Continuous Hyper-parameter Configuration for Particle Swarm Optimization via Auto-tuning

Jairo Rojas-Delgado[1], Vladimir Milián Núñez[1(✉)], Rafael Trujillo-Rasúa[1], and Rafael Bello[2]

[1] Universidad de las Ciencias Informáticas, Havana, Cuba
{jrdelgado,vmilian,trujillo}@uci.cu
[2] Universidad Central "Marta Abreu" de Las Villas, Santa Clara, Cuba
rbellop@uclv.edu.cu

Abstract. Hyper-Parameter configuration is a relatively novel field of paramount importance in machine learning and optimization. Hyper-parameters refers to the parameters that control the behavior of algorithms and are not tuned directly by such algorithms. For hyper-parameters of an optimization algorithm such as Particle Swarm Optimization, hyper-parameter configuration is a nested optimization problem. Usually, practitioners needs to use a second optimization algorithm such as grid search or random search to find proper hyper-parameters. However, this approach forces practitioners to know about two different algorithms. Moreover, hyper-parameter configuration algorithms also have hyper-parameters that need to be considered. In this work we use Particle Swarm Optimization to configure its own hyper-parameters. Results show that hyper-parameters configured by PSO are competitive with hyper-parameters found by other hyper-parameter configuration algorithms.

Keywords: Hyper-parameter · Optimization · Meta-heuristic · Particle Swarm Optimization

1 Introduction

Search is a core concept of Artificial Intelligence and Pattern Recognition. In this field, meta-heuristic algorithms are critical as they do not require any prior on the fitness function, and allows to explore large search spaces. This ability is extensively used for model fitting and several important tasks in Artificial Intelligence. However, meta-heuristics are very dependent on their parameterization and often, require experts to determine which hyper-parameters[1] to modify and how should they be tuned, meaning that for a non-expert it might be hard to find good settings.

[1] Hyper-parameters refers to the parameters that control the behavior of the meta-heuristic algorithm and are not tuned directly by such algorithm.

I. Nyström et al. (Eds.): CIARP 2019, LNCS 11896, pp. 458–468, 2019.
https://doi.org/10.1007/978-3-030-33904-3_43

For meta-heuristics, hyper-parameters appear frequently and have a direct impact in the convergence speed and accuracy of the algorithms [7]. Also, hyper-parameters affect the execution time and memory cost of running the algorithm, the quality of the model resulting from the training process, or its ability to generalize to the unseen data. For these reasons, finding the best possible hyper-parameters becomes crucial [13].

Hyper-parameter Optimization (HPO), is a field of paramount importance in machine learning and optimization [5,17]. HPO basically stands for using optimization algorithms in order to perform automatic hyper-parameter configuration. HPO is a difficult optimization problem. Until recently, HPO in several areas, such as machine learning, was considered a combination of science and art due to the high computational cost of such task [1].

There are two main approaches to HPO: manual or automatic methods. Manually approach assumes that there exists an understanding of how the hyper-parameters affect the algorithm. On the other hand, automatic approach greatly reduce the need for this understanding, but they come at the expense of the costlier computation.

In the latest years there has been an increase in the efforts to address automatic HPO with very good results in contrast with manually choosing the hyper-parameters [3,14]. In the literature, among the most popular algorithms are Random Search (RS) [2], Sequential Model-based Algorithm Configuration (SMAC) [8] and Tree Parzen Estimators (TPE) [3]. HPO involves two nested cycles of optimization when configuring hyper-parameters for an arbitrary meta-heuristic algorithm. Here we refer to the regular optimization algorithm simply as base-algorithm. We call parent-algorithm to the optimization algorithm that deals with choosing the hyper-parameters of the base-algorithm.

When performing HPO, the base-algorithm may need to deal with thousands of dimensions and expensive fitness functions. For example, training a neural network is an NP-hard optimization problem [12] with tens of thousands of dimensions. Setting the hyper-parameters for such training algorithms is a challenge for practitioners because standard recommendations in the literature are most of the time useless. Trial and error is very tedious, is not reproducible for others and is prone to produce over-fitting. Additionally, insights on the hyper-surface structure usually are not available due to the large amount of dimensions, hence common sense may not be applicable.

When sufficient computational resources are provided, practitioners may choose to use a parent-algorithm (such as RS, SMAC or TPE) to perform HPO. There are several examples of successful applications of such parent-algorithms [3]. However, for the engineering mind this approach faces to major drawbacks:

- the practitioner needs to know the specific details of two (probably different) optimization algorithms and
- parent-algorithms also have hyper-parameters that need to be configured, manually most of the time.

For meta-heuristics, that do not impose restrictions (such as derivatives or convexity) on the fitness function, tuning its own hyper-parameters may be

a good approach to perform HPO. In this case, the parent-algorithm hyper-parameters also need to be somehow configured. However, the small number of hyper-parameters, usually less than ten, makes standard recommendations in the literature more suitable. Even if standard recommendations do not produce good results, the practitioner may use its experience about the meta-heuristic and three-dimensional insights to come up with a working set of hyper-parameters for the parent-algorithm.

In this paper, we deal with the problem of automated HPO of Particle Swarm Optimization (PSO) meta-heuristic algorithm by using the same meta-heuristic as parent-algorithm. We tested this approach with several artificial benchmark functions for optimization. Our meta-heuristic HPO of meta-heuristics approach is able to find hyper-parameters that leads to more stable and accurate optimization of the base-algorithm when compared with RS, SMAC and TPE as parent-algorithms. The main limitation of this work is that we only consider continuous hyper-parameters.

In the Sect. 2 we discuss the state of the art on hyper-parameter optimization algorithms. In Sect. 3 we describe PSO algorithm in detail and in Sect. 4 we present the benchmark test functions to be optimized by PSO. Section 5 presents the experimental results when comparing the stability and accuracy of our HPO approach. Finally, some conclusions and recommendations are given. Throughout this article we use x for scalars, $\boldsymbol{w}$ for vectors and X for sets.

2 Automatic Hyper-parameter Optimization

There are at least two kind of methods to perform automatic HPO: model-free and model-based methods. Model-based techniques build a surrogate model of the hyper-parameter space through its careful exploration and exploitation. Alternatively, model-free algorithms do not utilize the knowledge about the solution space extracted during optimization.

2.1 Model-Free Methods

Model-free approaches are characterized by not utilizing the knowledge about the solution space extracted during the optimization. This lack of adaptability makes them simple to implement at the expense of poor results for large hyper-parameter spaces.

RS: is a trivial to implement alternative to Grid Search [2], more convenient to use and faster to converge to an acceptable set of hyper-parameters. There are approaches designed to enhance RS capabilities, for example, the random sampling can be intensified in the neighborhood of the best hyper-parameters. Finally, there are hybrid algorithms to couple RS with other techniques for refining its performance,for example, manual updates provided by an expert [2,9].

2.2 Model-Based Methods

Model-based methods build a model of the fitness function, and then elaborate hyper-parameter values by performing optimization within this model. Most of such techniques use a Bayesian regression model turning this problem into a trade-off between the exploration and exploitation.

SMAC is a model-based method for optimizing algorithm hyper-parameters [8]. SMAC is effective for HPO of machine learning algorithms, scaling better to high dimensions and discrete input dimensions than other algorithms [11].

TPE method sequentially construct models to approximate the performance of hyper-parameters based on historical measurements, and then subsequently choose new hyper-parameters to test based on this model. This optimization approach is described in detail in [3].

3 Particle Swarm Optimization

PSO is a population based meta-heuristic algorithm [4]. PSO has several hyper-parameters such as: global contribution (α), local contribution (β), maximum speed (v_{max}) and minimum speed (v_{min}). Let q be the number of particles in the population, $w_i \in \mathbb{R}^n$ a particle's position in the search space and $f(\boldsymbol{w}_i)$ the fitness value of the particle.

The speed of a particle in the population is given by Eq. 1 where $\boldsymbol{w}_* \in \mathbb{R}^n$ is the position of the particle with the lowest fitness function value in the population and $\boldsymbol{w}_{i*} \in \mathbb{R}^n$ is the position where $\boldsymbol{w}_i$ achieved its lowest fitness function value. Particle $\boldsymbol{w}_*$ is known as *best-global* particle and $\boldsymbol{w}_i^*$ as *best-so-far* particle of $\boldsymbol{w}_i$.

$$\boldsymbol{v}_i \leftarrow \gamma \boldsymbol{v}_i + \alpha \boldsymbol{\omega}_1 \odot \boldsymbol{w}_* + \beta \boldsymbol{\omega}_2 \odot \boldsymbol{w}_{i*} \tag{1}$$

The parameter γ is known as speed parameter such as $v_{min} \leq \gamma \leq v_{max}$. The vectors $\boldsymbol{\omega}_1$ and $\boldsymbol{\omega}_2$ are randomly generated by means of an uniform distribution such as $\omega_{1,i} \sim U(0,1)$ and $\omega_{2,i} \sim U(0,1)$. Finally, a particle's position is updated according to Eq. 2:

$$\boldsymbol{w}_i \leftarrow \boldsymbol{w}_i + \boldsymbol{v}_i \tag{2}$$

Due to the diversity of PSO algorithms in the literature, we provide further details on the implementation used in this work as described in Algorithm 1. In step 1 the population is randomly generated. After that, in steps 2 and 3 the best-so-far positions of each particle and the best-global position of the population are set. The algorithm is considered in a convergence state after reaching a given number of fitness function evaluations determined by the value of η in step 5.

For each iteration of PSO algorithm, the best-so-far and best particles are updated in steps 10 and 13. Then, for each particle in the population a new position is calculated in steps 7–8 and the speed parameter is linearly decreased in step 17. Finally, the best particle of the population is returned in step 20.

Algorithm 1. Particle Swarm Optimization Algorithm.

Initialize population randomly
Set best-so-far particles such as $\boldsymbol{w}_{i*} \leftarrow \boldsymbol{w}_i$
Set best-global particle $\boldsymbol{w}_* \leftarrow argmin_{\boldsymbol{w}_i}(f(\boldsymbol{w}_i))$ for $1 \leq i < q$
Set speed param value at the maximum speed: $\gamma \leftarrow v_{max}$
while FITNESS_FUNCTION_EVALUATIONS( ) $< \eta$ **do**
    for j = 1 : q **do**
        Calculate speed of particle j according to Equation 1
        Update position of particle j according to Equation 2
        if $f(\boldsymbol{w}_j) < f(\boldsymbol{w}_{j*})$ **then**
            Update $\boldsymbol{w}_{j*} \leftarrow \boldsymbol{w}_j$
        end if
        if $f(\boldsymbol{w}_j) < f(\boldsymbol{w}_*)$ **then**
            Update $\boldsymbol{w}_* \leftarrow \boldsymbol{w}_j$
        end if
    end for
    if $\gamma > v_{min}$ **then**
        Reduce linarly the speed param: $\gamma \leftarrow \gamma \cdot 0.99$
    end if
end while
return Best particle in the population $\boldsymbol{w}_*$

PSO have at least four continuous hyper-parameters: local contribution, global contribution, maximum speed and minimum speed. In addition, PSO also have several other hyper-parameters of non-continuous nature, e.g. the number of particles in the population, the number of evaluations of the fitness function to archive convergence, the lower and upper bounds of the search space, etc. Here we will only deal with the four continuous hyper-parameters while tuning the others manually. Let $\bar{\boldsymbol{w}}_i \in \mathbb{R}^4$ be a vector that contains the hyper-parameters of PSO algorithm and $\bar{f}(\bar{\boldsymbol{w}}_i)$ the quality of a given set of hyper-parameters, the hyper-parameter configuration problem can be defined as:

$$\bar{\boldsymbol{w}}_* \leftarrow argmin_{\bar{\boldsymbol{w}}_i \in \mathbb{R}^4} \left[\bar{f}(\bar{\boldsymbol{w}}_i)\right] \tag{3}$$

In Eq. 3, $\bar{f}(\bar{\boldsymbol{w}}_i) = f(\boldsymbol{w}_*)$ where $\boldsymbol{w}_*$ is the result of optimizing $f(\boldsymbol{w}_j)$ by means of PSO with hyper-parameters $\bar{\boldsymbol{w}}_i$. Notice here the two nested cycles of optimization: in the outer cycle we try to minimize $\bar{f}(\bar{\boldsymbol{w}}_i)$ while in the inner cycle we try to minimize $f(\boldsymbol{w})$. While the outer cycle may be solved with GS, TPE or SMAC, here both optimization cycles are solved by PSO.

Several standard recommendations for PSO hyper-parameters are available in the literature. Here we will consider such recommendations for hyper-parameter configuration in order to compare such recommendation with automatic HPO. Some of the PSO recommendations come in the form of a general heuristic, for example, $\alpha + \beta \leq 4$ [16]. Here we will consider the following more specific recommendations:

- From [16], we denote STD.R.1 to the hyper-parameter set: $\alpha = 0.5, \beta = 0.5, v_{min} = 0, v_{max} = 1.2$.
- From [6], we denote STD.R.2 to the hyper-parameter set:, $\alpha = 2, \beta = 2, v_{min} = 0.9, v_{max} = 0.9$.

4 Optimization Benchmark Problems

In this section we describe a set of optimization problems to be solved by base-algorithms. In the literature, several benchmark test functions for optimization have been suggested. We considered five properties regarding to the test functions: continuous/discontinuous, differentiable/non-differentiable, separable/non-separable, scalable/non-scalable and uni-modal/multi-modal [10, 15]. The following list enumerate benchmark test function used in this work and its properties, in all cases consider that $\boldsymbol{w} \in \mathbb{R}^n$.

1. De Jung function:

$$f_1(\boldsymbol{w}) = \sum_{i=0}^{n} {\boldsymbol{w}_i}^2 \tag{4}$$

   has a global minima in $f_1(0, ..., 0) = 0$. De jung is continuous, differentiable, separable, scalable and multi-modal. A commonly used search domain is $0 \leq \boldsymbol{w}_i \leq 10$.
2. Ackley function:

$$f_2(\boldsymbol{w}) = -20e^{0.02\sqrt{n^{-1}\sum_{i=0}^{n} \boldsymbol{w}_i^2}} - e^{n^{-1}\sum_{i=0}^{n} \cos(2\pi \boldsymbol{w}_i)} + 20 + e \tag{5}$$

   has a global minima in $f_2(0, ..., 0) = 0$. Ackley is continuous, differentiable, non-separable, scalable and multi-modal. A commonly used search domain is $-35 \leq \boldsymbol{w}_i \leq 35$.
3. Grienwangk function:

$$f_3(\boldsymbol{w}) = \sum_{i=0}^{n} \boldsymbol{w}_i^2/4000 - \prod_{i=0}^{n} \cos(\boldsymbol{w}_i/\sqrt{i}) + 1 \tag{6}$$

   has a global minima in $f_3(0, ..., 0) = 0$. Griendwangk is continuous, differentiable, non-separable, scalable and multi-modal. A commonly used search domain is $-100 \leq \boldsymbol{w}_i \leq 100$.
4. Rastrigin function:

$$f_4(\boldsymbol{w}) = 10n + \sum_{i=0}^{n} \boldsymbol{w}_i^2 - 10\cos(2\pi \boldsymbol{w}_i) \tag{7}$$

   has a global minima in $f_4(0, ..., 0) = 0$. Rastrigin is continuous, differentiable, separable, scalable and multi-modal. A commonly used search domain is $-5.12 \leq \boldsymbol{w}_i \leq 5.12$.

5. Rosenbrock function:

$$f_5(\boldsymbol{w}) = \sum_{i=0}^{n-1} [100(\boldsymbol{w}_{i+1} - \boldsymbol{w}_i^2)^2 + (\boldsymbol{w}_i - 1)^2] \tag{8}$$

has a global minima in $f_5(1, ..., 1) = 0$. Rosenbrock is continuous, differentiable, non-separable, non-scalable and multi-modal. A commonly used search domain is $-30 \leq \boldsymbol{w}_i \leq 30$.

6. Schwefel function:

$$f_6(\boldsymbol{w}) = -1/n \sum_{i=0}^{n} \boldsymbol{w}_i \sin \sqrt{|\boldsymbol{w}_i|} \tag{9}$$

has a global minima in $f_6() = -418.983$. Schwefel is continuous, differentiable, separable, scalable and multi-modal. A commonly used search domain is $-500 \leq \boldsymbol{w}_i \leq 500$.

7. Styblinski-Tang function:

$$f_7(\boldsymbol{w}) = 0.5 * \sum_{i=0}^{n} \boldsymbol{w}_i^4 + 16\boldsymbol{w}_i^2 + 5\boldsymbol{w}_i \tag{10}$$

has a global minima in $f_7(-2.903534, ..., -2.903534) = -78.332$. Styblinski-Tang is continuous, differentiable, non-separable, non-scalable and multi-modal. A commonly used search domain is $-5 \leq \boldsymbol{w}_i \leq 5$.

8. Step function:

$$f_8(\boldsymbol{w}) = \sum_{i=0}^{n} (\lfloor \boldsymbol{w}_i + 0.5 \rfloor)^2 \tag{11}$$

has a global minima in $f_8(0, ..., 0) = 0$. Step is discontinuous, non-differentiable, separable, scalable and uni-modal. A commonly used search domain is $-100 \leq \boldsymbol{w}_i \leq 100$.

9. Alpine function:

$$f_9(\boldsymbol{w}) = \sum_{i=0}^{n} |\boldsymbol{w}_i \sin(\boldsymbol{w}_i) + 0.1\boldsymbol{w}_i| \tag{12}$$

has a global minima in $f_9(0, ..., 0) = 0$. Alpine is continuous, non-differentiable, separable, scalable and uni-modal. A commonly used search domain is $10 \leq \boldsymbol{w}_i \leq 10$.

5 Results and Discussion

This section describes the experimental setup and results of the proposed PSO continuous hyper-parameter optimization approach. We compare quality of PSO

hyper-parameters from configured with RS, SMAC, TPE and PSO it self. Moreover, we present a comparison of automatic hyper-parameter optimization with standard recommendation in the literature: STD.R.1 and STD.R.2.

Base PSO was configured with a population size of 40 particles and a search domain based on the literature recommendation (see Sect. 4). For PSO as base-algorithm, we consider convergence when the number of fitness function evaluations is above 4.0E+4. For parent-algorithms (RS, SMAC, TPE and PSO), we consider convergence when number of fitness function evaluations is above 100. For PSO as parent-algorithm the population size is of 10 particles and the search domain is defined in the interval $[0, 2]$. Hyper-parameters for PSO as parent-algorithm are configured according to STD.R.1.

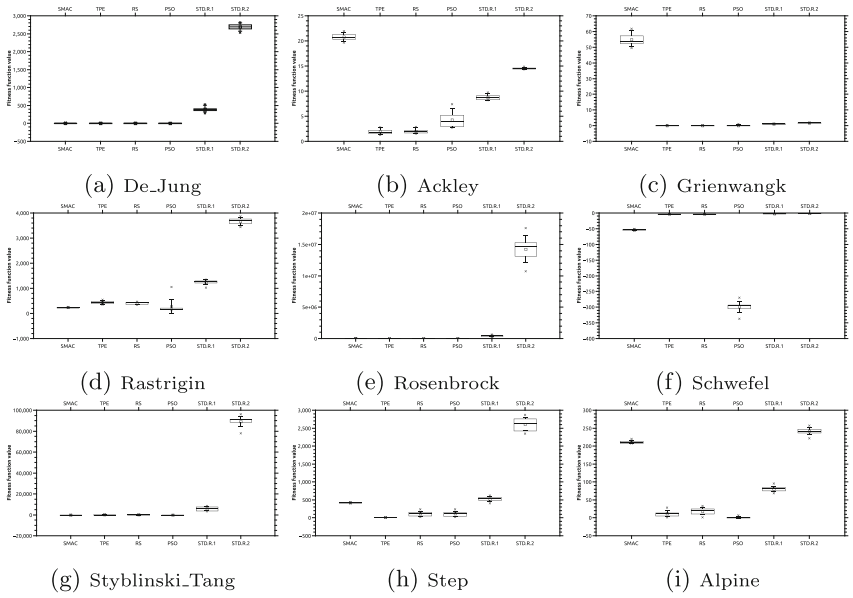

Fig. 1. Accuracy of PSO when considering different parent-algorithms for HPO.

Figure 1 shows the accuracy of PSO when considering different benchmark optimization problems after performing hyper-parameter configuration by means of different parent-algorithms. The bottom and top lines in the box plots represents the first and third quartiles, the line in the middle represents the median of 10 measurements and the whiskers represent standard deviation. The small square represents the average of the measurements and the (x) marks represent the maximum and minimum value.

In Table 1 we a summary of the accuracy of PSO when considering different benchmark optimization problems. The presented accuracy values are the averages of 10 measurements. In addition, we show the global best for each optimization problem.

Table 1. Accuracy of PSO algorithm with hyper-parameters optimized by different parent-algorithms.

Fitness function	BEST	SMAC	TPE	RS	PSO	STD.R.1	STD.R.2
De_jung	0.00E+00	0.00E+00	6.63E−01	2.63E+00	**0.00E+00**	3.96E+02	2.69E+03
Ackley	0.00E+00	2.08E+01	2.56E+00	**1.98E+00**	3.26E+00	8.75E+00	1.45E+01
Grienwangk	0.00E+00	5.40E+01	1.54E−01	**5.56E−02**	7.02E-01	1.12E+00	1.67E+00
Rastrigin	0.00E+00	2.35E+02	5.06E+02	4.10E+02	**1.28E+02**	1.24E+03	3.67E+03
Rosenbrock	0.00E+00	4.28E+04	**8.44E+02**	1.11E+03	1.85E+05	4.43E+05	1.42E+07
Schwefel	−4.18E+02	−5.31E+01	−3.41E+00	−3.35E+00	**−2.42E+02**	−2.04E+00	−1.05E+00
Styblinski_tang	−7.81E+01	−1.08E+01	−3.00E+00	1.57E+01	**−1.45E+01**	5.79E+03	8.92E+04
Step	0.00E+00	4.24E+2	**1.30E+01**	1.16E+02	4.43E+02	5.27E+02	2.61E+03
Alpine	0.00E+00	2.13E+02	2.54E+01	1.82E+01	**5.33E+00**	8.11E+01	2.42E+02

We use the non-parametric Friedman test of differences among repeated measures to find statistical significant differences between parent-algorithms SMAC, TPE, RS, PSO and standard recommendations STD.R.1 and STD.R.2. The results of this test rendered a Chi-square value of 10.54 which was significant for a p-value < 0.01. This way we reject the null hypothesis, hence, the means of the results of two or more algorithms are not the same.

Furthermore, we conduct a Benferroni-Dunn test to compare the control parent-algorithm (PSO) with the other alternatives for hyper-parameter configuration. This way, we reject the null hypothesis of similar means for PSO vs. STD.R.1 (Chi-square value of 2.83 which was significant for a p-value $= 0.022$) and for PSO vs. STD.R.2 (Chi-square value of 4.09 which was significant for a p-value < 0.01). For PSO vs. SMAC, PSO vs. TPE and PSO vs. RS we accept the null hypothesis which means that in such cases the mean of the results of the control method against each other groups is equal.

As can be seen, for each optimization problem, automatic HPO always outperforms standard recommendation in the literature with statistical significant differences. In addition, PSO obtains as average better results than RS, SMAC and TPE although we don't see statistical significant differences. Also, the computational cost (execution time) of the parent-algorithms are very similar.

6 Conclusions and Recommendations

In this paper, we compare different strategies to perform HPO of PSO. We observe that automatic HPO provides better results than standard recommendation in the literature. Moreover, optimizing PSO hyper-parameters with PSO it self proved to be a competitive approach when compared with popular HPO algorithms such as RS, SMAC or TPE. This approach has the advantage that the practitioner do not need to consider several optimization algorithms in order to find proper hyper-parameters. However, the main limitation of this work is that only continuous hyper-parameters can be configured with this approach. Future research should investigate hybrid meta-heuristic approaches that allow to consider not only continuous hyper-parameters.

References

1. Bengio, Y.: Practical recommendations for gradient-based training of deep architectures. In: Montavon, G., Orr, G.B., Müller, K.R. (eds.) Neural Networks: Tricks of the Trade, pp. 437–478. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-35289-8_26
2. Bergstra, J., Bengio, Y.: Random search for hyper-parameter optimization. J. Mach. Learn. Res. **13**(Feb), 281–305 (2012)
3. Bergstra, J., Yamins, D., Cox, D.D.: Making a science of model search: hyperparameter optimization in hundreds of dimensions for vision architectures. In: Proceedings of the 30th International Conference on International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, vol. 28, pp. I-115–I-123. JMLR.org (2013). http://dl.acm.org/citation.cfm?id=3042817.3042832
4. Bonyadi, M.R., Michalewicz, Z.: Particle swarm optimization for single objective continuous space problems: a review. Evol. Comput. **25**, 1–54 (2017)
5. Candelieri, A., et al.: Tuning hyperparameters of a SVM-based water demand forecasting system through parallel global optimization. Comput. Oper. Res. **106**, 202–209 (2019)
6. Elegbede, C.: Structural reliability assessment based on particles swarm optimization. Struct. Saf. **27**(2), 171–186 (2005)
7. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT Press, Cambridge (2016)
8. Hutter, F., Hoos, H.H., Leyton-Brown, K.: Sequential model-based optimization for general algorithm configuration. In: Coello, C.A.C. (ed.) LION 2011. LNCS, vol. 6683, pp. 507–523. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-25566-3_40
9. Larochelle, H., Erhan, D., Courville, A., Bergstra, J., Bengio, Y.: An empirical evaluation of deep architectures on problems with many factors of variation. In: Proceedings of the 24th International Conference on Machine Learning, pp. 473–480. ACM (2007)
10. Liang, J.J., Suganthan, P.N., Deb, K.: Novel composition test functions for numerical global optimization. In: Proceedings of 2005 IEEE Swarm Intelligence Symposium, SIS 2005, pp. 68–75. IEEE (2005)
11. Lindauer, M., Hutter, F.: Warmstarting of model-based algorithm configuration. In: Thirty-Second AAAI Conference on Artificial Intelligence (2018)
12. Livni, R., Shalev-Shwartz, S., Shamir, O.: On the computational efficiency of training neural networks. In: Advances in Neural Information Processing Systems, pp. 855–863 (2014)
13. Lorenzo, P.R., Nalepa, J., Ramos, L.S., Pastor, J.R.: Hyper-parameter selection in deep neural networks using parallel particle swarm optimization. In: Proceedings of the Genetic and Evolutionary Computation Conference Companion, pp. 1864–1871. ACM (2017)
14. Maclaurin, D., Duvenaud, D., Adams, R.: Gradient-based hyperparameter optimization through reversible learning. In: International Conference on Machine Learning, pp. 2113–2122 (2015)
15. Momin, J., Yang, X.S.: A literature survey of benchmark functions for global optimization problems. J. Math. Model. Numer. Optim. **4**(2), 150–194 (2013)

16. Parsopoulos, K.E., Vrahatis, M.N.: Recent approaches to global optimization problems through particle swarm optimization. Nat. Comput. **1**(2–3), 235–306 (2002)
17. Probst, P., Wright, M.N., Boulesteix, A.L.: Hyperparameters and tuning strategies for random forest. Wiley Interdiscip. Rev. Data Mining Knowl. Discov. **9**(3), e1301 (2019)