Algorithm Portfolio for Parameter Tuned Evolutionary Algorithms

Hao Tong, Shuyi Zhang, Changwu Huang, and Xin Yao*
University Key Laboratory of Evolving Intelligent Systems of Guangdong Province
Department of Computer Science and Engineering
Southern University of Science and Technology, Shenzhen 518055, China
Emails: htong6@outlook.com, shuyi9404@sina.com, {huangcw3, xiny}@sustech.edu.cn

***Abstract*—Evolutionary algorithms' performance can be enhanced significantly by using suitable parameter configurations when solving optimization problems. Most existing parameter-tuning methods are inefficient, which tune algorithm's parameters using whole benchmark function set and only obtain one parameter configuration. Moreover, the only obtained parameter configuration is likely to fail when solving different problems. In this paper, we propose a framework that applying portfolio for parameter-tuned algorithm (PPTA) to address these challenges. PPTA uses the parameter-tuned algorithm to tune algorithm's parameters on one instance of each problem category, but not to all functions in the benchmark. As a result, it can obtain one parameter configuration for each problem category. Then, PPTA combines several instantiations of the same algorithms with different tuned parameters by portfolio method to decrease the risk of solving unknown problems. In order to analyse the performance of PPTA framework, we embed several test algorithms (i.e. GA, DE and PSO) into PPTA framework constructing algorithm instances. And the PPTA instances are compared with default test algorithms on BBOB2009 and CEC2005 benchmark functions. The experimental results has shown PPTA framework can significantly enhance the basic algorithm's performance and reduce its optimization risk as well as the algorithm's parameter-tuning time.**

***Keywords*—Knowledge transfer, Auto parameter tuning, Algorithm portfolio, Evolutionary algorithm.**

I. Instruction

All evolutionary algorithms have some parameters which have strong influence on the algorithm's performance, which is required to be tuned. The basic idea of parameter tuning is to generate many parameter settings in the configuration space and evaluate their performance on some selected problem instances. The configuration performing best is selected as the best parameter setting. Many effective parameter tuning frameworks are proposed in the literature [1], [2], [3]. F-Race method evaluates the configuration candidates on some instances incrementally and once sufficient evidence proving some candidates are significantly worse than others, these candidates will be removed [4]. ParamILS framework applies the iterated local search in the default configuration and some randomly generated configurations to find much better parameter configurations [5]. The commonly used metric for evaluating parameter configuration is to run parameter candidates several times on problem instances and calculate its average performance [6], which is usually very expensive. Therefore, sequential parameter optimization (SPO) framework considers the parameter tuning problem as a computationally expensive problem and employs the efficient global optimization (EGO) algorithm [7] to help find the optimal parameter configuration [8].

According to no free lunch (NFL) theorem [9], one parameter configuration which is suitable for some problems might fail in optimizing other kinds of problems. Algorithm portfolio can be applied to meet this challenge. It allocates the limited computational resources to different algorithms or operators to reduce the risk of failing in optimizing problems. Peng et al. [10] proposed population-based algorithm portfolio (PAP) framework to reduce the risk of problem solving for evolutionary algorithm by running algorithm constituents parallelly and using individual migration to share information. Shiu et al. [11] employed a learning method to predict the online performance of different algorithms and proposed Multi-EAs portfolio frameworks. Hao et al. [12] employed reinforcement learning strategy to design a portfolio framework for surrogate-assisted evolutionary algorithms. Besides, algorithm portfolio is also successfully applied to solve SAT problems [13] and noisy optimization problems [14], [15].

In this paper, we consider algorithm parameter configurations which are best in different problem instances. The single best parameter setting has a high probability of failing in solving unknown problems. Therefore, we employed an algorithm portfolio framework to combine several instantiations of the same algorithms with different best parameters obtained from different problem categories. We optimize the test algorithms' parameters on one instance of each problem category so that we can obtain several best parameter configurations. This process can be regarded as training process and the problem instance used to tune the parameters can be called training instance. Then, the test algorithm assisted by these tuned-parameters are combined by portfolio method and the ensemble algorithm is used to solving the remaining benchmark functions. And this phase can be regarded as testing process. In our experimental studies, the CEC2005 and BBOB2009 benchmark functions are used as training instances as well as testing instances, both of which are classified into several categories.

The remainder of this paper is structured as follows. Section

* Corresponding author

II will explain the motivation of this paper and some related work are also introduced in this section. The main structure of our proposed framework is introduced in Section III. A series of experiments and further analysis will be presented in Section IV. Finally, this paper will end with a brief conclusion and a discussion of future work in Section V.

II. Motivation and related-work

A. Motivation

As we all know, the NFL indicates that no algorithm can outperform all other algorithms in all problems. Similarly, there's no best parameter configuration for one algorithm which can obtain the best performance in every problem. That is, researchers obtained suitable parameter configuration in some training instances, and algorithm with these parameters is effective on this kind of problems, but it is likely to fail in another kind of problems which we called as optimization risk. Algorithm portfolio allocates computational resources to different algorithms on the basis of algorithm constituents' performance to reduce the optimization risk. Therefore, we can employ the algorithm portfolio to reduce the risk by combining several instantiations of the same algorithms with best parameters obtained from different categories of problem instances.

On the other hand, we also have mentioned that the performance metric in the parameter tuning method is usually to repeat executing the algorithm with parameter configuration candidate and use average performance as the measurement. Thus, if we want to obtain a suitable parameter configuration on a benchmark, it is required to run one parameter configuration on all functions several runs. Assume the average running time of an algorithm on one function is t and the benchmark contains M functions totally. Therefore, if we evaluate parameter configuration by running K times independently on one function, the computational time T_{s0} for evaluating one parameter configuration will be

$$T_{s0} = KM \cdot t \tag{1}$$

Obviously, it is a computationally expensive problem.

However, if problems have some similar characteristics, i.e. similar fitness landscape, the algorithm's performance is also similar in these problems. The fitness landscape analysis uses this basic idea to select a suitable algorithm for unknown problems [16]. Therefore, it motivated us to tune parameters on one problem and use them to optimize the similar problems. So, we only need to tune parameters on one problem instance for each problem category and then obtain several best parameters. As a result, the computational time for parameter tuning can be reduced significantly.

Assume we have N categories of training instances, then the total computational time T_{s1} for evaluating parameter is

$$T_{s1} = KN \cdot t, N \ll M \tag{2}$$

As a result, we design a new framework which combines parameter tuning method and algorithm portfolio techniques to reduce the optimization risk and reduce the time for tuning parameters.

B. Related work

1) Sequential Parameter Optimization: As we analysed before, the parameter tuning problem is a computationally expensive problem. Therefore, researchers employed surrogate-assisted optimization algorithm to search the suitable parameter configuration. Sequential parameter optimization (SPO) [8] is a typical algorithm which uses the EGO algorithm to handle expensive problems. The pseudo-code of the SPO algorithm is presented in Algorithm 1.

Algorithm 1: Sequential Parameter Optimization [8]

```
Use DOE to initial a set of parameter configurations;
Evaluate these initial parameter configurations and add
  them into the empty set D_c;
Set D_c = {(c_i, y_i)|i = 1, 2, ..., N_p}
while Stop criterion is not satisfied do
    Construct Kriging model M_K using points in D_c;
    Search for the most promising points c_k;
    Evaluate the new parameter configuration (c_k, y_k);
    Add (c_k, y_k) into D_c.
end
```

Firstly, SPO starts with the initialization of a set of points, i.e. parameter candidates, by Design of Experiments (DOE), like Latin hypercube sampling (LHS) [17]. Each generated parameter configuration candidate will be evaluated by performance metric and get its 'fitness'. And all these design points will be added into an archive. After that, SPO will employ the Kriging model to construct a surrogate model. Then, SPO uses the optimization algorithm to search the most promising point based on expected improvement and re-evaluate the new parameter configuration by the performance metric. Finally, the new parameter candidate will be added into the archive for the next iteration. The iteration will be ended by running out of max fitness evaluations.

2) Population-based Algorithm Portfolio: Population-based algorithm portfolio (PAP) is a very popular and typical portfolio framework in the literature [10]. It combines different types of population-based evolutionary algorithms to reduce the optimization risk. The algorithm constituents being embed into the framework are required to be complementary so that the algorithm portfolio can effectively handle different types of problems. The pseudo-code of PAP is presented in Algorithm 2.

Assume that there are m algorithm constituents, PAP framework starts by the initialization of m subpopulations that each algorithm has its own independent population. In the PAP framework, the computational resources, i.e. FEs, is allocated to every algorithm before the optimization process according to experts' knowledge. After that, each algorithm will evolve independently from each other. However, in order to enhance the information sharing between different algorithm

Algorithm 2: Population-based Algorithm Portfolio [10]

```
Initial m subpopulations: pop_1, pop_2, ..., pop_m ;
while Stop criterion is not stastified do
    for i ← 1 to m do
        Using algorithm i to operate pop_i;
    end
    if Reached migration generation then
        for i ← 1 to m do
            The worst individual of pop_i is replaced by
              best individual of remainder m − 1
              subpopulations;
        end
    end
end
```

constituents, the framework uses the migration strategy every $migration\_interval$ generations that replaces the worst individual of each subpopulation by the best individual of remainder subpopulations. Finally, the framework terminates once the computational resources for each algorithm constituents are exhausted.

III. Proposed framework

In this section, we propose a new framework combining parameter tuning and algorithm portfolio techniques, named portfolio for parameter-tuned algorithm (PPTA). More details about the proposed framework will be introduced in the following.

A. Main structure

The main structure of the PPTA framework is presented in Figure 1. It mainly contains three phases, including *Instance Classification, Parameter Tuning* and *Algorithm portfolio*. The first two phases are offline procedures to obtain several better parameters and the third phase is an online procedure which is used for solving test problems or real-world problems.

1) ***Instances Classification****:* The first phase for the PPTA framework is instance classification. In this stage, the training instances (i.e. benchmark functions) will be classified into several categories according to their characteristics. For each category, one instance will be chosen as the training instance and the remaining instances are used as test problems to validate the performance of our proposed framework. That is, we tune the test algorithm's parameters in the training functions and used these tuned parameters to construct algorithm portfolio solving remainder benchmark functions as shown in Figure 1.

2) ***Parameter Tuning****:* The second phase of PPTA is the parameter tuning process. In this stage, the algorithm's parameters are tuned by SPO algorithm on training instances selected in the first phase. Different from the normal parameter-tuning algorithm, we evaluate each parameter candidates only on one problem instance which is much faster than using a set of benchmark functions. Therefore, the SPO algorithm in PPTA framework for a specific algorithm can be regarded as a function that the input is one training instance and the output is the best parameter configuration ever found for this problem.

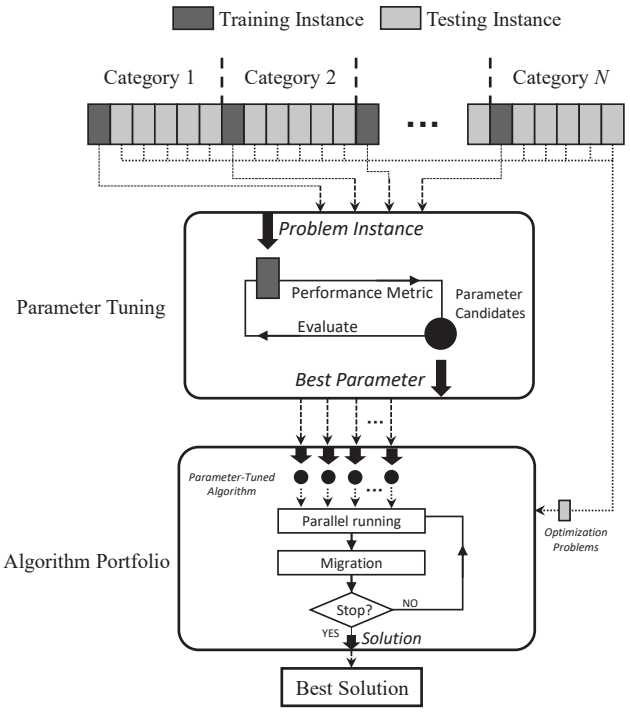

Fig. 1: The structure of proposed framework.

For each selected problem instance in *Instance Classification* phase, the SPO algorithm in *Parameter Tuning* phase will obtain a suitable parameter configuration. Assume we have N categories functions in the training instance set, we can get parameter configuration set $\mathcal{C} = \{c_i | i = 1, 2, ..., N\}$. This phase is just aimed to obtain several good parameters so that this phase is an offline stage which means the parameters obtained can be used for an algorithm to solve many real-world problems. At the end of this stage, we will obtain a set of parameter candidates which is suitable for different categories of problems. Test algorithm using these parameters constructs the constituents of an algorithm portfolio in the next stage.

3) ***Algorithm Portfolio****:* Finally, the third phase of PPTA is an algorithm portfolio that combines several instantiations of the same algorithm with different parameters. This stage can also be considered as an independent part which is used for solving real-world problems. The parameters obtained in the second stage can be used forever. When we face an unknown problem, it's not required to tune the parameters again, and the portfolio algorithm instances can be used directly. Therefore, this stage is an online procedure that we just use this part to optimize the problem as long as we obtain a set of parameters in the second stage.

Assume we have obtained N parameters, i.e. $\mathcal{C} = \{c_i | i = 1, 2, ..., N\}$, then we can get the algorithm constituents set

$\mathcal{A} = \{A_i | i = 1, 2, ..., N\}$. And actually, all algorithms A_i are the same algorithm but only with different parameter settings. Then the PAP framework is applied to assist the algorithm portfolio. All algorithm constituents run in parallel, and the migration process executes every several generations to share information between algorithm constituents.

B. Discussion

1) Generality: The proposed PPTA framework mainly contains SPO algorithm for parameter tuning and PAP algorithm for algorithm portfolio. However, the PPTA framework actually can be regarded as a general framework of the portfolio for parameter-tuned algorithm. Therefore, the algorithm for parameter tuning and algorithm portfolio can be replaced by any efficient algorithm. For example, the ParamILS algorithm can also be used to tune parameters and MultiEA framework can be used for algorithm portfolio. The generality makes PPTA be a more general framework.

2) Universality: In the literature, the algorithm portfolio is usually used for different algorithms. But different algorithms have different strategies which are difficult to combine them together, especially in sequential portfolio framework. However, this problem doesn't exist in parameter-tuned algorithm portfolio because all algorithm constituents are the same algorithm except some important parameter settings. Therefore, the portfolio framework in PPTA framework can be parallel-based framework as well as sequential-based framework.

The algorithm portfolio requires algorithm constituents to be complementary so that the portfolio algorithm can handle different types of problems. But it is not easy to select the complementary algorithms which need to test algorithm candidates in benchmark functions. In PPTA framework, the parameter configuration obtained from different kinds of problems are naturally complementary thanks to the classification of training instances. Therefore, we can obtain the complementary algorithm much easier to construct the algorithm portfolio.

3) Robustness: For each algorithm constituent in the portfolio, it is the best parameter configuration found for the specific kind of problems. Each algorithm is suitable for a kind of problem so that the portfolio algorithm can handle several kinds of problems. As a result, the optimization risk is obviously reduced.

4) Efficiency: The time complexity of parameter tuning algorithm in PPTA framework can be reduced significantly compared with commonly used parameter tuning methods. As mentioned before, the performance metric only evaluates the parameter candidates on one specific problem instance.

In the algorithm portfolio phase, the time complexity doesn't increase a lot compared with the single algorithm because the operators of the algorithm in PPTA are the same with single algorithm and the total computational resources (i.e. max FEs) are also the same. The only difference is the strategy allocating computational resources which varies from algorithm to algorithm.

IV. Experiments

In order to analyse the performance of PPTA framework, we conduct a series of experiments in two commonly used benchmark functions, respectively. The detail of experimental configuration and obtained results will be discussed in this section.

A. Experimental setting

1) Benchmark function: The benchmark functions used in this paper are CEC2005 [18] and BBOB2009 [19] benchmark functions which is very common in the literature. At the same time, the testing problems of two benchmarks have been classified into several categories which are very convenient to test the proposed framework. The detail of two benchmarks are presented in Table I.

For each category, we select one problem as training instance and the remaining problems are used to test the PPTA framework. Therefore, for each testing algorithm, we could obtain 5 best parameter configuration in BBOB benchmark and 4 best parameter configurations in CEC benchmark. Then the algorithm portfolio combining 4 and 5 algorithm constituents will be tested in remainder CEC2005 problems and BBOB2009 problems, respectively.

TABLE I: Details of test benchmark functions.

Benchmark	Types	Problem Types	Training Instances	Testing Instances
BBOB2009	T_{B1}	Separable	f_1	f_2, f_3, f_4, f_5
	T_{B2}	Functions with low or moderate conditioning	f_6	f_7, f_8, f_9
	T_{B3}	Functions with high conditioning and unimodal	f_{10}	$f_{11}, f_{12}, f_{13}, f_{14}$
	T_{B4}	Multi-modal functions with adequate global structure	f_{15}	$f_{16}, f_{17}, f_{18}, f_{19}$
	T_{B5}	Multi-modal functions with weak global structure	f_{20}	$f_{21}, f_{22}, f_{23}, f_{24}$
CEC2005	T_{C1}	Unimodal Functions	f_2	f_3, f_4, f_5
	T_{C2}	Basic Multimodal Functions	f_6	f_{7-12}
	T_{C3}	Expanded Multimodal Functions	f_{13}	f_{14}
	T_{C4}	Hybrid Composition Functions	f_{15}	f_{16-25}

2) Test algorithms: We select three basic algorithms to test the performance of PPTA, including Genetic Algorithm (GA) [20], Differential Evolution (DE) [21] and Particle Swarm Optimization (PSO) [22]. Each algorithm contains several important parameters in the algorithm and we tune them in the selected training instances. The parameters we tuned for each algorithm and their default values, (i.e. the original values) are presented in Table II.

TABLE II: Parameter setting in test algorithms.

Algorithm	Parameters	Default Value
GA	$gamma; mu$	$gamma = 0.4; mu = 0.1$
DE	policy; $F; CR$	policy=$/rand/1$; $F = 0.5; CR = 0.9$
PSO	$c_1; c_2$	$c_1 = 2; c_2 = 2$

The $gamma$ and mu in GA represent the crossover rate and mutation rate [20]. In DE algorithm, policy represents the choice of mutation operator and F, CR are two important parameters in mutation and crossover operators [21]. The policy candidates contain $/rand/1$, $/best/1$, $/current-to-best/1$, $/best/2$ and $/rand/2$. In PSO algorithm, c_1, c_2 are two flying factors for each particle [22].

TABLE III: The tuned parameters for test algorithms with Dimension $d = 30$.

Types	GA		DE			PSO	
	gamma	mu	policy	F	CR	c1	c2
TB1	0.4294	0.1728	4	0.4155	0.5441	1.7848	0.5908
TB2	0.5972	0.0617	3	0.6553	0.5350	0.1467	3.0000
TB3	0.5493	0.9737	3	0.7791	0.9308	0.7161	1.9731
TB4	0.5329	0.6145	1	0.9584	0.9273	1.7811	1.7866
TB5	0.9964	0.0839	1	0.4453	0.0581	0.4452	3.0000
TC1	0.5972	0.0617	1	0.7791	0.9308	1.2194	2.3496
TC2	0.6975	0.7769	1	0.6184	0.7194	0.7161	1.9731
TC3	0.8586	0.0275	5	0.4000	0.0000	2.6162	0.9326
TC4	0.9057	0.4392	2	0.7341	0.0122	2.5756	0.4760

The tuned parameters for each testing algorithm in every type of problem is showed in Table III. We can find the best parameters ever found for each type of problems are totally different.

3) Running condition: For every test algorithm, we tune its parameters in training instances and the metric for evaluating the parameter configuration's performance is to repeat executing the algorithm 15 times in the training problem instance.

Then, we construct algorithm portfolio instances in two benchmark problems. We run PAP instances for 15 times as well in each test problems. The problems dimension are set as 30 and the max fitness evaluation number for training and testing are both 300000.

B. Experiment Results and Analysis

1) Comparative result for test algorithms: The results obtained by PPTA instances and default parameter test algorithms over 15 independent runs on two benchmark functions are presented in Tables IV and V. And we plotted one evolve curves for each problem types because of the page limitation, which is shown in Figure 2.

In Tables IV and V, the figures in each cell represent the average of best fitness and standard deviation over 15 runs. The average ranking is calculated and listed in the penultimate row. Furtherly, the Wilcoxon test with a 0.05 significance level is used and the results are provided in the last row of each table, in which the PPTA instance is the control method for each testing algorithm. The 'win-draw-lose' represents the PPTA instance is superior, not significantly different and inferior to the default algorithm. From the average ranking and Wilcoxon Test, we can easily find that the PPTA instance performs better than default parameter test algorithms in most cases and have a similar performance in a few cases. Therefore, we can obtain the overall conclusion that our proposed framework, PPTA, is much more effective than the default parameter algorithm although it might fail in a few problems.

In two result tables, we can find that the PPTA instances are not always better than default algorithms in test cases. In two benchmarks, the PPTA framework enhances the performance of DE and GA most obviously that DE/GA-PPTA outperforms DE/GA-Default in most test cases. By contrast, PSO-PPTA obtains a similar performance to PSO-Default. We can analyse that the parameters in DE/GA are more important than parameters in PSO with respect to the influence on algorithm's performance. As a result, the effect of PPTA to DE/GA is more obvious than PSO.

In the typical evolutionary curves we selected showing in Figure 2 for each type, lines without any symbol represent PPTA instances and the remaining lines are default parameter value algorithms. Different test algorithms have different line colour and line type. From all evolve curves, there mainly exists three situations between PPTA instance and default test algorithm:

- Firstly, default algorithm converges much faster than PPTA at first stage but it is exceeded by PPTA in the later stage. It is probably because that PPTA contains several search agents and the evolutionary iteration is much fewer than default algorithm so that the performance at first is inferior to the single default test algorithms. But at later stage, thanks to the several agents, the search diversity of PPTA instance is much more than the default algorithm so that it has a more powerful optimization ability.
- The second situation is default algorithm converges much faster than PPTA at first stage and the final performance is similar to PPTA. This situation only happens in some cases, such as DE in $TB1$ and GA in $TB3$. The most possible reason for this situation is the algorithm can't handle this type of problem even though the parameters have been tuned.
- The third situation is the behaviour of PPTA and the default algorithm is almost same with each other, like PSO algorithm in almost all types. It is because the parameters we selected have no great influence on algorithm's performance as mentioned before.

What's more, the algorithm portfolio with different tuned parameters is not effective all the time, such as the PSO algorithm in the results as we discussed before. The main reason for deteriorative performance is the parameter has no great influence on the optimization ability and the algorithm ensemble reduce the evolutionary iteration at the same time. As a result, the PPTA framework can't assist all algorithms and even pull down algorithm's performance. On the other hand, the similarity in fitness landscape also can't guarantee parameters' optimality in different problems.

2) Performance of parameter transfer and computational time: In order to analyse the assumption that the tuned-parameters for one problem instance are also effective in similar problem instances, we calculate the Wilcoxon-test results for each type of problems which is presented in Table VI. Three figures in each cell represent PPTA instance win, draw and lose default algorithm. It's obvious that PPTA assisted algorithm performs better or equivalent to the default algorithm in most types. We can conclude that the basic idea of the proposed framework is significative to some extent. But it doesn't work to all types because of the problem's complexity, parameters' importance, etc. The tuned parameters can transfer among problems with similar fitness landscape, which means it has the probability to enhance the performance but can't guarantee the optimization ability.

TABLE IV: The results of comparing PPTA instance with original test algorithm in BBOB2009 benchmark over 15 runs including the average best fitness and standard deviation shown as AVR ± STD. The boldface figures are the best fitness between two algorithms in each test problem.

Problems	GA		DE		PSO	
	GA-PPTA	GA-Default	DE-PPTA	DE-Default	PSO-PPTA	PSO-Default
f2	**0.0000e+00 ± 0.0000e+00**	**0.0000e+00 ± 0.0000e+00**	**0.0000e+00 ± 0.0000e+00**	7.6278e+01 ± 1.6427e+02	**0.0000e+00 ± 0.0000e+00**	**0.0000e+00 ± 0.0000e+00**
f3	1.2641e+01 ± 2.6399e+00	**1.4600e+00 ± 1.1806e+00**	**6.6331e-02 ± 2.5690e-01**	3.0495e+01 ± 8.9158e+00	**2.6798e+01 ± 1.0411e+01**	3.8472e+01 ± 1.2617e+01
f4	3.4109e+01 ± 4.1466e+00	**4.9270e+00 ± 1.3782e+00**	**1.9899e+00 ± 1.3027e+00**	4.8084e+01 ± 1.9459e+01	**3.6747e+01 ± 1.2019e+01**	4.9615e+01 ± 1.5028e+01
f5	**0.0000e+00 ± 0.0000e+00**	4.8845e-03 ± 1.8918e-02	**0.0000e+00 ± 0.0000e+00**	**0.0000e+00 ± 0.0000e+00**	**0.0000e+00 ± 0.0000e+00**	**0.0000e+00 ± 0.0000e+00**
f7	**8.9767e+00 ± 3.3446e+00**	1.6607e+01 ± 6.2799e+00	**3.1756e+00 ± 1.8344e+00**	1.3897e+01 ± 5.8760e+00	1.2717e+01 ± 4.0246e+00	**6.2410e+00 ± 3.3496e+00**
f8	**2.4902e+01 ± 3.1017e+00**	4.9646e+01 ± 2.7835e+01	**1.2461e+00 ± 1.0657e+00**	2.5621e+01 ± 1.7331e+01	**1.0110e+01 ± 3.6648e+00**	2.5038e+01 ± 1.5229e+01
f9	**2.7236e+01 ± 7.2740e-01**	2.7492e+01 ± 8.4798e-01	**2.0814e+01 ± 1.9752e+00**	3.0395e+01 ± 1.4933e+01	2.6844e+01 ± 1.4470e+01	**2.5747e+01 ± 1.4084e+00**
f11	1.1984e+02 ± 1.9751e+01	**5.7813e+01 ± 1.4567e+01**	5.9507e+00 ± 3.3361e+00	**2.4119e+00 ± 1.4160e+00**	**3.9809e-01 ± 2.3267e-01**	7.7133e-01 ± 3.2137e-01
f12	**6.4202e-01 ± 1.0055e+00**	2.3592e+00 ± 2.6143e+00	**1.8149e+00 ± 2.9906e+00**	1.0379e+04 ± 2.7526e+04	**3.0245e+00 ± 4.5186e+00**	9.9257e+00 ± 1.9776e+01
f13	**2.4200e+00 ± 2.9217e+00**	8.4107e+00 ± 1.2720e+01	**2.6380e-01 ± 3.7061e-01**	2.4583e+01 ± 3.5764e+01	7.1866e+00 ± 8.8890e+00	**4.6516e+00 ± 5.0239e+00**
f14	**2.3478e-03 ± 4.1513e-04**	2.4340e-03 ± 6.9208e-04	**7.6400e-05 ± 1.7041e-05**	4.5520e-03 ± 1.2540e-02	**2.1847e-04 ± 4.3629e-05**	6.5608e-04 ± 8.0042e-05
f16	**1.3954e+01 ± 3.3080e+00**	1.7148e+01 ± 4.3513e+00	**1.0940e+01 ± 8.3082e+00**	1.6667e+01 ± 8.3433e+00	4.6111e+00 ± 1.6060e+00	**2.8337e+00 ± 1.6970e+00**
f17	**2.8123e-02 ± 2.4036e-02**	1.0297e-01 ± 5.6067e-02	**1.4907e-02 ± 2.0704e-02**	2.3217e-01 ± 2.2622e-01	1.0650e+00 ± 4.7242e-01	**4.2668e-01 ± 3.2249e-01**
f18	**3.0281e-01 ± 8.8656e-02**	7.9195e-01 ± 6.4349e-01	**2.3498e-01 ± 1.5333e-01**	9.2068e-01 ± 5.1392e-01	3.7183e+00 ± 1.7225e+00	**3.0491e+00 ± 2.0895e+00**
f19	4.8468e+00 ± 3.3242e-01	**4.5757e+00 ± 5.2172e-01**	5.0401e+00 ± 3.7062e-01	**4.9410e+00 ± 2.8360e-01**	**1.7610e+00 ± 5.0988e-01**	4.6573e+00 ± 4.1360e-01
f21	**2.0830e+00 ± 2.3032e+00**	6.3242e+00 ± 6.3516e+00	**2.5571e+00 ± 2.6250e+00**	5.4393e+00 ± 5.2590e+00	**5.8533e+00 ± 5.7619e+00**	9.1549e+00 ± 8.1228e+00
f22	**5.2821e+00 ± 5.0408e+00**	8.9384e+00 ± 1.0774e+01	**3.3952e+00 ± 3.1125e+00**	9.2652e+00 ± 9.0434e+00	**1.0480e+01 ± 1.0667e+01**	1.1099e+01 ± 1.1002e+01
f23	2.3991e+00 ± 2.2698e-01	**1.9655e+00 ± 3.1610e-01**	**2.4876e+00 ± 3.7048e-01**	2.4984e+00 ± 2.5580e-01	**7.3317e-01 ± 2.2159e-01**	2.0445e+00 ± 3.8059e-01
f24	1.9499e+02 ± 8.3644e+00	**1.3721e+02 ± 2.5477e+01**	**1.8465e+02 ± 4.4532e+01**	1.9464e+02 ± 1.4760e+01	**8.8574e+01 ± 2.0867e+01**	1.9051e+02 ± 2.5754e+01
Average-Ranking	1.32	1.63	1.11	1.84	1.32	1.58
Wilcoxon Test	-	5-9-5	-	12-6-1	-	8-7-4

TABLE V: The results of comparing PPTA instance with original test algorithm in CEC2005 benchmark over 15 runs including the average best fitness and standard deviation shown as AVR ± STD. The boldface figures are the best fitness between two algorithms in each test problem.

Problems	GA		DE		PSO	
	GA-PPTA	GA-Default	DE-PPTA	DE-Default	PSO-PPTA	PSO-Default
f3	5.3423e+06 ± 1.4106e+06	**4.1932e+06 ± 2.8881e+06**	1.2164e+06 ± 5.5069e+05	**3.1916e+05 ± 1.5064e+05**	**9.9410e+05 ± 6.7446e+05**	1.9506e+06 ± 6.8260e+05
f4	5.7691e+03 ± 2.6113e+03	**5.3280e+03 ± 2.5487e+03**	2.8003e+02 ± 2.1700e+02	**6.2936e+01 ± 2.0412e+02**	1.0710e+02 ± 5.6632e+01	**2.9289e+01 ± 2.6077e+01**
f5	**2.5907e+03 ± 5.5534e+02**	3.9035e+03 ± 5.0190e+02	**5.3387e+02 ± 3.4683e+02**	2.6498e+03 ± 6.4260e+02	2.6531e+03 ± 7.7626e+02	**2.6376e+03 ± 1.0516e+03**
f7	**4.6963e+03 ± 6.4311e-13**	4.6963e+03 ± 5.9540e-13	4.6963e+03 ± 4.8615e-13	**4.6963e+03 ± 5.9540e-13**	**1.2672e+04 ± 4.4787e+02**	1.2672e+04 ± 4.4787e+02
f8	**2.0540e+01 ± 1.4625e-01**	2.0901e+01 ± 7.0198e-02	**2.0865e+01 ± 9.5074e-02**	2.0958e+01 ± 5.4680e-02	**2.0126e+01 ± 5.3124e-02**	2.0894e+01 ± 8.2810e-02
f9	4.7617e+00 ± 1.4204e+00	**1.3269e-01 ± 3.5015e-01**	**6.6331e-02 ± 2.5690e-01**	2.2965e+01 ± 7.0217e+00	**5.6779e+01 ± 1.3753e+01**	6.8586e+01 ± 2.1155e+01
f10	**5.1186e+01 ± 1.2520e+01**	5.2508e+01 ± 2.2521e+01	7.7741e+01 ± 1.4423e+01	**4.2496e+01 ± 1.5422e+01**	7.6612e+01 ± 2.3019e+01	**7.6175e+01 ± 4.3265e+01**
f11	**2.4181e+01 ± 3.6630e+00**	3.0620e+01 ± 3.3841e+00	1.8455e+01 ± 4.1337e+00	**1.5630e+01 ± 6.3484e+00**	**1.4519e+01 ± 2.0604e+00**	1.4919e+01 ± 2.5181e+00
f12	**6.7454e+03 ± 3.2518e+03**	1.1076e+04 ± 5.6851e+03	**4.7289e+03 ± 5.2990e+03**	5.5502e+03 ± 4.8715e+03	**3.8205e+03 ± 2.9206e+03**	9.5885e+03 ± 9.8688e+03
f14	**1.2813e+01 ± 2.5906e-01**	1.2914e+01 ± 6.4411e-01	**1.2531e+01 ± 4.1973e-01**	1.2898e+01 ± 3.1202e-01	**1.2192e+01 ± 3.7661e-01**	1.2233e+01 ± 3.6329e-01
f16	**7.8441e+01 ± 1.7081e+01**	1.3443e+02 ± 1.3141e+02	**8.7705e+01 ± 1.1210e+01**	1.5646e+02 ± 1.6542e+02	**3.5098e+02 ± 2.2490e+02**	3.6560e+02 ± 2.2267e+02
f17	**1.1414e+02 ± 4.4157e+01**	1.8082e+02 ± 1.4580e+02	**1.1038e+02 ± 2.1228e+01**	1.5656e+02 ± 1.6455e+02	**3.0268e+02 ± 1.8932e+02**	3.5129e+02 ± 1.8886e+02
f18	**9.0697e+02 ± 1.1405e+00**	9.2553e+02 ± 7.1451e+00	**9.0403e+02 ± 4.5061e-01**	9.1310e+02 ± 1.4103e+01	8.2912e+02 ± 2.3554e+00	**8.2754e+02 ± 2.0672e+00**
f19	**9.0755e+02 ± 1.5729e+00**	9.2271e+02 ± 6.3091e+00	**9.0398e+02 ± 4.0834e-01**	9.1008e+02 ± 3.8013e+00	8.2800e+02 ± 1.6733e+00	**8.2715e+02 ± 1.8558e+00**
f20	**9.0689e+02 ± 1.0076e+00**	9.2408e+02 ± 5.3431e+00	**9.0399e+02 ± 4.0579e-01**	9.1008e+02 ± 3.8013e+00	8.2816e+02 ± 2.1293e+00	**8.2715e+02 ± 1.8558e+00**
f21	**5.0000e+02 ± 1.4412e-13**	5.0000e+02 ± 1.7322e-13	**5.0000e+02 ± 1.4412e-13**	5.4376e+02 ± 1.0798e+02	**6.1088e+02 ± 2.0621e+02**	7.1422e+02 ± 2.3579e+02
f22	**9.1470e+02 ± 1.9350e+01**	9.5137e+02 ± 1.2237e+01	**8.6809e+02 ± 1.9578e+01**	8.8930e+02 ± 1.9615e+01	5.1669e+02 ± 1.5146e+00	**5.1662e+02 ± 2.0181e+00**
f23	**5.3416e+02 ± 3.6591e-04**	5.3417e+02 ± 1.2828e-02	**5.3416e+02 ± 3.8074e-04**	6.3605e+02 ± 1.7532e+02	6.7839e+02 ± 2.2030e+02	**6.0853e+02 ± 1.5460e+02**
f24	**2.0000e+02 ± 2.9419e-14**	2.0000e+02 ± 3.7606e-06	**2.0000e+02 ± 1.6103e-12**	2.5119e+02 ± 1.9825e+02	2.8309e+02 ± 2.6501e+02	**2.7951e+02 ± 2.6493e+02**
f25	**1.6117e+03 ± 3.5998e+00**	1.6128e+03 ± 7.4134e+00	**1.6262e+03 ± 6.7518e+00**	1.6269e+03 ± 5.7205e+00	**2.0152e+03 ± 4.6354e+01**	2.0152e+03 ± 4.6354e+01
Average-Ranking	1.15	1.85	1.25	1.75	1.45	1.55
Wilcoxon Test	-	10-8-2	-	10-6-4	-	3-14-3

TABLE VI: The Wilcoxon test result for each type of problems. The 'win-draw-lose' represents the PPTA instance is superior, not significantly different and inferior to the default algorithm.

Benchmark	Problem Types	GA	DE	PSO
BBOB	TB1	0-2-2	3-1-0	2-2-0
	TB2	2-1-0	3-0-0	1-0-2
	TB3	0-3-1	3-0-1	2-2-0
	TB4	3-1-0	2-2-0	1-1-2
	TB5	0-2-2	1-3-0	2-2-0
CEC2005	TC1	1-1-1	1-0-2	1-1-1
	TC2	4-1-1	2-2-2	2-4-0
	TC3	0-1-0	1-0-0	0-1-0
	TC4	5-5-0	6-4-0	0-8-2

An obvious advantage of transfer tuned-parameters to a similar problem is to reduce the time for parameter-tuning. It doesn't need to evaluate the parameter configuration in all functions but only on one instance for each problem category. To illustrate the efficiency of PPTA, we record the running time of each testing algorithm in each test functions. Take GA as an example, the Eq. (1) and Eq. (2) for BBOB2009 benchmark are

$$T_{s0} = 1605s, T_{s1} = 300s$$

and for CEC2005 benchmark are

$$T_{s0} = 11595s, T_{s1} = 1110s$$

From the result, it is obvious the training time is reduced significantly when we only tune parameters on one instance for each problem type.

3) Optimization Risk: The motivation for proposing PPTA is to reduce the optimization risk for the single parameter-tuned algorithm. Therefore, we employ the metric from the work in [10] to analyse the risk of the proposed framework. It analyses the risk by comparing the quality of solutions obtained in a set of test problems with several runs. The risk

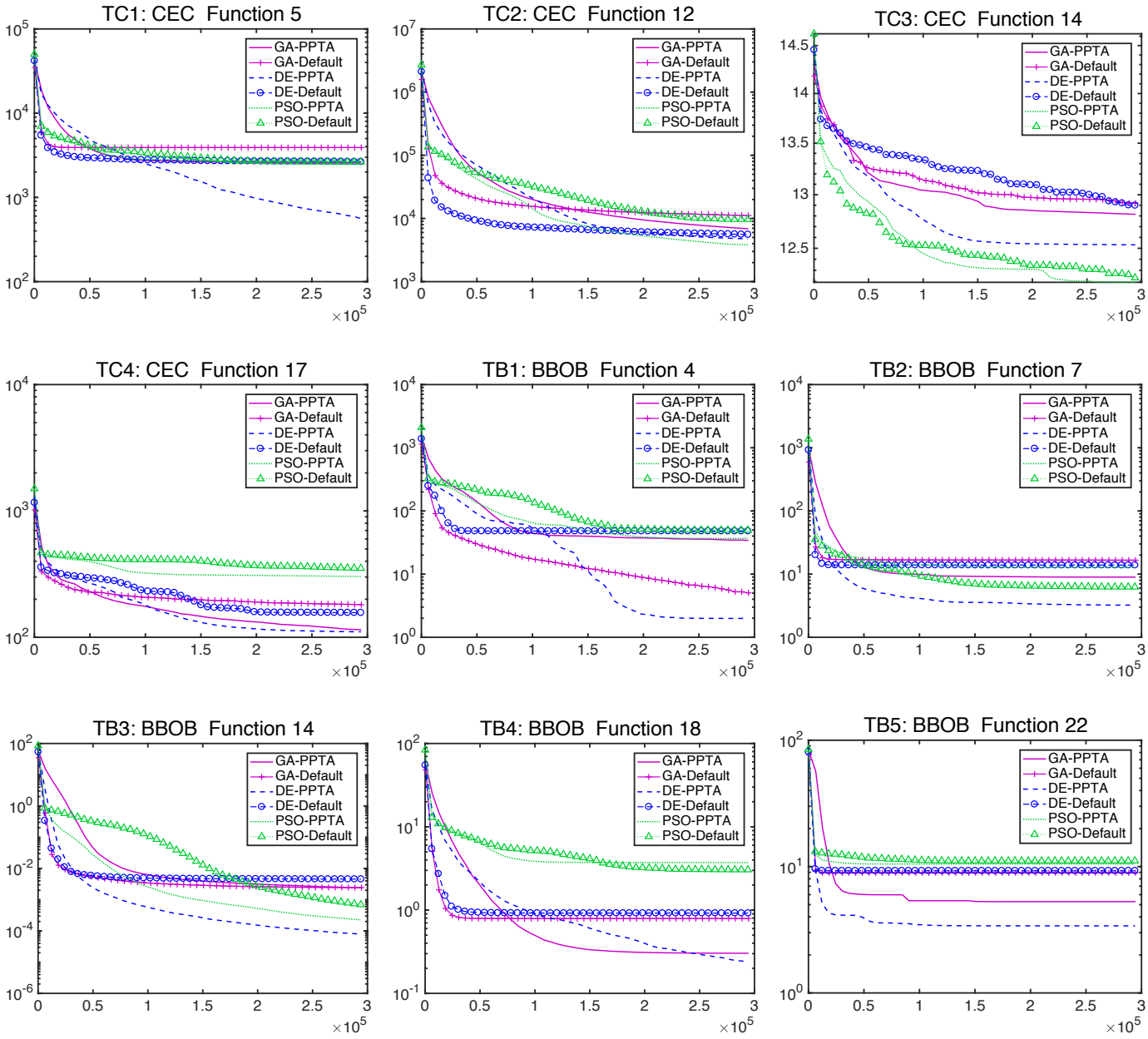

Fig. 2: Convergence curves of PPTA instances and default parameter value algorithms on each type of problems. Lines without symbols are PPTA instances.

analysis result is presented in Table VII. The first figure in each cell represents the probability that PPTA outperforms single algorithm. And the second one represents the probability that the single algorithm outperforms PPTA.

From the table, we can easily find that every PPTA instance obtains a higher probability than the default parameter value algorithm in two benchmark functions. The default algorithm's parameters might be tuned by researchers' experience or in a complete benchmark containing many kinds of problems so that it has a high probability in failing in some problems. But PPTA framework combines some suitable parameter con-

TABLE VII: The comparison risk of PPTA instance and default test algorithm. The two figures in each cell stand for the probabilities that PPTA and single algorithm outperformed each other.

		BBOB	CEC2005
		Default	
GA	PPTA	0.56-0.44	0.73-0.27
DE		0.80-0.20	0.63-0.37
PSO		0.69-0.31	0.52-0.48

figurations obtained in different kinds of problem so that its performance is more robust and perform much better than the default algorithm in some problems.

V. Conclusion

In this paper, we consider the optimization risk for single parameter-tuned algorithm and the computationally expensive problem for tuning the algorithm parameter. Motivated from the algorithm portfolio and fitness landscape analysis, we combined parameter-tuned algorithm with algorithm portfolio technique to propose a new framework named portfolio for parameter-tuned algorithm (PPTA). In our PPTA framework, the SPO and PAP algorithm are employed, which are both very effective in parameter-tuning and algorithm portfolio area. PPTA framework firstly selects several problems which have different characteristics, and uses SPO algorithm to obtain the best parameter configuration on one training instance for each problem category, respectively. Then, PPTA uses an algorithm portfolio technique to combine several instantiations of the same algorithm with different parameter configurations. The portfolio instance is used to solve optimization problems. We test the PPTA framework in three basic algorithms, including GA, PSO and DE, on two very popular benchmark functions: CEC2005 and BBOB2009. The experimental results showed that the algorithm equipped with PPTA framework performs much better than the algorithm with default parameter. Additionally, we also confirm the effectiveness of transferring tuned parameters to the same type of problem and the significant decrease of running time for parameter tuning when we apply the PPTA framework. The optimization risk is also analysed through the experimental results and it shows that the optimization risk is reduced significantly after using the PPTA framework.

The PPTA framework could enhance the performance of single parameter-tuned algorithm but the ability is also limited when the parameters are not very important to the algorithm's performance. So, how to detect the parameters' importance can be a significant future work which is not only suitable for this work but also makes sense to the parameter tuning area.

ACKNOWLEDGEMENT

This work was supported by National Key R&D Program of China (Grant No. 2017YFC0804003), National Natural Science Foundation of China (Grant No. 61976111), the Program for Guangdong Introducing Innovative and Enterpreneurial Teams (Grant No. 2017ZT07X386), Shenzhen Peacock Plan (Grant No. KQTD2016112514355531) the Science and Technology Innovation Committee Foundation of Shenzhen (Grant Nos. JCYJ20170817112421757 and JCYJ20180504165652917) and the Program for University Key Laboratory of Guangdong Province (Grant No. 2017KSYS008).

References

[1] F. Dobslaw, "Recent development in automatic parameter tuning for metaheuristics," in *Proceedings of the 19th Annual Conference of Doctoral Students-WDS 2010*, 2010.

[2] E. Montero, M. C. Riff, and B. Neveu, "A beginner's guide to tuning methods," *Applied Soft Computing*, vol. 17, pp. 39–51, 2014.

[3] C. Huang, Y. Li, and X. Yao, "A survey of automatic parameter tuning methods for metaheuristics," *IEEE Transactions on Evolutionary Computation*, pp. 1–1, 2019.

[4] M. Birattari, T. Stützle, L. Paquete, and K. Varrentrapp, "A racing algorithm for configuring metaheuristics," in *Proceedings of the 4th Annual Conference on Genetic and Evolutionary Computation*. Morgan Kaufmann Publishers Inc., 2002, pp. 11–18.

[5] F. Hutter, H. H. Hoos, and T. Stützle, "Automatic algorithm configuration based on local search," in *AAAI*, vol. 7, 2007, pp. 1152–1157.

[6] Z. Yuan, M. A. M. De Oca, M. Birattari, and T. Stützle, "Continuous optimization algorithms for tuning real and integer parameters of swarm intelligence algorithms," *Swarm Intelligence*, vol. 6, no. 1, pp. 49–75, 2012.

[7] D. R. Jones, M. Schonlau, and W. J. Welch, "Efficient global optimization of expensive black-box functions," *Journal of Global optimization*, vol. 13, no. 4, pp. 455–492, 1998.

[8] T. Bartz-Beielstein, C. W. Lasarczyk, and M. Preuß, "Sequential parameter optimization," in *2005 IEEE congress on evolutionary computation*, vol. 1. IEEE, 2005, pp. 773–780.

[9] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Trans. Evolutionary Computation*, vol. 1, no. 1, pp. 67–82, 1997.

[10] F. Peng, K. Tang, G. Chen, and X. Yao, "Population-based algorithm portfolios for numerical optimization," *IEEE Transactions on Evolutionary Computation*, vol. 14, no. 5, pp. 782–800, 2010.

[11] S. Y. Yuen, C. K. Chow, X. Zhang, and Y. Lou, "Which algorithm should i choose: An evolutionary algorithm portfolio approach," *Applied Soft Computing*, vol. 40, pp. 654–673, 2016.

[12] H. Tong, J. Liu, and X. Yao, "Algorithm portfolio for individual-based surrogate-assisted evolutionary algorithms," in *Proceedings of the Genetic and Evolutionary Computation Conference*, ser. GECCO '19. ACM, 2019, pp. 943–950.

[13] L. Xu, F. Hutter, H. H. Hoos, and K. Leytonbrown, "Satzilla: portfolio-based algorithm selection for sat," *Journal of Artificial Intelligence Research*, vol. 32, no. 1, pp. 565–606, 2008.

[14] M. Cauwet, J. Liu, B. Roziere, and O. Teytaud, "Algorithm portfolios for noisy optimization," *Annals of Mathematics and Artificial Intelligence*, vol. 76, no. 1, pp. 143–172, 2016.

[15] M.-L. Cauwet, J. Liu, and O. Teytaud, "Algorithm portfolios for noisy optimization: Compare solvers early," in *International Conference on Learning and Intelligent Optimization*. Springer, 2014, pp. 1–15.

[16] P. A. Consoli, Y. Mei, L. L. Minku, and X. Yao, "Dynamic selection of evolutionary operators based on online learning and fitness landscape analysis," *Soft Computing*, vol. 20, no. 10, pp. 3889–3914, 2016.

[17] F. A. C. Viana, "A tutorial on latin hypercube design of experiments," *Quality and Reliability Eng. Int.*, vol. 32, no. 5, pp. 1975–1985, 2016.

[18] P. N. Suganthan, N. Hansen, J. J. Liang, K. Deb, Y.-P. Chen, A. Auger, and S. Tiwari, "Problem definitions and evaluation criteria for the cec 2005 special session on real-parameter optimization," *KanGAL report*, vol. 2005005, p. 2005, 2005.

[19] S. Finck, N. Hansen, R. Ros, and A. Auger, "Real-parameter black-box optimization benchmarking 2009: Presentation of the noiseless functions," Citeseer, Tech. Rep., 2010.

[20] D. Whitley, "A genetic algorithm tutorial," *Statistics and Computing*, vol. 4, no. 2, pp. 65–85, 1994.

[21] R. Storn and K. Price, "Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[22] Y. Shi and R. Eberhart, "A modified particle swarm optimizer," in *1998 IEEE International Conference on Evolutionary Computation Proceedings.* IEEE, 1998, pp. 69–73.