MISTA 2009

A self-organising hyper-heuristic framework

E. Özcan · M. Mısır · E. K. Burke

1 Introduction

Hyper-heuristics are heuristic management methodologies aiming a high level of generality in problem solving [1]. Most of the meta-heuristics or other approaches perform a search over the solution space directly, whereas, hyper-heuristics as high level strategies search the heuristic space, generated by a set of low-level heuristics. One of the hyper-heuristic frameworks makes use of multiple perturbative (improvement) heuristics in a single-point iterative search setting. At each step, one of the low level heuristics is selected and applied to the candidate solution at hand. The resulting move is either accepted or rejected based on some problem independent criteria. This study focuses on such *perturbative hyper-heuristics*.

In [3], two types of low level heuristics are identified: *mutational heuristics* and *hill climbers*. A mutational heuristic perturbs a given candidate solution in a systematic way using a stochastic component without any expectation regarding to the change in the quality of the solution. On the other hand, a hill climber returns a solution that has either better or equal quality with the input solution. [3] investigates four different types of perturbative hyper-heuristic frameworks that aim to make better use of multiple mutational and hill climbing heuristics as illustrated in Figure 1: F_A, F_B, F_C and F_D. Type A framework is the traditional framework where all low level heuristics are utilised simultaneously. A type B framework also functions similar to a type A framework. The

Ender Özcan · Edmund K. Burke
University of Nottingham, School of Computer Science & IT
The Automated Scheduling, Optimisation and Planning (ASAP) Research Group
Jubilee Campus, Wollaton Road, Nottingham
NG8 1BB, UK
E-mail: {exo,ekb}@cs.nott.ac.uk

Mustafa Mısır *
Katholieke Universiteit Leuven, Department of Computer Science
Celestijnenlaan 200A
3001 Heverlee-Leuven/BELGIUM

KaHo Sint-Lieven, IT Research Group
Gebroeders Desmetstraat 1
9000 Gent/BELGIUM
E-mail: mustafa.misir@kahosl.be

only difference is that if the heuristic selection method chooses and applies a mutational heuristic to the candidate solution, then a predefined hill climber is invoked afterwards. If the chosen heuristic is a hill climber, the second process does not take place. The resulting solution is fed into the move acceptance strategy. In type C framework, the low level heuristic set contains only mutational heuristics. A predetermined hill climber is applied following the chosen mutational heuristic. The last framework is the most general framework. Type D framework uses two hyper-heuristics successively at each iteration. The first hyper-heuristic manages a set of mutational heuristics, while the second one manages a set of hill climbers. Both hyper-heuristics can be the same or each hyper-heuristic mechanism can have its own heuristic selection and move acceptance strategy. It is also possible to design a type D framework which passes information between hyper-heuristics. The experimental results show that F_C is superior to the others including the traditional framework. Although F_D has not performed well, this might be due to the choice of hyper-heuristics which function independently and do not communicate any information with each other. Notice that F_C is a special case of F_D.

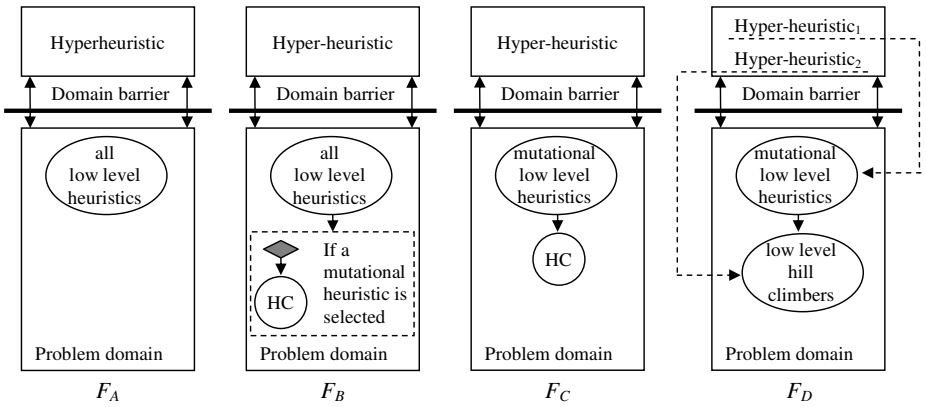

Fig. 1 Hyper-heuristic frameworks presented in [3] that make use of multiple mutational and hill climbing (HC) heuristics.

In this study, a hyper-heuristic framework of type D is investigated. It is assumed that the nature of each low level heuristics is not known. Reinforcement learning is used as a mechanism to adaptively classify low level heuristics into two categories as more likely to be mutational heuristics and hill climbers. Moreover, instead of two separate hyper-heuristics, a unified single hyper-heuristic that self organises itself to operate as a type D framework is utilised.

2 A self organising hyper-heuristic framework based on reinforcement learning

Hyper-heuristic research values machine learning techniques to perform better choices during the search process. Within the context of perturbative heuristics, these learning techniques are utilised by the heuristic selection component. The most common approach is the reinforcement learning due to its simple structure and effectiveness. One

of the significant studies on reinforcement learning based hyper-heuristics is provided by Nareyek (2004) in [2]. Depending on the performance of a given heuristic, it is either rewarded or punished. This study investigates different rates for reward and punishment as well as different heuristic selection mechanisms using the scores. This study adapts the same idea, but employs a different strategy during the heuristic selection process.

In our approach, we use +1 as a reward factor as suggested in [2]. For the punishment, three different scoring schemes are utilised:

$$RL_1 : w_i = w_i - 1 \tag{1}$$

$$RL_2 : w_i = w_i/2 \tag{2}$$

$$RL_3 : w_i = \sqrt{w_i} \tag{3}$$

Given n heuristics, these scores indicate the individual performance of each low-level heuristic. Moreover, they are used to generate two different heuristic partitions from the whole low level heuristic set. These sets are dynamically determined based on the average score of the whole low level heuristics at any given iteration:

$$avr = \sum_{\forall i} w_i/n \tag{4}$$

Heuristics having a score below the average form the set C_1 (C_1=$\{\forall i; w_i < avr\}$), while heuristics having a score above or equal to the average form the set C_2 (C_2=$\{\forall i; w_i \geq avr\}$). *Improving or Equal* (IE) is preferred as move acceptance criteria, since it is reported to be the best choice for the benchmark function optimisation in [3].

At any point of search, it is expected that the heuristics in C_2 will be hill climbers and the rest in C_1 will be the mutational heuristics. The proposed hyper-heuristic framework works in two consecutive stages by choosing a heuristic from C_1 randomly, applying it to the current solution and feeding the new solution into the move acceptance mechanism in the first stage. Then, the same procedure is followed for the C_2 heuristics in the second stage. We can categorise this hyper-heuristic as a type D framework. Three reinforcement learning based hyper-heuristics of this type are tested over 13 well-known mathematical benchmark functions (see [3] for more details). Each hyper-heuristic denoted as FD_1, FD_2 and FD_3 utilises different punishment rates as provided in Equations 1, 2 and 3, respectively.

3 Results

During the experiments, Pentium IV 3 GHz LINUX (Fedora Core 8) machines having 2 Gb memories are used. Fifty runs are performed during each test on a benchmark function. For a fair comparison between all algorithms, the experiments are terminated if the execution time exceeds 600 CPU seconds or the expected fitness (Opt) is achieved. Success rate denotes the ratio of successful runs in which the expected fitness is achieved to the total number of runs. So, for instance, if an algorithm finds the optimum solution for the given benchmark for each trial, then its success rate is 1.0.

The results are compared to the traditional reinforcement learning with maximal utility selection. In Table 1,the ranking of each hyper-heuristic based on the success rates and in case of equality based on the number of fitness evaluations are provided. 1 indicates the best hyper-heuristic, while 6 denotes the worst one. These rankings show that self organising framework improves the traditional one that embeds reinforcement

Function	Dim	Opt	RL_1	RL_2	RL_3	FD_1	FD_2	FD_3
Sphere	10	0.0	4.00	5.50	5.50	3.00	1.50	1.50
Step	10	0.0	5.00	4.00	1.00	3.00	6.00	2.00
Quartic	10	0.0	6.00	3.50	1.00	5.00	2.00	3.50
Foxhole	2	0.0	2.00	2.00	6.00	2.00	4.50	4.50
Rastrigin	10	0.0	1.00	5.00	6.00	4.00	3.00	2.00
Schewefel	10	0.0	6.00	5.00	4.00	1.00	2.00	3.00
Griewangk	10	0.0	2.00	4.50	4.50	1.00	4.50	4.50
Ackley	10	0.0	5.00	4.00	6.00	1.00	3.00	2.00
Easom	6	1.0	4.00	2.00	1.00	6.00	3.00	5.00
Schewefel's Double Sum	10	0.0	3.50	3.50	3.50	3.50	3.50	3.50
Royal Road	8	0.0	5.00	2.00	6.00	3.00	1.00	4.00
Goldberg	30	0.0	1.00	3.50	5.50	3.50	5.50	2.00
Whitley	6	0.0	5.00	6.00	3.00	2.00	4.00	1.00
		avg.	3.81	3.88	4.08	2.92	3.35	2.96

Table 1 Ranks based on success rate and average number of fitness evaluations.

learning. The subtractive punishment rate turns out to be the best choice for negative reinforcement considering the average rank of each hyper-heuristic. This study shows that type D hyper-heuristics have potential. In this study, self organisation of the heuristics is obtained by using the scores generated by reinforcement learning. It is possible to design different self organising hyper-heuristics based on different strategies.

Acknowledgements The experiments were performed at Yeditepe University, Computer Engineering Department, Istanbul, Turkey.

References

1. Burke, E.K., Hart, E., Kendall, G., Newall, J., Ross, P., Schulenburg, S.: Hyper-heuristics: An emerging direction in modern search technology. In: F. Glover, G. Kochenberger (eds.) Handbook of Metaheuristics, pp. 457–474. Kluwer (2003)
2. Nareyek, A.: Choosing search heuristics by non-stationary reinforcement learning. In: M.G.C. Resende, J.P. de Sousa (eds.) Metaheuristics: Computer Decision-Making, chap. 9, pp. 523–544. Kluwer (2003)
3. Ozcan, E., Bilgin, B., Korkmaz, E.E.: A comprehensive survey of hyperheuristics. Intelligent Data Analysis **12**(1), 1–21 (2008)