Hybridisation of Evolutionary Algorithms through Hyper-heuristics for Global Continuous Optimisation

Eduardo Segredo[1], Eduardo Lalla-Ruiz[2],
Emma Hart[1], Ben Paechter[1], and Stefan Voß[2]

[1] School of Computing
Edinburgh Napier University
Edinburgh, Scotland, UK
{e.segredo,e.hart,b.paechter}@napier.ac.uk
[2] Institute of Information Systems
University of Hamburg
Hamburg, Germany
{eduardo.lalla-ruiz,stefan.voss}@uni-hamburg.de

Abstract. Choosing the correct algorithm to solve a problem still remains an issue 40 years after the *Algorithm Selection Problem* was first posed. Here we propose a hyper-heuristic which can apply one of two meta-heuristics at the current stage of the search. A scoring function is used to select the most appropriate algorithm based on an estimate of the improvement that might be made by applying each algorithm. We use a differential evolution algorithm and a genetic algorithm as the two meta-heuristics and assess performance on a suite of 18 functions provided by the *Generalization-based Contest in Global Optimization* (GENOPT). The experimental evaluation shows that the hybridisation is able to provide an improvement with respect to the results obtained by both the differential evolution scheme and the genetic algorithm when they are executed independently. In addition, the high performance of our hybrid approach allowed two out of the three prizes available at GENOPT to be obtained.

Keywords: global search; differential evolution; genetic algorithm; global continuous optimisation; hyper-heuristic

1 Introduction

A significant amount of real-world applications requires finding global optima over continuous decision spaces. Examples from diverse domains such as economics and finance [28], circuit design [21], control theory [1], chemistry [23], and electricity [26], among others, highlight the importance of properly addressing them in order to provide satisfactory solutions. Due to this, the development of efficient algorithms has been of increasing interest for researchers, also accompanied by the urgency from the side of practitioners requiring high-quality feasible and fast solutions for their difficult problems at hand.

In this context, various are the approaches that have been recently proposed for non-differentiable global optimisation. For instance, in [13], a predictive approach to the reproduction phase of new individuals for a well-known meta-heuristic was proposed. Another example is given by [14], where a derivative-free global heuristic, which deals with constraints by static and dynamic penalty function techniques, was presented. Finally, a modification over an existing exact penalty algorithm for making it derivative-free, which in addition makes use of a local search procedure, was introduced in [7].

Evolutionary Computation (EC) is a relevant field with many applications within global optimisation [5,25]. Its main goal is to study, develop, and analyse algorithms following the biological notion of evolution within the Darwinian principles. The above has motivated the development of a wide variety of algorithms. In this regard, some of the most frequently used methods, which belong to the family of *Evolutionary Algorithms* (EAs), are *Genetic Algorithms* (GAs) [10], due to their easy and flexible implementation, as well as their exhibited performance. Furthermore, during the last two decades, another EA called *Differential Evolution* (DE), proposed in [22], has been successfully applied not only to benchmark problems but also to several real-world applications [4].

Another field of research that has gained a significant popularity during last years is that of *Hyper-heuristics* (HH). A HH can be defined as a search method or a learning mechanism for *selecting* or *generating* meta-heuristics or tailored heuristics to solve computational search problems [2]. Therefore, they function at a higher level of abstraction when compared to meta-heuristics and heuristics, and usually have no knowledge about the domain of the problem at hand. In this context, HH based on selection try to address the *Algorithm Selection Problem* [15] by iteratively identifying and selecting the most promising low-level meta-heuristics or heuristics, from a set of candidates, for solving a particular instance of an optimisation problem [3]. This can be done by means of a scoring function that is used for assessing the performance of each low-level approach.

In this work, we propose a hybridised EA that uses a selection-based HH to address the set of global continuous optimisation problems provided for the *Generalization-based Contest in Global Optimization* (GENOPT)[1] organised in the field of the *Learning and Intelligent Optimization Conference* (LION 10). The HH selects the most suitable meta-heuristic to be applied at the current stage of the search procedure, choosing between a DE scheme and a GA. If both algorithms are applied in isolation, then for some problems, DE is the best performing approach, with the GA failing to converge to high-quality solutions, while for other instances, the opposite situation is observed. By combining the EAs by means of a HH, we are able to produce a more powerful approach that overcomes the weaknesses of the individual algorithms on the majority of considered problems.

The remainder of this paper is organised as follows. Section 2 describes our hybridisation of EAs through the use of a HH. Section 3 describes the experimental evaluation and provides a discussion of the results obtained. Finally, Section 4

[1] The manifesto of the contest, including its instructions and rules, can be found in the following URL: `http://genopt.org/genopt.pdf`.

draws the main conclusions extracted from the work and provides several lines for further research.

2 Hybridisation of evolutionary algorithms

This section is devoted to the description of the hybridisation, through the use of a HH (Section 2.4), of two meta-heuristics. Section 2.1 introduces an adaptive version of DE, while Section 2.2 presents the implementation of the GA applied. Additionally, with the aim of increasing the convergence speed of the whole optimisation scheme, a *Global Search* (GS) procedure is described in Section 2.3, which is incorporated into both meta-heuristics.

2.1 Adaptive differential evolution

DE is a stochastic direct search method particularly suited for continuous global optimisation [22]. In DE, the decision variables of a given problem are defined by a vector $\boldsymbol{X} = [x_1, x_2, \ldots, x_i, \ldots, x_D]$, being D the number of decision variables or the dimensionality of the problem, and every x_i a real number. The quality of each vector $\boldsymbol{X}$ is given by the objective function $f(\boldsymbol{X})(f : \Omega \subseteq \mathbb{R}^D \rightarrow \mathbb{R})$. The goal of global optimisation, considering a minimisation problem, is thus to find a vector $\boldsymbol{X}* \in \Omega$ where $f(\boldsymbol{X}*) \leq f(\boldsymbol{X})$ holds for all $\boldsymbol{X} \in \Omega$. In the particular case of box-constrained optimisation problems, the feasible region Ω is defined by particular values for the lower (a_i) and upper (b_i) bounds of each variable, *i.e.* $\Omega = \prod_{i=1}^{D}[a_i, b_i]$.

In this work, we apply an adaptive version of the approach DE/current-to-pbest/1/bin, which uses JADE [27] as the control scheme. We selected this variant since it showed to be one of the best exploitative schemes in [19]. JADE is responsible for adapting the values of the *mutation scale factor* F and the *crossover rate* CR of DE, which will be introduced in the following lines.

The operation of this DE scheme is as follows. First of all, a population P with NP individuals ($P = [\boldsymbol{X}_1, \boldsymbol{X}_2, \ldots, \boldsymbol{X}_j, \ldots, \boldsymbol{X}_{NP}]$), also called vectors in the scope of DE, is initialised by using a particular strategy. Each individual comprises D decision variables. The value of the decision variable i belonging to the individual $\boldsymbol{X}_j$ is denoted by $x_{j,i}$. Then, successive iterations are evolved by executing the following steps, until a stopping criterion is satisfied. For each vector in the current population, referred to as *target vector* ($\boldsymbol{X}_j$), a new *mutant vector* ($\boldsymbol{V}_j$) is created using a *mutant vector generation strategy*. In our case, we apply the current-to-pbest/1 scheme. Any vector in the population different from the target vector is randomly selected as the *base vector*. The mutant vector $\boldsymbol{V}_j$ for target vector $\boldsymbol{X}_j$ is thus created as shown in Eq. 1, where r_1 and r_2 are mutually exclusive integers different from the index j chosen at random from the range $[1, NP]$. Moreover, the individual $\boldsymbol{X}_{r_3}$ is randomly selected from the fittest $p \times 100\%$ individuals. Another parameter K is also introduced, but in order to facilitate the parameterisation of the whole scheme, $K = F$ is usually considered, with F the mutation scale factor allowing the exploration and exploitation abilities of DE to be balanced.

$$\boldsymbol{V}_j = \boldsymbol{X}_j + K \times (\boldsymbol{X}_{r_3} - \boldsymbol{X}_j) + F \times (\boldsymbol{X}_{r_1} - \boldsymbol{X}_{r_2}) \quad (1)$$

After applying the mutant vector generation strategy, the mutant vector is combined with the target vector to generate a *trial vector* ($\boldsymbol{U}_j$) through a crossover operator. The combination of the mutant vector generation strategy and the crossover operator is usually referred to as the *trial vector generation strategy*. The most commonly applied operator for combining the target and mutant vectors, and the one considered herein, is the *binomial crossover* (*bin*). The crossover operation is controlled by means of the crossover rate CR. The binomial crossover generates a trial vector as shown in Eq. 2. A uniformly distributed random number in the range $[0, 1]$ is given by $rand_{j,i}$, and $i_{rand} \in [1, 2, ..., D]$ is an index selected in a random way that ensures that at least one variable is propagated from the mutant vector to the trial one. For the remaining cases, the probability of the variable being inherited from the mutant vector is CR. Otherwise, the variable of the target vector is considered.

$$u_{j,i} = \begin{cases} v_{j,i} \; if \; (rand_{j,i} \; \leq \; CR \; or \; i \; = \; i_{rand}) \\ x_{j,i} \; otherwise \end{cases} \quad (2)$$

The trial vector generation strategy, as described above, might generate vectors outside the feasible region Ω. One of the most widely used schemes is based on randomly reinitialising the infeasible values in their corresponding feasible ranges, and it is the one applied herein. After generating NP trial vectors, each one is compared against its corresponding target vector. For each pair, the one that minimises the objective function is selected to survive. In case of a tie, in our version the trial vector survives. Finally, the GS depicted in Section 2.3 is applied to the surviving population.

2.2 Genetic algorithm

The other approach we selected for our hybridisation is a generational GA with elitism preservation. This was selected as it has previously been demonstrated to be the best performing mono-objective approach when solving continuous optimisation problems with different dimensions [20]. The operation of this algorithm follows the typical scheme of a GA. First of all, an initial population with NP individuals is randomly generated. Then, for each generation, $NP-1$ offspring are created. Parents are selected by using the well-known *Binary Tournament* [8], while offspring are obtained by applying the *Uniform Mutation* operator [8] and the *Simulated Binary Crossover* operator [6], with mutation and crossover rates p_m and p_c, respectively. Afterwards, during the replacement stage, all parents, except the fittest one, are discarded, and they are replaced by the generated offspring. Finally, the last step of the algorithm is the application of the GS described in Section 2.3 to the surviving population. The above process is repeated until a given stopping criterion is achieved. In order to complete the definition of this GA, we should note that individuals are represented by a vector of D real numbers, being D the number of decision variables of the problem considered.

2.3 Global search procedure

In order to address potential slow convergence in both DE and GA that arises when addressing difficult problems, and to improve the quality of the solutions provided, a GS procedure based on the one proposed in [11], is applied to both algorithms. It is defined as follows. Given an individual $\boldsymbol{X}_k$ randomly selected from the current population, a new individual $\boldsymbol{V}$ is generated by means of Eq. 3,

$$\boldsymbol{V} = a_1 \times \boldsymbol{X}_k + a_2 \times \boldsymbol{X}_{Best} + a_3 \times (\boldsymbol{X}_{r_1} - \boldsymbol{X}_{r_2}), \tag{3}$$

with a_1, a_2, and a_3 being three numbers randomly selected from the range $[0, 1]$, and for which the condition $a_1 + a_2 + a_3 = 1$ is satisfied. $\boldsymbol{X}_{Best}$ is the best individual in the current population, *i.e.* the one with the lowest objective value, and $\boldsymbol{X}_{r_1}$ and $\boldsymbol{X}_{r_2}$ represent two different individuals randomly selected from the current population. We should note that indexes k, r_1, and r_2 are mutually exclusive. Once the new individual $\boldsymbol{V}$ is generated, it is evaluated and compared to the individual $\boldsymbol{X}_k$. In case $f(\boldsymbol{V}) < f(\boldsymbol{X}_k)$, $\boldsymbol{V}$ replaces $\boldsymbol{X}_k$ in the current population, *i.e.* $\boldsymbol{X}_k = \boldsymbol{V}$, and the GS starts another iteration for trying to improve $\boldsymbol{X}_k$. Otherwise, individual $\boldsymbol{V}$ is discarded and the GS is stopped. The main novelty in our work is that the GS is iteratively applied to individual $\boldsymbol{X}_k$ until it cannot be improved anymore. This contrasts to earlier work in [11] in which the GS is only applied once to individual $\boldsymbol{X}_k$.

2.4 Hyper-heuristic

A variant of the selection HH firstly proposed in [24] is used to select between the two aforementioned EAs. The said variant was proposed and has been successfully applied by the authors in previous work [16,17,20]. It is based on using a scoring and a selection strategy for choosing the most suitable low-level configuration. Once a low-level configuration is selected, only that strategy is executed until a local stopping criterion is achieved. When this happens, another low-level configuration is selected and executed. The final population of the last low-level configuration becomes the initial population of the new low-level configuration. This process continues until a global stopping criterion is satisfied. In the particular case of the current work, a fixed number of evaluations, established by the GENOPT rules, is considered as the global stopping criterion.

The low-level configuration that must be executed is selected as follows. First, the *scoring strategy* assigns a score to each of the two EAs. This score estimates the improvement that each configuration might achieve starting from the current population. Larger values are assigned to more promising approaches, based on their historical performance. To calculate this estimate, the previous improvements in the objective value achieved by each configuration are used. The improvement γ is defined as the difference, in terms of the objective value, between the best individual found so far, and the best initial individual. Given a configuration $conf$ that has been executed j times, the score $s(conf)$ is calculated as a weighted average of its last k improvements. This is shown in Eq. 4, where $\gamma[conf][j-i]$ represents the improvement achieved by the configuration $conf$ in

execution number $j-i$. The adaptation level of HH, *i.e.* the amount of historical knowledge considered to perform its decisions, can be varied depending on the value of k. Finally, the weighted average assigns a greater importance to the most recent executions, with the aim of better adapting decisions to the current stage of the search procedure, thus discarding too old information.

$$s(conf) = \frac{\sum_{i=1}^{min(k,j)} (min(k,j) + 1 - i) \cdot \gamma[conf][j-i]}{\sum_{i=1}^{min(k,j)} i} \tag{4}$$

The HH is elitist, namely, it selects the low-level configuration that maximises the score $s(conf)$. However, some selections are randomly performed by following a uniform distribution: this is tuned by means of a parameter β, which represents the minimum selection probability that should be assigned to each low-level configuration. If n_h is the number of low-level configurations involved, then a random selection is performed in $\beta \cdot n_h$ percentage of the cases.

3 Experimental evaluation

This section is focused on describing the experiments conducted with the optimisation scheme introduced in Section 2.

Experimental method The EAs, as well as the HH framework, were implemented using the *Meta-heuristic-based Extensible Tool for Cooperative Optimisation* (METCO) [12]. Tests were run on a Debian GNU/Linux computer with four AMD® Opteron™ processors (model number 6164 HE) at 1.7 GHz and 64 GB RAM. Since all experiments used stochastic algorithms, each execution was repeated 100 times with different initial seeds. With respect to the former, comparisons between algorithms were carried out by applying the following statistical analysis [18]. First, a *Shapiro-Wilk test* was performed to check whether the values of the results followed a normal (Gaussian) distribution. If so, the *Levene test* checked for the homogeneity of the variances. If the samples had equal variance, an ANOVA *test* was done. Otherwise, a *Welch test* was performed. For non-Gaussian distributions, the non-parametric *Kruskal-Wallis* test was used. For all tests, a significance level $\alpha = 0.05$ was considered.

Problem set Experiments were carried out using the set of continuous optimisation problems proposed for the GENOPT. The set is composed of three families of functions with different features, and a particular function is defined by its identifier. For the contest, 6 functions created by the GKLS generator [9] (f_1–f_6), 6 conditioned transforms of classical benchmarks (f_7–f_{12}), and 6 composite functions (f_{13}–f_{18}), were proposed. Functions f_1–f_{12} were defined by identifiers 0–11, while functions f_{13}–f_{18} were defined by identifiers

Table 1: Parameterisation of the genetic algorithm

Parameter	Value	Parameter	Value
Stopping criterion	$1 \cdot 10^6$ evals.	Mutation rate (p_m)	$1/D$
Population size (NP)	5	Crossover rate (p_c)	1

Table 2: Parameterisation of the differential evolution scheme

Parameter	Value	Parameter	Value
Stopping criterion	$1 \cdot 10^6$ evals.	Mutation scale factor (F)	JADE
Population size (NP)	32	Crossover rate (CR)	JADE
% of best individuals (p)	0.1		

Table 3: Parameterisation of the hyper-heuristic

Parameter	Value	Parameter	Value
Local stopping criterion	$1.2 \cdot 10^4$ evals.	Minimum selection rate (β)	0.1
Low-level configs. (n_h)	2	Historical knowledge (k)	5

1586038869–1586038874. Initial seeds were fixed by the GENOPT organisation to values 1586038869–1586038968. Finally, following the instructions given for the contest, in the current work, for those functions with an even identifier, the number of decision variables D was fixed to 10. For the remaining functions, 30 decision variables were considered.

Parameters Tables 1 and 2 show the parameterisation for the GA and DE, respectively. Parameter values for both EAs were selected based on previous knowledge of the authors [19,20]. However, in order to fix parameter values for HH, different parameterisations were considered, which did not present statistically significant differences among them. The above means that HH is robust from the point of view of its parameters, since altering them is not going to significantly affect the performance of the whole optimisation scheme. Table 3 shows the particular configuration of HH that we applied for the set of functions considered. Regarding the number of low-level configurations n_h, we should note that different values were also tested, by taking into account different parameterisations for DE and GA as the candidate set of HH. Nevertheless, the usage of more than two low-level configurations, *i.e.* $n_h > 2$, started to degrade somewhat the performance of the whole optimisation scheme. The reader should recall that HH makes some random decisions. If some candidate configurations do not perform properly, some function evaluations might be lost due to the random selection of one of those configurations, with the consequent decrease in performance of the whole search procedure. This is the main reason why we selected only two low-level configurations, one based on DE and the other one based on GA.

Table 4 shows, for each considered problem, a statistical comparison among HH and each of both EAs executed independently. Particularly, it shows if HH statistically outperformed DE or GA (↑), if HH was statistically outperformed by DE

Table 4: Statistical comparison between HH and EAs considering problems f_1–f_{18}

f	Alg.	p-value	Dif.	f	Alg.	p-value	Dif.	f	Alg.	p-value	Dif.
f_1	DE	1.53e-26	↑	f_2	DE	2.51e-28	↑	f_3	DE	1.81e-29	↑
	GA	6.69e-12	↑		GA	1.38e-10	↑		GA	1.28e-12	↑
f_4	DE	2.69e-21	↑	f_5	DE	2.19e-28	↑	f_6	DE	2.23e-18	↑
	GA	4.75e-05	↑		GA	3.07e-13	↑		GA	1.25e-03	↑
f_7	DE	2.40e-16	↑	f_8	DE	1.48e-01	↔	f_9	DE	8.09e-02	↔
	GA	2.52e-34	↑		GA	2.52e-34	↑		GA	3.07e-34	↑
f_{10}	DE	5.53e-39	↑	f_{11}	DE	6.94e-14	↑	f_{12}	DE	7.70e-12	↑
	GA	5.54e-39	↑		GA	2.52e-34	↑		GA	2.52e-34	↑
f_{13}	DE	9.98e-09	↑	f_{14}	DE	3.27e-02	↑	f_{15}	DE	1.98e-05	↓
	GA	2.52e-34	↑		GA	2.66e-33	↑		GA	2.52e-34	↑
f_{16}	DE	5.18e-34	↑	f_{17}	DE	2.87e-30	↑	f_{18}	DE	5.17e-05	↑
	GA	2.52e-34	↑		GA	2.52e-34	↑		GA	2.52e-34	↑

or GA (↓), and cases for which statistically significant differences did not appear between HH and DE or GA (↔). We should note that HH statistically outperforms a particular EA if there exist statistically significant differences between them, *i.e.* if the p-value is lower than $\alpha = 0.05$, and if at the same time, the *Vargha Delaney effect size* between HH and the given EA is lower than 0.5, since we are dealing with minimisation problems.

It can be observed that HH was statistically better in 33 out of 36 statistical comparisons. In 15 out of 18 problems, HH was able to provide statistically better solutions than DE and GA. For problems f_8 and f_9, DE did not present statistically significant differences with HH, but the latter was able to statistically outperform GA. Finally, considering the problem f_{15}, HH was statistically outperformed by DE. Bearing the above in mind, the superiority of HH when compared to DE and GA executed independently is clear. Using HH removes the issue of algorithm selection from the user, with the HH autonomously selecting the most appropriate algorithm at the current stage of the search for a given instance, and enabling a hybridisation of both algorithms.

Additionally, HH is able to provide even better solutions than those obtained by DE or GA executed independently for most of the considered problems, thus showing that it is able to properly combine the benefits of both EAs for solving global continuous optimisation problems.

4 Conclusions and future work

In this work, a hyper-heuristic solution approach, HH, enabling hybridisations of EAs for solving global continuous optimisation problems was proposed. The approach hybridised a differential evolution DE, and genetic algorithm GA. Furthermore, it included the use of a stochastic global search following the selection of the surviving population. The HH selects the most appropriate method to use at each point based on a scoring function that estimates potential improvement.

The method is evaluated using a set of continuous optimisation problems proposed for the *Generalization-based Contest in Global Optimization* (GENOPT).

The computational results show that the use of our proposed hyper-heuristic framework leads to an overall enhancement when compared to the evolutionary algorithms executed in isolation. This highlights the capability of HH for switching the best evolutionary algorithm along the search. Moreover, in the majority of the cases, the improvement exhibited by HH goes further the best performing EA for each given problem, suggesting its use instead of using the embedded methods independently. Finally, it is worth mentioning that the high performance of our hybridisation through HH was recognised with two out of the three prizes available at the GENOPT.

On the basis of the findings presented in this paper, the next stage of our research will be focused on extending the numerical experimentation including the assessment of the different parameters of HH and its integrated EAS, as well as studying the performance of HH with additional algorithms and/or problems. Another promising line of research would be to analyse the impact that different scoring functions have over the performance of the whole optimisation scheme.

References

1. Bingül, Z., Karahan, O.: A Fuzzy Logic Controller tuned with PSO for 2 DOF robot trajectory control. Expert Systems with Applications 38(1), 1017–1031 (2011)
2. Burke, E.K., Gendreau, M., Hyde, M., Kendall, G., Ochoa, G., Ozcan, E., Qu, R.: Hyper-heuristics: A survey of the state of the art. Journal of the Operational Research Society 64(12), 1695–1724 (Dec 2013)
3. Burke, E., Kendall, G., Newall, J., Hart, E., Ross, P., Schulenburg, S.: Hyper-heuristics: An emerging direction in modern search technology. In: Glover, F., Kochenberger, G.A. (eds.) Handbook of Metaheuristics, International Series in Operations Research & Management Science, vol. 57, pp. 457–474. Springer US (2003)
4. Das, S., Mullick, S.S., Suganthan, P.: Recent advances in differential evolution – An updated survey. Swarm and Evolutionary Computation 27, 1 – 30 (2016)
5. Dasgupta, D., Michalewicz, Z.: Evolutionary algorithms in engineering applications. Springer Science & Business Media (2013)
6. Deb, K., Agrawal, R.B.: Simulated binary crossover for continuous search space. Complex Systems 9, 115–148 (1995)
7. Di Pillo, G., Lucidi, S., Rinaldi, F.: A derivative-free algorithm for constrained global optimization based on exact penalty functions. Journal of Optimization Theory and Applications 164(3), 862–882 (2015)
8. Eiben, A.E., Smith, J.E.: Introduction to Evolutionary Computing. Springer Berlin Heidelberg, Berlin, Heidelberg (2015)
9. Gaviano, M., Kvasov, D.E., Lera, D., Sergeyev, Y.D.: Algorithm 829: Software for generation of classes of test functions with known local and global minima for global optimization. ACM Trans. Math. Softw. 29(4), 469–480 (Dec 2003)
10. Goldberg, D.E.: Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley Publishing Company, Reading, Massachusetts (1989)
11. Guo, Z., Liu, G., Li, D., Wang, S.: Self-adaptive differential evolution with global neighborhood search. Soft Computing pp. 1–10 (2016)

12. León, C., Miranda, G., Segura, C.: METCO: A Parallel Plugin-Based Framework for Multi-Objective Optimization. International Journal on Artificial Intelligence Tools 18(4), 569–588 (2009)
13. Li, Y.L., Zhan, Z.H., Gong, Y.J., Chen, W.N., Zhang, J., Li, Y.: Differential evolution with an evolution path: A DEEP evolutionary algorithm. IEEE Transactions on Cybernetics 45(9), 1798–1810 (2015)
14. Liu, J., Teo, K.L., Wang, X., Wu, C.: An exact penalty function-based differential search algorithm for constrained global optimization. Soft Computing 20(4), 1305–1313 (2016)
15. Rice, J.R.: The Algorithm Selection Problem. Advances in Computers, vol. 15, pp. 65 – 118. Elsevier (1976)
16. Segredo, E., Segura, C., León, C.: Memetic algorithms and hyperheuristics applied to a multiobjectivised two-dimensional packing problem. Journal of Global Optimization 58(4), 769–794 (2013)
17. Segredo, E., Segura, C., León, C.: Fuzzy logic-controlled diversity-based multiobjective memetic algorithm applied to a frequency assignment problem. Engineering Applications of Artificial Intelligence 30, 199 – 212 (2014)
18. Segura, C., Coello, C.A.C., Segredo, E., Aguirre, A.H.: A Novel Diversity-Based Replacement Strategy for Evolutionary Algorithms. IEEE Transactions on Cybernetics pp. 1–14 (2015)
19. Segura, C., Coello Coello, C.A., Segredo, E., León, C.: On the adaptation of the mutation scale factor in differential evolution. Optimization Letters 9(1), 189–198 (2014)
20. Segura, C., Segredo, E., León, C.: Analysing the robustness of multiobjectivisation approaches applied to large scale optimisation problems. In: EVOLVE- A Bridge between Probability, Set Oriented Numerics and Evolutionary Computation, Studies in Computational Intelligence, vol. 447, pp. 365–391. Springer Berlin Heidelberg (2013)
21. Storn, R.: On the usage of differential evolution for function optimization. In: 1996 Biennial Conference of the North American Fuzzy Information Processing Society (NAFIPS). pp. 519–523. IEEE (1996)
22. Storn, R., Price, K.: Differential Evolution - A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. Journal of Global Optimization 11(4), 341–359 (Dec 1997)
23. Thomsen, R.: Flexible ligand docking using differential evolution. In: 2003 IEEE Congress on Evolutionary Computation (CEC). vol. 4, pp. 2354–2361. IEEE (2003)
24. Vinkó, T., Izzo, D.: Learning the best combination of solvers in a distributed global optimization environment. In: Proceedings of Advances in Global Optimization: Methods and Applications (AGO). pp. 13–17. Mykonos, Greece (June 2007)
25. Yao, X.: Evolutionary computation: Theory and applications. World Scientific (1999)
26. Yuan, X., Zhang, Y., Wang, L., Yuan, Y.: An enhanced differential evolution algorithm for daily optimal hydro generation scheduling. Computers & Mathematics with Applications 55(11), 2458–2468 (2008)
27. Zhang, J., Sanderson, A.: JADE: Adaptive Differential Evolution With Optional External Archive. IEEE Transactions on Evolutionary Computation 13(5), 945–958 (2009)
28. Zhu, H., Wang, Y., Wang, K., Chen, Y.: Particle Swarm Optimization (PSO) for the constrained portfolio optimization problem. Expert Systems with Applications 38(8), 10161–10169 (2011)