Two Frameworks for Cross-Domain Heuristic and Parameter Selection Using Harmony Search

Paul Dempster and John H. Drake

Abstract Harmony Search is a metaheuristic technique for optimizing problems involving sets of continuous or discrete variables, inspired by musicians searching for harmony between instruments in a performance. Here we investigate two frameworks, using Harmony Search to select a mixture of continuous and discrete variables forming the components of a Memetic Algorithm for cross-domain heuristic search. The first is a single-point based framework which maintains a single solution, updating the harmony memory based on performance from a fixed starting position. The second is a population-based method which co-evolves a set of solutions to a problem alongside a set of harmony vectors. This work examines the behaviour of each framework over thirty problem instances taken from six different, real-world problem domains. The results suggest that population co-evolution performs better in a time-constrained scenario, however both approaches are ultimately constrained by the underlying metaphors.

Keywords Harmony search · Hyper-heuristics · Combinatorial optimisation · Metaheuristics · Memetic algorithms

1 Introduction

Harmony Search (HS) is a population-based metaheuristic technique introduced by Geem et al [1], inspired by the improvisation process of musicians. HS is an Evolutionary Algorithm (EA) which seeks to optimise a given set of parameters,

P. Dempster(✉) · J.H. Drake
School of Computer Science, University of Nottingham Ningbo, Ningbo 315100, China
e-mail: {paul.dempster,john.drake}@nottingham.edu.cn

J.H. Drake
ASAP Research Group, School of Computer Science, University of Nottingham,
Jubilee Campus, Wollaton Road, Nottingham NG8 1BB, UK
e-mail: drakejohnh@gmail.com

J.H. Kim and Z.W. Geem (eds.), *Harmony Search Algorithm,*
Advances in Intelligent Systems and Computing 382,
DOI: 10.1007/978-3-662-47926-1_10

analogous to musicians searching for a 'harmonious' combination of musical notes. Using a short-term memory, a population of vectors representing a set of decision variables is maintained and updated over time. At each step, a new vector is generated and compared to the existing vectors within the memory based on a given quality measure, and if the new vector is deemed to be of better quality than one of the existing vectors it replaces that vector within the memory.

Over recent years, many variants of HS have been proposed and applied to combinatorial optimisation problems including educational timetabling [2], flow shop scheduling [3] and the travelling salesman problem [1]. For the interested reader, a survey of the applications HS has been applied to is provided by Manjarres et al. [4].

Cowling et al. [5] introduced the term 'hyper-heuristic' to the field of combinatorial optimisation, defining hyper-heuristics as '*heuristics to choose heuristics*'. Unlike traditional search and optimisation techniques, hyper-heuristics operate over a space of low-level heuristics or heuristic components, rather than directly over a search space of solutions. More recently, changes in research trends have led to a variety of hyper-heuristic approaches being developed for which this original definition does not provide the scope to cover. Burke et al. [6,7] offered a more general definition covering the two main classes of hyper-heuristics, selection hyper-heuristics (e.g. [8,9]) and generation hyper-heuristics (e.g. [10,11]):

> 'A hyper-heuristic is a search method or learning mechanism for selecting or generating heuristics to solve computational search problems.'

Hyper-heuristics have been applied successfully to a variety of problems such as bin packing [12], dynamic environments [13], examination timetabling [14,15], multidimensional knapsack problem [9,16], nurse scheduling [14], production scheduling [17], sports scheduling [18] and vehicle routing [19].

In this paper we investigate two frameworks for using Harmony Search to choose the components and parameter settings of a selection hyper-heuristic applied to multiple problem domains.

2 Selection Hyper-Heuristics and Cross-Domain Heuristic Search

Many optimisation problems create a search space which is too large to enumerate exhaustively to check every possible solution. A variety of heuristic and metaheuristic methods have been used previously to solve such problems. A disadvantage of these methods is the lack of flexibility to solve different types of problem instances or problem class. Typically this will result in the need to tune the proposed method each time a new type of problem is encountered. The goal of cross-domain heuristic search is to develop methods which are able to find high quality solutions consistently over multiple problem domains, operating over a search space of low-level heuristics. Hyper-heuristics, as introduced previously, are high-level search methodologies that operate at a higher level of abstraction than traditional search and optimisation techniques [20], searching a space of heuristics rather than

solutions. Single-point based selection hyper-heuristics typically rely on two components, a heuristic selection method and a move acceptance criteria [21]. Hyper-heuristics using this framework iteratively select and apply low-level heuristics to a single solution with a decision made at each step as to whether to accept the move. This process continues until some termination criteria is met.

The experiments in this paper will use the HyFlex framework [22] as a benchmark for comparison. Originally developed to support the first cross-domain heuristic search challenge (CHeSC2011) [23], HyFlex provides a common software interface with which high-level search methodologies can be implemented and compared. In addition, due to the nature of the framework, a direct comparison can be made to a large number of existing methods from the literature. The core HyFlex framework provides support for six widely studied real-world problems: MAX-SAT, one-dimensional bin packing, personnel scheduling, permutation flow shop, the travelling salesman problem and the vehicle routing problem. For each of these problem domains, a set of low-level heuristics is implemented, defining the search space within which a selection hyper-heuristic can operate. Each of these low-level heuristics belongs to one of four categories: mutation, hill-climbing, ruin-recreate or crossover. All low-level heuristics take a single solution as input and give a single solution as output, with the exception of crossover where two solutions are required for input. The number of low-level heuristics of each type varies, depending on the problem domain being considered. For the sake of our experimentation in this paper, we consider ruin-recreate heuristics within the set of mutation operators as both take a solution as input, perform some perturbation and return a new solution with no guarantee of quality. In addition to this, two continuous variables $\in [0, 1]$ are used to modify the behaviour of certain low-level heuristics. The *intensity of mutation* parameter defines the extent to which a mutation or ruin-recreate low-level heuristic modifies a given solution, with a value closer to 1 indicating a greater amount of change. The *depth of search* parameter relates to computational effort afforded to the hill-climbing heuristic, with a value closer to 1 indicating that the search will continue to a particular depth limit.

2.1 Memetic Algorithms within the HyFlex Framework

In addition to the single-point framework described above, it is also possible to implement population-based methods within the HyFlex framework. A Genetic Algorithm (GA) evolves a population of solutions to a problem, using crossover and mutation operators to recombine and modify solutions, inspired by the natural process of evolution and the concepts of selection and inheritance. A Memetic Algorithm (MA) [24] is an extension of a general GA which introduces *memes* [25] into the evolutionary process, where a meme is a 'unit of cultural transmission'. A basic MA simply introduces a hill-climbing phase into a GA after crossover and mutation have been applied. MAs are not the only approach to include an explicit hill-climbing phase into a search method. Özcan et al. [21] tested a number of frameworks for selection hyper-heuristics. Their work found that improved performance could be achieved on a set of benchmark functions by applying a

hill-climber following the application of a perturbative low-level heuristic. Ochoa et al. [26] implemented an MA variant within HyFlex, an Adaptive Memetic Algorithm, showing better performance than all of the CHeSC2011 entrants on instances of the vehicle routing problem.

3 Harmony Search-Based Frameworks for Heuristic and Parameter Selection

In this paper we test two frameworks to select the components of an MA. In this case, HS is operating as a hyper-heuristic, selecting which low-level heuristics and parameter values to use and apply to solutions to different problem domains. The first framework is based on a traditional selection hyper-heuristic, operating on a single solution, whereas the second is a population-based approach much closer to the traditional idea of an MA. The core of both of the frameworks tested relies on HS evolving vectors representing a choice of operator for each component and their associated parameter values. Each harmony corresponds to a set of three low-level heuristics, which are applied sequentially to a particular solution in the manner of an MA (i.e. crossover, followed by mutation then hill-climbing), and two associated parameter settings. The Harmony Memory (HM), of size HMS, within each framework is as follows:

$$HM = \begin{bmatrix} C^1 & M^1 & H^1 & I^1 & D^1 \\ C^2 & M^2 & H^2 & I^2 & D^2 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ C^{HMS} & M^{HMS} & H^{HMS} & I^{HMS} & D^{HMS} \end{bmatrix}$$

where *C*, *M* and *H* correspond to discrete choices of crossover, mutational and hill-climbing heuristics for the current problem domain. *I* and *D* represent continuous variables indicating the values of the *intensity of mutation* and *depth of search* parameters for these heuristics.

At each step a vector is generated ('improvised') and applied to a particular solution. The new vector is generated depending on the harmony memory consideration rate (*HMCR*) parameter, denoting the probability using existing information from memory as opposed to constructing a new vector from scratch. With probability *HMCR*, each of the individual values of decision variables in the new vector are taken randomly from one of the existing solutions in HM. In order to maintain some diversity within the search, the pitch adjustment rate (*PAR*) parameter defines the probability of modifying each of the two continuous parameters in the newly generated vector by a particular fret width (*FW*). In our experiments, the *PAR* does not affect the discrete decision variables. With probability (1 – *HMCR*) a completely new vector is generated with random valued assigned to each variable. The components of the new vector are then applied to a solution, in a

manner defined by one of the frameworks described below, with the new vector replacing the poorest quality vector in the HM if it is of better quality.

3.1 Single-Point Search Framework

This framework evolves a population of vectors, with each applied to a single solution and updated based on the performance observed. At each step a new harmony is improvised, then each harmony in HM and the newly generated harmony are all separately applied to the current solution. If the solution generated by the new harmony is better than any of the solutions generated by existing harmonies, it replaces that solution in the HM. In the case that any solution generated is better than the previous best solution, the best solution is updated to become that solution. Where a second input solution is required for crossover low-level heuristics, a solution is chosen at random from the set of solutions generated in the previous set of intermediate solutions. In the initial iteration, the second solution is generated randomly using the methods defined in the HyFlex framework. This method is similar to the *Greedy* selection hyper-heuristic of Cowling et al. [5] from their early work in selection hyper-heuristics. An overview of this framework is shown in Figure 1.

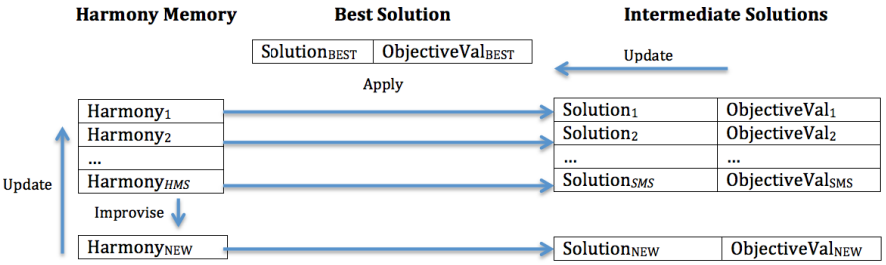

Fig. 1 Single-point search framework

3.2 Population-Based Co-evolutionary Framework

The second framework is very similar to the one proposed and applied to examination timetabling by Anwar et al. [27]. Rather than operating on a single solution, a population of solutions is maintained in a Solution Memory (SM) in addition to the population of heuristics and parameters managed using HS. In our experimentation the size of SM (Solution Memory Size (*SMS*)) and HMS are equal, with the harmony used to produce a particular solution contained at the same index in HM as the solution in SM. In the first instance, the contents of HM and SM are populated randomly, with each harmony in HM applied once to the solution in the corresponding index of SM, with random solutions taken from SM for crossover purposes. Following this initialisation period, a new harmony is improvised, either based on harmonies in memory or generated from scratch depending on whether a randomly generated variable $U(0,1)$ is less than or equal to the value of *HMCR*. The low-level heuristics in this harmony are then applied in order to a solution

from the SM as described above. In the case that the new harmony is based on an existing harmony in memory, one input solution is taken from SM at the same index as the crossover low-level heuristic was taken from HM, with a second solution (as required for crossover low-level heuristics) taken randomly. Where the harmony has been improvised randomly, two random solutions from SM are used as input for the crossover low-level heuristic. Following the application of crossover, the mutation and hill-climbing (with the parameter values for *intensity of mutation* and *depth of search*) are then applied to the resulting solution as defined by the new harmony. If the quality of the final solution is better than an existing solution in SM, the new solution replaces the existing solution in SM and the new harmony replaces the harmony at the corresponding index in HM. This general framework is shown in Figure 2.

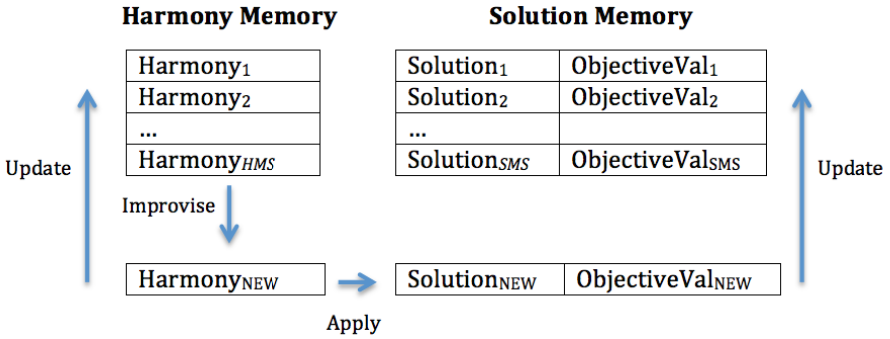

Fig. 2 Population-based co-evolutionary framework

3.3 Experimental Settings

As introduced in Section 2, the HyFlex framework will be used to compare the two proposed frameworks. We will use the 30 instances tested in the CHeSC2011 competition, with 5 instances taken from each of the 6 problem domains. All problem domains are minimization problems, with a lower objective value indicating better performance. In line with the competition, each hyper-heuristic is allowed a notional 10 minutes computation time as calculated by a benchmark provided by the organisers. Each instance is run 31 times, with the median and minimum values of all runs reported. In all of our experiments, *HMS* and *SMS* are set to 10, *HMCR* is 90%, *PAR* is 30% and *FW* is 0.2. The setting of *FW*, in particular, to 0.2 is due to the underlying nature of the HyFlex parameter values.

4 Computational Results

4.1 Direct Comparison of Single-Point and Population-Based Frameworks

Firstly we will directly compare the objective values obtained by the single-point and population-based variants described in Section 3.1 and Section 3.2. Table 1

Table 1 Median and minimum objective value obtained by Harmony Search MA variants

	Single-point		Population-based	
Instance	Median	Min	Median	Min
SAT0	**29**	20	**29**	**19**
SAT1	**57**	46	**57**	**40**
SAT2	**42**	**18**	**42**	19
SAT3	**30**	22	32	**15**
SAT4	**17**	**13**	20	14
BP0	0.076754703	0.063377656	**0.063257737**	**0.055634589**
BP1	0.011526797	0.007252416	**0.008133224**	**0.006673**
BP2	0.027774455	0.013815935	**0.014645751**	**0.012569296**
BP3	0.111261704	0.109816678	**0.10926376**	**0.108950592**
BP4	0.045656986	0.037023311	**0.029869874**	**0.023851741**
PS0	33	21	**27**	**20**
PS1	10688	10243	**9850**	**9556**
PS2	3327	3200	**3232**	**3130**
PS3	1830	1485	**1685**	**1443**
PS4	355	**320**	**350**	**320**
FS0	6287	6252	**6267**	**6237**
FS1	26847	26755	**26813**	**26754**
FS2	6367	6323	**6345**	**6303**
FS3	11436	11382	**11409**	**11336**
FS4	26658	26579	**26639**	**26534**
TSP0	48286.76001	**48194.9201**	**48194.9201**	**48194.9201**
TSP1	21308223.69	21076767.44	**20794298.01**	**20729164.65**
TSP2	6853.294645	6823.87626	**6822.783065**	**6798.088796**
TSP3	67692.9665	66570.50774	**67050.91033**	**66423.61825**
TSP4	**53699.5647**	**52272.74022**	54049.52449	52561.08631
VRP0	91485.99596	86722.503	**62722.29739**	**60850.09874**
VRP1	14395.55092	**13317.12627**	**13383.59972**	13334.93593
VRP2	201800.2381	147303.8795	**148281.8064**	**144058.3475**
VRP3	21672.79408	**20658.96394**	**21658.21568**	20658.96815
VRP4	178263.89	166947.0722	**147932.5119**	**146313.0592**

shows the median and minimum values observed over 31 runs for each of the 30 problem instances tested. The best values for each instance are marked in **bold**. From this table we can quickly see that the population-based version is performing better in the majority of cases in terms of both median and minimum objective value observed. Overall the population method achieves a better median value for 24 of the 30 instances tested and a better minimum value in 23 of the 30 instances. Whilst both methods show similar performance in the SAT instances, the objective values obtained are poor when compared to the literature standard. This will be discussed further in the following section.

4.2 Comparison to CHeSC2011 Entrants

Following the competition, the results were provided for the competition entries over a subset of the problems of all six problem domains. Methods are ranked using a scoring system inspired by the one used in Formula One motor racing (2003-2009).

Table 2 Rankings obtained by Harmony Search MA variants compared to CHeSC2011 entrants

Hyper-heuristic	Total
AdapHH	181
VNS-TW	133
ML	131.5
PHUNTER	93.25
EPH	89.75
NAHH	75
HAHA	73.75
ISEA	69
KSATS-HH	66.5
HAEA	53.5
ACO-HH	39
GenHive	36.5
DynILS	26
SA-ILS	24.25
XCJ	22.5
AVEG-Nep	21
GISS	16.75
SelfSearch	7
SP-MA	**6**
MCHH-S	4.75
Ant-Q	0

Hyper-heuristic	Total
AdapHH	173.5
VNS-TW	130.5
ML	129.5
PHUNTER	87.75
EPH	83.75
HAHA	71.083
NAHH	71
POP-MA	**65.5**
KSATS-HH	64.5
ISEA	63
HAEA	47.833
ACO-HH	36.33
GenHive	32.5
SA-ILS	23.25
XCJ	21.5
AVEG-Nep	21
DynILS	21
GISS	16.75
SelfSearch	5
MCHH-S	4.75
Ant-Q	0

For each of the 30 problem instances, the best performing hyper-heuristic of those currently being compared is awarded 10 points, the second best 8 points with each further method allocated 6, 5, 4, 3, 2, 1 and 0 points respectively. In the case there are more than 8 methods being compared, all methods ranked > 8th are awarded 0 points. The two HS hyper-heuristic variants can be compared directly against the CHeSC2011 entrants using this scoring system. Table 2 shows the relative ranking of each variant against the 20 entrants to CHeSC2011, where SP-MA is the single-point framework and POP-MA is the population-based variant.

When compared to the CHeSC2011 entrants, the population-based framework scores 65.5 points finishing 8th overall, with the single point approach scoring 6 points and finishing 19th out of 21. As this is a relative ranking system, it shows that not only is POP-MA outperforming SP-MA, in at least some problem domains or instances it is outperforming some of the leading entrants to the competition, taking 7.5 points away from the winning hyper-heuristic AdapHH. A breakdown of the points per problem domain scored by each method is given in Table 3.

Table 3 Formula One points obtained by each method in each problem domain

Domain	SP-MA	POP-MA
SAT	0	0
Bin Packing	0	0
Personnel Scheduling	0	**4.5**
Flow Shop	3	**16**
TSP	3	**30**
VRP	0	**15**

As can be seen, neither framework performed well in SAT or Bin Packing. The population-based framework performed particularly well in TSP, coming 3rd in this domain and beating all other hyper-heuristics in one particular instance. Given that the best results of the single-point framework also came in TSP, it suggests that the heuristics supplied for that domain produce good solutions when combined via MA. Figure 3 plots the number of points scored by each of the twenty competition entrants to CHeSC2011 and POP-MA for the TSP. An interesting observation when using this framework is that the number of heuristics that could be applied per run within the time limit varied dramatically. This is due to both the nature of the heuristics in a particular domain, and the evolved values of the heuristic parameters (e.g., between 415000 and 450 heuristic applications were observed for the first two TSP instances). Since HS requires a new harmony to be improvised each iteration and MA requires all three of cross-over, mutation, and hill-climbing, the high-level of retention of solutions by the population-based approach (keeping the best HMS − 1 solutions rather than throwing away the HMS − 1 worst as with single-point) may be the reason for the comparatively good Personnel Scheduling score, as this domain typically had low heuristic application

counts. It is also worth noting that as this ranking system only rewards the top 8 entrants, a score of 0 does not necessarily indicate that a method is one of the worst of the 21 being compared, only that it is not one of the top hyper-heuristics for that instance.

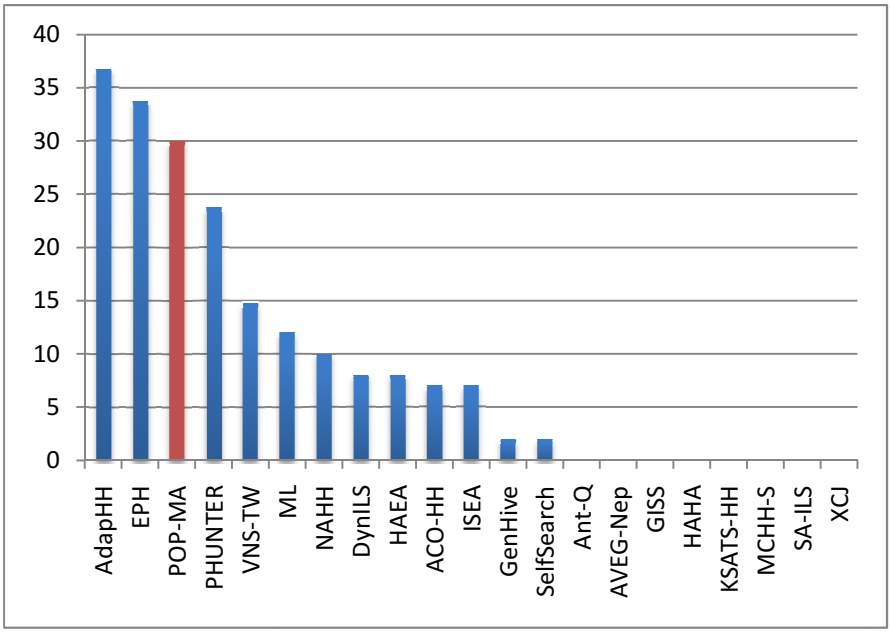

Fig. 3 Formula One points scored in the TSP problem domain by POP-MA and CHeSC2011 entrants

5 Conclusions and Future Work

In this work we have presented two frameworks, using Harmony Search (HS) to select the components and parameters for a Memetic Algorithm (MA) operating over multiple problem domains. The first framework is similar to a traditional greedy single-point selection hyper-heuristic, applying each set of low-level heuristic and parameter combinations in the Harmony Memory (HM) to a current best-of-run solution before updating HM and the best-of-run solution if an improvement in solution quality is found. The second is a population-based approach which co-evolves a set of solutions to the problem at the same time as evolving a set of harmonies representing the operator and parameter choices of an MA applied to the solutions. Our results indicate that the population-based approach is able to achieve significantly better results than the single-point approach. This method performed particularly well on instances of the Travelling Salesman Problem, outperforming all but two of the twenty entrants to CHeSC2011.

We intend to improve on this work in future by extending the framework in a number of ways. Currently the representation of each harmony is very rigid, due to the decision of conforming to the structure of an MA. More flexible representations are possible, allowing for a greater combination of low-level heuristics and more freedom with the order in which they are applied. Another constraining factor in the population-based framework is the decision to limit the Solution Memory Size and Harmony Memory Size to the same value. Future work will look at dynamic strategies for managing the length of these two components.

References

1. Geem, Z.W., Kim, J.H., Loganathan, G.V.: New heuristic optimization algorithm: Harmony search. Simulation **76**(2), 60–68 (2001)
2. Al-Betar, M.A., Khader, A.T.: A harmony search algorithm for university course timetabling. Ann. Oper. Res. **194**(1), 3–31 (2012)
3. Wang, L., Pan, Q.K., Tasgetiren, M.F.: Minimizing the total flow time in a flow shop with blocking by using hybrid harmony search algorithms. Expert Syst. Appl. **37**(12), 7929–7936 (2010)
4. Manjarres, D., Landa-Torres, I., Gil-Lopez, S., Ser, J.D., Bilbao, M., Salcedo-Sanz, S., Geem, Z.: A survey on applications of the harmony search algorithm. Eng. Appl. Artif. Intell. **26**(8), 1818–1831 (2013)
5. Cowling, P.I., Kendall, G., Soubeiga, E.: A hyperheuristic approach to scheduling a sales summit. In: Burke, E., Erben, W. (eds.) PATAT 2000. LNCS, vol. 2079, pp. 176–190. Springer, Heidelberg (2001)
6. Burke, E.K., Hyde, M., Kendall, G., Ochoa, G., Özcan, E., Woodward, J.: A classification of hyper-heuristics approaches. In: Handbook of Metaheuristics, 2nd edn., pp. 49–468 (2010)
7. Burke, E.K., Hyde, M., Kendall, G., Ochoa, G., Özcan, E., Qu, R.: Hyper-heuristics: A survey of the state of the art. J. Oper. Res. Soc. **64**(12), 1695–1724 (2013)
8. Drake, J.H., Özcan, E., Burke, E.K.: An improved choice function heuristic selection for cross domain heuristic search. In: Coello, C.A., Cutello, V., Deb, K., Forrest, S., Nicosia, G., Pavone, M. (eds.) PPSN 2012, Part II. LNCS, vol. 7492, pp. 307–316. Springer, Heidelberg (2012)
9. Drake, J.H., Özcan, E., Burke, E.K.: Modified choice function heuristic selection for the multidimensional knapsack problem. In: Sun, H., Yang, Chin-Yu., Lin, C.-W., Pan, J.-S., Snasel, V., Abraham, A. (eds.) Genetic and Evolutionary Computing. AISC, vol. 329, pp. 225–234. Springer, Heidelberg (2015)
10. Drake, J.H., Hyde, M., Ibrahim, K., Özcan, E.: A genetic programming hyper-heuristic for the multidimensional knapsack problem. Kybernetes **43**(9–10), 1500–1511 (2014)
11. Drake, J.H., Kililis, N., Özcan, E.: Generation of VNS components with grammatical evolution for vehicle routing. In: Krawiec, K., Moraglio, A., Hu, T., Etaner-Uyar, A., Hu, B. (eds.) EuroGP 2013. LNCS, vol. 7831, pp. 25–36. Springer, Heidelberg (2013)
12. López-Camacho, E., Terashima-Marín, H., Ross, P.: A hyper-heuristic for solving one and two-dimensional bin packing problems. In: Proceedings of the 13th Annual Conference Companion on Genetic and Evolutionary Computation, pp. 257–258 (2011)
13. Kiraz, B., Uyar, A.S., Özcan, E.: Selection hyper-heuristics in dynamic environments. J. Oper. Res. Soc. **64**(12), 1753–1769 (2013)

14. Burke, E.K., Kendall, G., Soubeiga, E.: A tabu-search hyper-heuristic for timetabling and rostering. J. Heuristics **9**(6), 451–470 (2003)
15. 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)
16. Drake, J.H., Özcan, E., Burke, E.K.: A case study of controlling crossover in a selection hyper-heuristic framework using the multidimensional knapsack problem. Evolutionary computation (2015)
17. Fisher, H., Thompson, G.: Probabilistic learning combinations of local job-shop scheduling rules. In: Factory Scheduling Conference, Carnegie Institute of Technology (1961)
18. Gibbs, J., Kendall, G., Özcan, E.: Scheduling english football fixtures over the holiday period using hyper-heuristics. In: Schaefer, R., Cotta, C., Kołodziej, J., Rudolph, G. (eds.) PPSN XI. LNCS, vol. 6238, pp. 496–505. Springer, Heidelberg (2010)
19. Garrido, P., Castro, C.: Stable solving of cvrps using hyperheuristics. In: Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation, pp. 255–262 (2009)
20. Burke, E., Kendall, G., Newall, J., Hart, E., Ross, P., Schulenburg, S.: Hyper-heuristics: an emerging direction in modern search technology. International series in operations research and management science, 457–474 (2003)
21. Özcan, E., Bilgin, B., Korkmaz, E.E.: Hill climbers and mutational heuristics in hyperheuristics. In: Runarsson, T.P., Beyer, H.-G., Burke, E.K., Merelo-Guervós, J.J., Whitley, L., Yao, X. (eds.) PPSN 2006. LNCS, vol. 4193, pp. 202–211. Springer, Heidelberg (2006)
22. Ochoa, G., Hyde, M., Curtois, T., Vazquez-Rodriguez, J. A., Walker, J., Gendreau, M., et al.: Hyflex: a benchmark framework for cross-domain heuristic search. In: Evolutionary Computation in Combinatorial Optimization, pp. 136–147 (2012)
23. Burke, E.K., Gendreau, M., Hyde, M., Kendall, G., McCollum, B., Ochoa, G., Parkes, A.J., Petrovic, S.: The cross-domain heuristic search challenge – an international research competition. In: Coello, C.A. (ed.) LION 2011. LNCS, vol. 6683, pp. 631–634. Springer, Heidelberg (2011)
24. Moscato, P.: On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms. Caltech concurrent computation program. C3P Report, 826 (1989)
25. Dawkins, R.: The Selfish Gene. Oxford University Press, Oxford (2006)
26. Ochoa, G., Walker, J., Hyde, M., Curtois, T.: Adaptive evolutionary algorithms and extensions to the hyflex hyper-heuristic framework. In: Coello, C.A., Cutello, V., Deb, K., Forrest, S., Nicosia, G., Pavone, M. (eds.) PPSN 2012, Part II. LNCS, vol. 7492, pp. 418–427. Springer, Heidelberg (2012)
27. Anwar, K., Khader, A.T., Al-Betar, M.A., Awadallah, M.: Harmony search-based hyper-heuristic for examination timetabling. In: 2013 IEEE 9th International Colloquium on Signal Processing and its Applications (CSPA), pp. 176–181 (2013)