduced by Cowling *et al*. (2000) and he further extends the use of hyper-heuristics in scheduling problems (Cowling and Chakhlevith, 2003). A throughout study of hyper-heuristics in recent-years was conducted by Burke *et al*. (2013) he review the method and structure of hyper-heuristics usage. Burke has also identified one of the main challenges of how to develop the framework of hyper-heuristics to be more generally applicable to all search methodologies.

There are only a few hyper-heuristics usages in continuous problem domain (Kiraz *et al*., 2013; Topcuoglu *et al*., 2014; Maashi *et al*., 2014). McClymont and Keedwell (2011) attempted to use hyper-heuristics in a reduce number of evaluations multi-objectives problems. Thus, in this study, we

Corresponding Author: Jia Hui Ong, Evolutionary Computing Lab., Faculty of Computing and Informatics, Universiti Malaysia Sabah, Kota Kinabalu, Malaysia

compare the selection mechanism and also compare hyper-heuristics against a tailored algorithm in continuous real expensive optimization problems.

First part of this study has been structured to go through the basic framework of a hyper-heuristics and explanation of how the component of the hyper-heuristics research. It is then followed by describing the CEC 2015 bench mark test problems. The purpose algorithm details will also be presented before proceeding to the results and discussion. The future research of the usage of hyper-heuristics will be the last study of this study.

MATERIALS AND METHODS

Hyper-heuristics

Strucuture of hyper-heuristics: Hyper-heuristics has two distinct categories namely hyper-heuristics selection and hyper-heuristics generative (Burke *et al*., 2010) as shown in Fig. 1. Selection hyper-heuristics will select and applies low level heuristics to modify a solution in hand (Kheiri, 2014) while generative hyper-heuristics refer to the hyper-heuristics methodologies for generating new heuristics from the component of existing ones (Burke *et al*., 2010).

Forming the second level of the hyper-heuristics framework is the perturbation and constructive heuristics (Hoos and Stutzle, 2004). In perturbation heuristics, the searching processesinvolve complete solutions while constructive heuristic will only processes partial solution. Feedback mechanism behind hyper-heuristics framework is to support learning algorithmand there are three categories in this feedback mechanism, online learning, offline learning and no learning. Online learning is a feedback that learns during the process while offline learning is a feedback that learns before the start of the process.

Selection hyper-heuristics: Selection hyper-heuristics are consider as a high level problem solving framework with two major mechanisms namely heuristic selection and move acceptance as shown in Fig. 2. There are various heuristic selection mechanisms and cowling presented a few simple heuristic selections in his research (Cowling *et al*., 2000). They studied the performance of Simple Random (SR), Random Descent (Gradient) (RD (G)), Random Permutation (RP), Random Permutation Descent (Gradient) (RPD (G)) and Choice Function (CF).

Simple random selection chooses randomly based on a uniform probability. Random descent (gradient) randomly selects the low-level heuristic and applies it until there is no improvement and the next selection will be randomly selected again. Random permutation

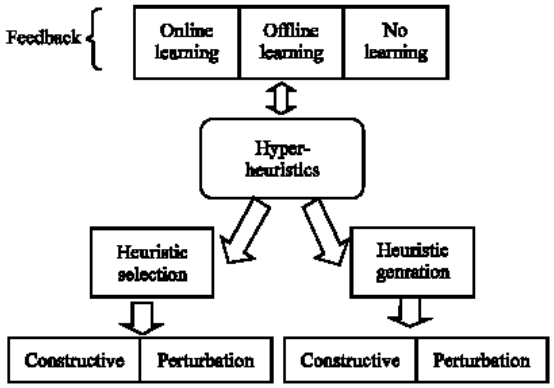

Fig. 1: Classification of hyper-heuristicss

S_{curr} = candidate solution
S_{best} = best solution
LLH = {LLH_1, LLH_2, LLH_3... LLH_n}
$S_{curr} \leftarrow S_{initial}$
$S_{best} \leftarrow S_{initial}$
WHILE Termination_Criteria equals **False** ***Do***
    $LLH_i \leftarrow$ Select_llh(LLH)
    $S_{new} \leftarrow$ Apply_heuristic(LLH_i, S_{curr})
    IF Acceptance(S_{curr}, S_{new}) **Then**
        $S_{curr} \leftarrow S_{new}$
    End
$S_{best} \leftarrow$ Update_Best_Solution(S_{curr})
End WHILE
return S_{best}

Fig. 2: Selection hyper-heuristics pseudocode

randomly generates low-level heuristic into a list of all low level heuristics and applies them one after another at each step sequentially. Random permutation descent (gradient) use the same concept as random permutation but randomdescent (gradient) approach without changing the order of heuristics. Greedy uses all low-level heuristics to generate potential solution and selects a heuristic that has the biggest improvement. Choice Function uses a mechanism that grades the low-level heuristic based on the improvement or worsen solution and also the duration of the last use low-level heuristic.

In Cowling's research, Tabu-searched was introduced as a selection mechanism (Cowling and Chakhlevith, 2003). Tabu-search will ranked the low-level heuristic to determine the next selection while keeping a list of disallow low-level heuristic to avoid using bad performing low-level heuristic. Tabu-search was first introduced by Glover (1986) in an integer programming.

Second mechanism of selection hyper-heuristics is move acceptance. This mechanism is the decision maker

to accept or not for the newly generated solution. The basic move acceptances that were suggested by Cowling *et al*. (2000) are:

- All Moves (AM)
- Only Improving (OI)
- Improving or Equal (IE)

AM will accept all the generated solution, OI will check whether the current solution improved from previous best and only accept all the improved solution and IE accepts non worsening solutions from the previous best solution. In move acceptance criteria, it can be further distinguished into deterministic and non-deterministic. Deterministic refers to move acceptance that return the same decision at every iteration while non-deterministic refer to move acceptance criteria that depends on the current iteration. It can then be categorized into stochastic or non-stochastic category. These categories exist when probabilistic framework is considered while making acceptance decision.

Multi-point search refers to the search that maintains a pool of population solution while single-point search refers to search that improves and maintains a single solution. Most of the researches done are base on single point perturbation approach and only a few studies use the multi-point perturbation approach (Burke *et al*., 2013).

Test suits and hyper-heuristics setup: Competition on expensive optimization was held in CEC 2015. In the competition, test suits of 15 benchmark problems were introduced and each test suits are only allowed up to 500 evaluations. This study will also use the same test suits in CEC 2015. It comprises from F1-F15 benchmark optimization problems as shown in Table 1. Mean Variance Mapping Optimization (MVMO) were the best performing algorithm in CEC 2015 expensive optimization competition, hence, the results from MVMO will be used as comparison guideline.

In this study, three variants of hyper-heuristics were used. Simple Random All Moves Acceptance (SRAMA), Tabu-Search All Moves Acceptance (TSAMA) and Random Gradient Descent All Moves Acceptance (RGDAMA). These variants will employ perturbation multi-point search. There are five low-level heuristics that will be used by SRAMA, TSAMA and RGDAMA:

- Covariance Matrix Adaptation (CMA), $\lambda = 4+3 \log (N)$, $\mu = \lambda/2$
- Particle Swarm Optimization (PSO), initial velocity 1 and maximum velocity 3
- Differential Evolution (DE), Cr = 0.9, F = 0.2

Table 1: Test function for CEC 2015

Function name	F_i
Rotated bent cigar function	100
Rotated discus function	200
Shifted and rotated weierstrass function	300
Shifted and rotated schwefel's function	400
Shifted and rotated katsuura function	500
Shifted and rotated happycat function	600
Shifted and rotated HGB at function	700
Shifted and rotated expanded griewan's plus rosenbrock's function	800
Shifted and rotated expanded scaffer's F6 function	900
Hybrid function 1 (N = 3)	1000
Hybrid function 2 (N = 4)	1100
Hybrid function 3 (N = 5)	1200
Composition function 1 (N = 5)	1300
Composition function 2 (N = 3)	1400
Composition function 3 (N = 5)	1500

- Genetic Algorithms (GA), Cr = 0.5, mutation rate = 0.1 tournament selection, tournament size = 2
- Mean variance mapping optimization, archive size = 3

These LLH were chosen as a popular algorithm used in many research fields such as software effort estimation (Thamarai and Murugavalli, 2016), cluster determination (Jun, 2006) and project scheduling (Nasereddin, 2006).

To utilize the basic MVMO an archive was used to store three best solutions found. This archive is used to calculate the mean and shape variable to generate new solution as describe by Gonzalez-Longatt *et al*. (2012). A basic version of MVMO was used instead of the extended version.

SRAMA uses simple random as heuristic selection and all moves acceptance criteria while TSAMA uses Tabu-Search heuristic selection and All Moves Acceptance criteria. In RGDAMA, Random Gradient Descent is the heuristic selection used while the acceptance criteria are the same as SRAMA and TSAMA. Island based uses a single heuristic chosen by heuristic selection and controls a population in a single iteration while single based will allow different heuristics to generate single solution that forms a population in a single iteration. It was found that island based setup yields better results in a multi-point search, hence, Island base setup will be used in this study for all algorithms.

Experimental setup: The proposed algorithm will be tested using the CEC 2015 expensive optimization bench mark test suites. Population size of 5 was used for all algorithms as it was found to be the ideal population size to be used in order to get a good result from island base setup.

Each algorithm will be tested on the fifteen benchmark optimization problems from F1-F15 and each function will be tested for 51 times as per CEC 2015 expensive optimization criteria. Evaluations of all

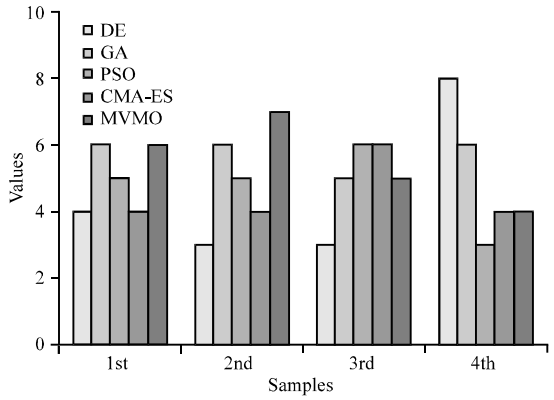

Fig. 3: TSAMA F1 LLH selection of random selected run

experiments were carried out on the same PC. As per standard rules in CEC 2015 expensive optimization competition only 500 evaluations are allowed (Fig. 3).

RESULTS AND DISCUSSION

In Table 2, SRAMA shows that it was able to compete with MVMO by performing well in eight out of fifteen functions in the test suits. SRAMA was not able to out perform MVMO in unimodal function which is F1 and F2. MVMO and SRAMA results in simple multimodal functions (F3-F9) are comparable, since, SRAMA able to perform well in four out of five simple multimodal functions. In hybrid functions (F10-F12) and composite function (F13-F15) SRAMA out perform MVMO where it manages to do well in two out of three functions, respectively for both hybrid and composite functions.

Table 3 shows the results of TSAMA compared to MVMO. TSAMA performed well in eight functions where else MVMO performed well in seven functions. Again in unimodal functions MVMO dominates as TSAMA wasn't able to perform well in any of the unimodal functions. In simple multimodal functions TSAMA performed well in four of seven functions. While in hybrid functions and composite function TSAMA out perform MVMO.

RGDAMA performance against MVMO is shown in Table 4. MVMO shows dominance in unimodal functions by performing better against all three of the hyper-heuristics variant but could not perform well in hybrid and composite functions against all three algorithms.

In Table 5, the three algorithms are compared against each other in order to investigate the better performance of these selection mechanisms against each other. TSAMA outper form SRAMA and RGDAMA in ten out of fifteen functions. SRAMA manage to perform well in four functions while RGDAMA only manage to

Table 2: Results of SRAMA-I5 against MVMO

F-values	SRAMA	MVMO	Results
1	1.47E+08	1.93E+02	Worse
2	1.47E+04	1.68E-02	Worse
3	7.26E+00	9.40E+00	Better
4	5.78E+02	4.65E+02	Worse
5	1.12E+00	1.13E+00	Better
6	9.61E-01	3.26E-01	Worse
7	4.43E+00	6.37E-01	Worse
8	2.31E+01	4.14E+01	Better
9	3.65E+00	4.01E+00	Better
10	4.61E+04	4.97E+02	Worse
11	7.14E+00	1.17E+01	Better
12	1.27E+02	2.00E+02	Better
13	3.19E+02	3.16E+02	Worse
14	1.95E+02	2.06E+02	Better
15	1.99E+02	4.76E+02	Better

Table 3: Results of TSAMA-I5 against MVMO

F-values	TSAMA	MVMO	Results
1	1.30E+08	1.93E+02	Worse
2	1.38E+04	1.68E-02	Worse
3	7.15E+00	9.40E+00	Better
4	5.86E+02	4.65E+02	Worse
5	1.11E+00	1.13E+00	Better
6	9.31E-01	3.26E-01	Worse
7	2.88E+00	6.37E-01	Worse
8	1.22E+01	4.14E+01	Better
9	3.60E+00	4.01E+00	Better
10	2.66E+04	4.97E+02	Worse
11	7.15E+00	1.17E+01	Better
12	1.37E+02	2.00E+02	Better
13	3.17E+02	3.16E+02	Worse
14	1.95E+02	2.06E+02	Better
15	2.24E+02	4.76E+02	Better

Table 4: Results of RDGAMA-I5 against MVMO

F-values	RGDAMA	MVMO	Results
1	3.81E+08	1.93E+02	Worse
2	1.70E+04	1.68E-02	Worse
3	7.69E+00	9.40E+00	Better
4	6.61E+02	4.65E+02	Worse
5	1.05E+00	1.13E+00	Better
6	1.53E+00	3.26E-01	Worse
7	8.67E+00	6.37E-01	Worse
8	1.31E+02	4.14E+01	Better
9	3.65E+00	4.01E+00	Better
10	9.97E+04	4.97E+02	Worse
11	7.78E+00	1.17E+01	Better
12	1.28E+02	2.00E+02	Better
13	3.25E+02	3.16E+02	Worse
14	1.97E+02	2.06E+02	Better
15	3.16E+02	4.76E+02	Better

Table 5: Results of SRAMA-I5 against MVMO

F-values	SRAMA	MVMO	Results
1	1.47E+08	1.93E+02	Worse
2	1.47E+04	1.68E-02	Worse
3	7.26E+00	9.40E+00	Better
4	5.78E+02	4.65E+02	Worse
5	1.12E+00	1.13E+00	Better
6	9.61E-01	3.26E-01	Worse
7	4.43E+00	6.37E-01	Worse
8	2.31E+01	4.14E+01	Better
9	3.65E+00	4.01E+00	Better
10	4.61E+04	4.97E+02	Worse
11	7.14E+00	1.17E+01	Better
12	1.27E+02	2.00E+02	Better
13	3.19E+02	3.16E+02	Worse
14	1.95E+02	2.06E+02	Better
15	1.99E+02	4.76E+02	Better

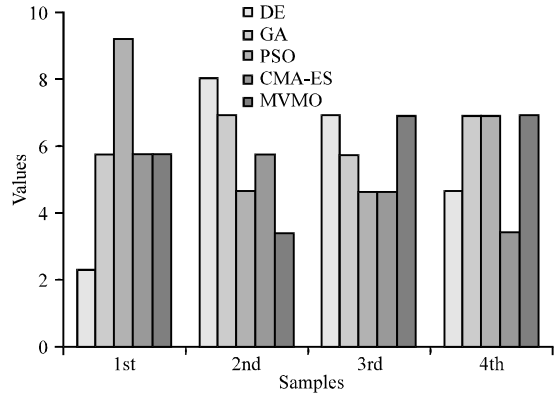

Fig. 4: TSAMA F1 LLH selection for the best run

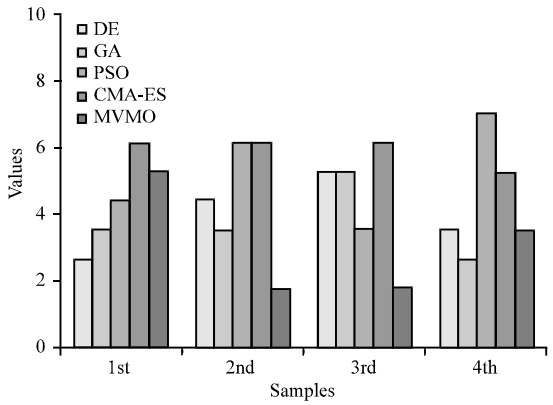

Fig. 5: TSAMA F3 LLH selection of random selected run

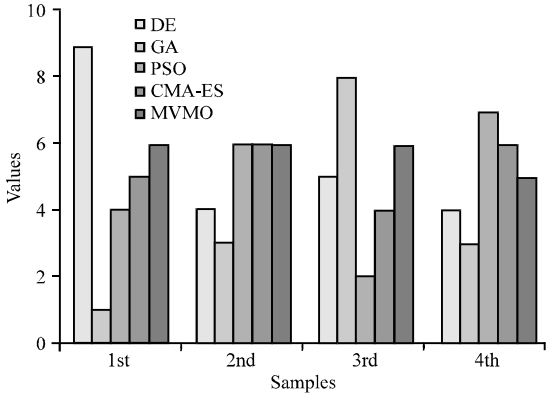

Fig. 6: TSAMA F3 LLH selection for the best run

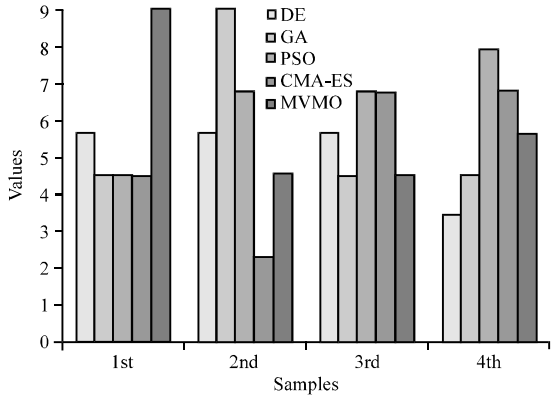

Fig. 7: TSAMA F10 LLH selection of random selected run

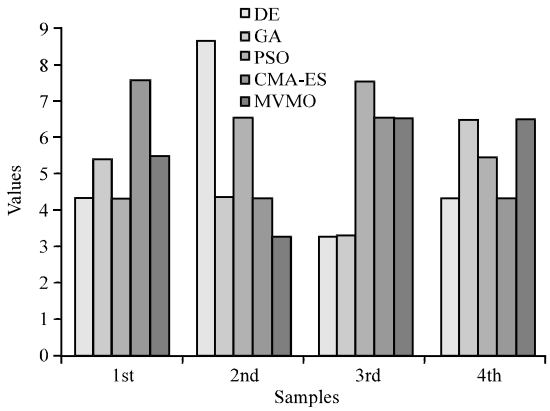

Fig. 8: TSAMA F10 LLH selection for the best run

perform well in one functions. TSAMA performed well in the unimodal, simple multimodal and composite functions. SRAMA has shown better results in hybrid functions.

To investigate further, the selection of LLH made by the selection mechanism for all three algorithms was recorded. For each type of function group in the test suit only one function are selected and presented in this study. Unimodal function F1, simple multimodal F3, hybrid F10 and composite functions F13 are selected. A random run will be selected to be compare with the best run in the same function. Since, TSAMA outperform the other two algorithms, the LLH selection by TSAMA are use as comparison. The 500 evaluations are then broken into four parts for example in Fig. 4. For purpose of this is to gain more detail observation of the LLH selection. Figure 4 and 5 shows the TSAMA selection best run. In early first quarter of the evaluation PSO are used more often while DE are used mostly in second and third quarter of the evaluation. Figure 6-8 shows TSAMA best run in F3. In the first quarter of the LLH selection DE was mainly used to generate new solution. GA in the other hand was not use that often in the first quarter but in the third quarter GA was choosen more often to be used to generate new solution. In the hybrid function F10, the combination of DE, GA and CMA-ES was used for most of the time in the first and second quarter. In composite function F13 best run by TSAMA is shown in Fig. 9 and 10 overall the usage of GA and CMA-ES are mainly used to generate new solution.

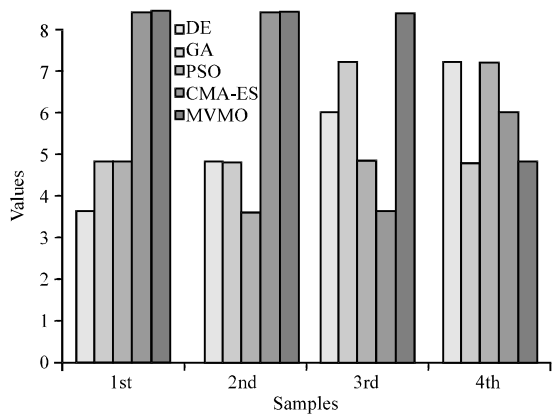

Fig. 9: TSAMA F13 LLH selection of random selected run

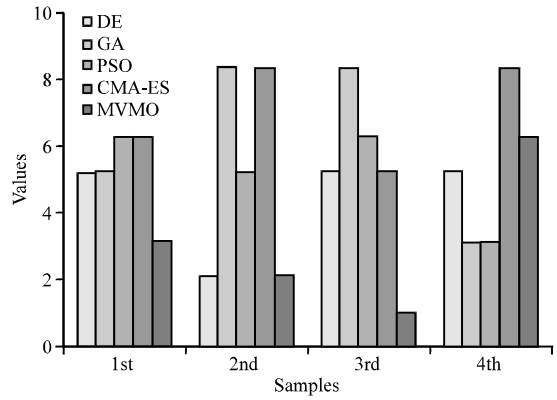

Fig. 10: TSAMA F13 LLH selection for the best run

CONCLUSION

From the results obtained, it shows that hyper-heuristics is capable to post good results as a generalized algorithm against a tailored algorithm. If a generalized algorithm is able to post good solutions it will reduce the time and expertise needed to tailored an algorithm for different problem instances. More attention and research on hyper-heuristics in continuous problem domain should be done in order to gain from the convenient of using hyper-heuristics.

Hyper-heuristics leverage on the usage of different LLH to solve different type of problems and this might help in addressing no free lunch theorem. From the results of observing the LLH selection by using different algorithm at different stages of the search process will enable good results that can be compared with specific tailored algorithms.

More research on researching hyper-heuristics selection mechanism is need as well as addressing the acceptance criteria. Both of mechanism plays an important role that will enable hyper-heuristics to be a generalized algorithm that can be applied in any problem instances without major tailoring towards the problem.

REFERENCES

Burke, E.K., M. Gendreau, M. Hyde, G. Kendall and G. Ochoa *et al*., 2013. Hyper-heuristics: A survey of the state of the art. J. Oper. Res. Soc., 64: 1695-1724.

Burke, E.K., M. Hyde, G. Kendall, G. Ochoa and E. Ozcan *et al*., 2010. A Classification of Hyper-Heuristic Approaches. In: Handbook of Metaheuristics, Gendreau, M. and J.Y. Potvin (Eds.). Springer, Boston, MA., ISBN:978-1-4419-1663-1, pp: 449-468.

Cowling, P. and K. Chakhlevitch, 2003. Hyperheuristics for managing a large collection of low level heuristics to schedule personnel. Proceedings of the 2003 International Congress on Evolutionary Computation (CEC'03) Vol. 2, December 8-12, 2003, IEEE, Canberra, ACT, Australia, pp: 1214-1221.

Cowling, P., G. Kendall and E. Soubeiga, 2000. A hyperheuristic approach to scheduling a sales summit. Proceedings of the 3rd International Conference on the Practice and Theory of Automated Timetabling, August 16-18, 2000, Springer,Berlin,Heidelberg,ISBN:978-3-540-42421-5, pp: 176-190.

Glover, F., 1986. Future paths for integer programming and links to artificial intelligence. Comput. Operat. Res., 13: 533-549.

Gonzalez-Longatt, F., J. Rueda, I. Erlich, W. Villa and D. Bogdanov, 2012. Mean variance mapping optimization for the identification of Gaussian mixture model: Test case. Proceedings of the 2012 6th IEEE International Conference on Intelligent Systems (IS), September 6-8, 2012, IEEE, Sofia, Bulgaria, ISBN: 978-1-4673-2276-8, pp: 158-163.

Hoos, H.H. and T. Stutzle, 2004. Stochastic Local Search: Foundations and Applications. Elsevier, Amsterdam, The Netherlands.

Jun, S., 2006. A hybrid genetic algorithm and new criterion for determining the number of clusters. Int. J. Soft Comput., 1: 313-318.

Kheiri, A., 2014. Multi-stage hyper-heuristics for optimisation problems. PhD Thesis, University of Nottingham, Nottingham NG7 2RD, UK.

Kiraz, B., A.S. Etaner-Uyar and E. Ozcan, 2013. Selection hyper-heuristics in dynamic environments. J. Oper. Res. Soc., 64: 1753-1769.

Maashi, M., E. Ozcan and G. Kendall, 2014. A multi-objective hyper-heuristic based on choice function. Expert Syst. Appl., 41: 4475-4493.

McClymont, K. and E.C. Keedwell, 2011. Markov Chain Hyper-Heuristic (MCHH): An online selective hyper-heuristic for multi-objective continuous problems. Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation, July 12-16, 2011, ACM, Dublin, Ireland, ISBN:978-1-4503-0557-0, pp: 2003-2010.

Nasereddin, M., 2006. Using genetic algorithms to find weights for multiple heuristic for the stochastic resource constrained project scheduling problem. Int. J. Soft Comput., 1: 255-260.

Thamarai, I. and S. Murugavalli, 2016. Analogy based software effort estimation based on differential evolution and hybrid fuzzy logic and firefly algorithm. Asian J. Inf. Technol., 15: 1484-1493.

Topcuoglu, H.R., A. Ucar and L. Altin, 2014. A hyper-heuristic based framework for dynamic optimization problems. Appl. Soft Comput., 19: 236-251.