ISSN: 2664-9640

A Metamodel-Assisted Particle Swarm Optimization for Expensive Real-World Problems

Chongzheng Na, Huixin Liu *, He Li

School of Information and Control, Shenyang Institute of Technology, China

* Corresponding Author

Abstract

In the recent decades, researches of metamodel-assisted optimization algorithms, also known as surrogate-assisted optimization algorithms, were rapidlly proposed for solving many kinds of expensive real-world problems, which are generally non-differential, multi-modal and noisy. Therefore, evolutionary algorithms (EAs), which are gradient-free and black-box friendly, are likely to be the promising approaches. Nevertheless, there are still challenges that all these algorithms require a big number of evaluation calls before achieving solution close to the global optimum. As a result, a bunch of approximation-based algorithms were proposed with the propose of avoiding the costly evaluations. In this paper, we present a brief investigation of these approximation-based researches. And a metamodel-assisted dual swarm particle swarm optimization is proposed. Experimental results obtained on benchmark functions from Black-box optimization benchmarking (BBOB) are presented.

Keywords

Metamodel-assisted optimization, Evolutionary algorithms, Particle swarm optimization, Real-world problem, Black-box optimization benchmarking.

1. Introduction

Typically, the goal of real-world engineering optimization problems is to carry out the optimum designs, parameters or characteristics under specified design criteria and time limit. Many of them, such as manufacturing engineering [1], food industry [2], steel casting [3] and drug design [4], require significant resources for evaluations which are normally black-boxing, experiment-based or simulation-based. Black-box problems are a group of problems with a least one unknown function that given a list of inputs, corresponding outputs can be obtained without knowing its internal structure or expression [5]. Moreover, optimization problems containing functions that discontinue, non-smooth, noisy or lack of gradient information are generally considered as black-boxing due to they cannot apply traditional gradient-based mathematical programming methods[6]. Evolutionary algorithms are widely used gradient-free methods to solve black-box problems. Unfortunately, these kinds of algorithms require a big number of evaluation calls before achieving solution close to the global optimum, which is generally unaffordable if the problems are experiment-based or simulation-based as mentioned above. For instance, photobioreactor design [7], wind energy evaluation [8] and agricultural industry design [9], which are computational fluid dynamics (CFDs) based, cost minutes or hours to run the expensive computationally analysis code for a single evaluation call. Hence, metamodel-assisted algorithms, also known as surrogate-assisted algorithms, were proposed, focusing to reduce computational time in evolutionary optimizations of real-world expensive problems by using approximation functions along with costly real functions.

In recent years, various researches about metamodel-assisted algorithms were reported. [10] proposed a framework using artificial neural network (ANN) and covariance matrix

ISSN: 2664-9640

adaptation evolution strategy (CMA-ES). [11] employed radial basis function (RBF) based hybrid genetic algorithm (GA) for solving high-fidelity engineering design problems. [12] combined gaussian process regression (GPR) and differential evolution (DE) for medium scale expensive optimization problems. [13] applied fitness inheritance surrogate model to real-coded genetic algorithms. [14] introduced a method with kriging and particle swarm optimization. [15] introduced a two-layer surrogate-assisted particle swarm optimization algorithm. [16] presented a gaussian process regression and radial basis function hybrid assisted evolutionary algorithm. [17] introduced self-adaptive evolution strategies bases local support vector machine (SVM) constraint surrogate models.

These researches proved that metamodel-assisted methods greatly increase the power of the optimizations. The same number of evaluations could be done to get closer to the global optimum or to solve more difficult optimization problems.[18] For all that there are encouraging efforts, metamodel-assisted algorithms have their own disadvantage, which is known as curse of uncertainty [19]. Researchers suggest that the approximation error of approximation models may lead algorithms converge to false optima. Still and all, the uncertainty does not always harm, due to its insensitivity to multimodal or noisy landscape of the complex problem. Hence, making use of the advantage and overcoming the disadvantage are constantly research direction of this topic.

2. Particle Swarm Optimization

Particle swarm optimization, which was firstly proposed in 1995 [20], is a swarm theory based optimization algorithm for solving continuous nonlinear problems, which can be define as follow:

$$minimize\ f(\boldsymbol{x}) \tag{1}$$

$$s.t.\ \boldsymbol{x}_{lb} \le \boldsymbol{x} \le \boldsymbol{x}_{ub} \tag{2}$$

where $\boldsymbol{x}$ is the decision variable vector, $f(\boldsymbol{x})$ is the objective function, $\boldsymbol{x}_{lb}$ and $\boldsymbol{x}_{ub}$ are the upper and lower boundary vectors of $\boldsymbol{x}$. In this algorithm, a group of particles start randomly in the search space. Each of the particles has its own position and velocity and is guided by two bests: one is the particle best (pbest), which records the best position that one particle has found, the other is the global best (gbest), which record the best position that the whole swarm has found. Updating rules at iteration $t \to t+1$ can be defined as:

$$\boldsymbol{v}_i^{t+1} = \boldsymbol{v}_i^t + c_1 r_1(\boldsymbol{pbest}_i^t - x_i^t) + c_2 r_2(\boldsymbol{gbest}^t - \boldsymbol{x}_i^t) \tag{3}$$

$$\boldsymbol{x}_i^{t+1} = \boldsymbol{x}_i^t + \boldsymbol{v}_i^{t+1} \tag{4}$$

where $\boldsymbol{x}_i^t$ is the position of particle i at iteration t, $\boldsymbol{v}_i^t$ is the velocity of particle i at iteration t, $\boldsymbol{pbest}_i^t$ is the best historical position of particle i before iteration t, $\boldsymbol{gbest}^t$ is the best historical position of the whole swarm before iteration t, c_1 and c_2 are the acceleration coefficients, r_1 and r_2 are random numbers in the range (0,1].

Algorithm 1: The original particle swarm optimization

Initialize particle position $\boldsymbol{x}$ and velocity $\boldsymbol{v}$ randomly within boundary
For each particle i in the swarm
  $\boldsymbol{Pbest}_i = \boldsymbol{x}_i$
End For
While not stopping criterion is not met
  For each particle i in the swarm
    Evaluate particle i
    If i is the new particle best
      update $\boldsymbol{Pbest}_i = \boldsymbol{x}_i$

ISSN: 2664-9640

End IF
If i is the new global best
update $\boldsymbol{Gbest} = \boldsymbol{x_i}$
End IF
Update velocity using (3)
Update position using (4)
End For
End While

In spite of simplicity and efficiency, the original PSO has its own flaws that the search sometimes traps into local optimum, and lack of convergence prove. Hence, several variants of PSO have been proposed, such as PSO with neighborhood operator to improve its exploration ability[21], inertia weight and constriction factor based method to insure convergence[22] [23] [24]. The chosen algorithm in this paper is PSO with inertia weight, which particles updating criterion can be altered as:

$$\boldsymbol{v}_i^{t+1} = w^t \boldsymbol{v}_i^t + c_1 r_1 (\boldsymbol{pbest}_i^t - x_i^t) + c_2 r_2 (\boldsymbol{gbest}^t - \boldsymbol{x}_i^t) \tag{5}$$

$$\boldsymbol{x}_i^{t+1} = \boldsymbol{x}_i^t + \boldsymbol{v}_i^{t+1} \tag{6}$$

where w^t is inertia weight. Normally, the inertia weight changes linearly[25], reducing from w_{max} to w_{min} over the search iterations:

$$w^t = w_{max} - \frac{t(w_{max} - w_{min})}{t_{max}} \tag{7}$$

where t_{max} is the designed max iterations. w_{min} and w_{max} are usually set to 0.4 and 0.9.

3. Support Vector Regression

Support vector machine (SVM) is a kernel-based learning method for solving classification and regression [26].The regression form of SVM are proposed in 1998, which is called ε-SVR[27]. The goal is to find a function $f(x)$ that has at most ε deviation from the actually obtained targets y_i for all the training data, and at the same time is as flat as possible[28]. The standard form of support vector regression is[29]:

$$\begin{aligned} \min_{w,b,\xi,\xi^*} \quad & \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + C\sum_{i=1}^{l}\xi_i + C\sum_{i=1}^{l}\xi_i^* \\ \text{subject to} \quad & \boldsymbol{w}^T\phi(\boldsymbol{x}_i) + b - z_i \le \epsilon + \xi_i \\ & z_i - \boldsymbol{w}^T\phi(\boldsymbol{x}_i) - b \le \epsilon + \xi_i^*, \\ & \xi_i, \xi_i^* \ge 0, i = 1, \dots, l. \end{aligned} \tag{8}$$

The dual problem is

$$\begin{aligned} \min_{\alpha,\alpha^*} \quad & \frac{1}{2}(\boldsymbol{\alpha} - \boldsymbol{\alpha}^*)^T\boldsymbol{Q}(\boldsymbol{\alpha} - \boldsymbol{\alpha}^*) + \epsilon\sum_{i=1}^{l}(\alpha_i + \alpha_i^*) + \sum_{i=1}^{l} z_i(\alpha_i - \alpha_i^*) \\ \text{subject to} \quad & \boldsymbol{e}^T(\boldsymbol{\alpha} - \boldsymbol{\alpha}^*) = 0 \\ & 0 \le \alpha_i, \alpha_i^* \le C, i = 1, \dots, l \end{aligned} \tag{9}$$

$\boldsymbol{Q}$ is a matrix, $\boldsymbol{Q_{ij}} = K(\boldsymbol{x_i}, \boldsymbol{x_j}) = \phi(\boldsymbol{x}_i)^T \phi(\boldsymbol{x}_j)$, and $K(x, x')$ is the kernel function, which is supposed to be given priority. The most commonly used kernel function is the Gaussian Kernel:

$$K(\boldsymbol{x}, \boldsymbol{x}') = e^{-\frac{\|x - x'\|^2}{2\sigma^2}} \tag{10}$$

After solving (9) the approximate function can be given:

ISSN: 2664-9640

$$\hat{f}(x) = \sum_{i=1}^{l} (-a_i + a_i^*) K(x_i, x) + b \tag{11}$$

l is the total sample number.

4. Meta-model Assisted Dual Swarm Particle Swarm Optimization (MADSPSO)

4.1. The Framework of MADSPSO

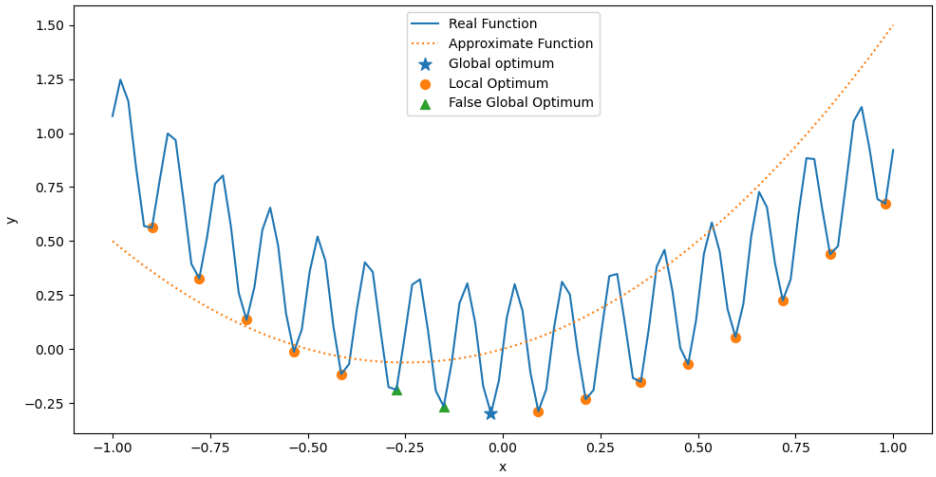

Figure 1: 2D case: meta-models are insensitivity to tiny local optimum, but could lead search trapping into unexpected local optimum

Even though meta-model assisted algorithm could strikingly decrease real functions evaluated of expensive problems, the way to retain exploration ability is constantly an open issue. The uncertainty produced by meta-models brings both positive and negative efforts [19]. For instance, for many noisy and multi-model problems, if several particles of PSOs are guided by meta-models, the particles may smooth the tiny local optimum out, boosting exploration in the early stage, due to the uncertainty is insensitivity to the tiny changes of the original functions. At the same time, the uncertainty of meta-models also creates false optimum, which could lead the search trapping into unexpected local optimum. The interested reader is referred to the relevant researches [11][19].

The proposed algorithm includes two swarms, one is the classic swarm, which is similar to the original PSO, evaluating the real function, the other is called the surrogate swarm, which is guided by the meta-model approximate function. Like other meta-model assisted algorithms, the proposed method is divided into two phases. In the first phase, normally called initialization phase, the classic swarm initializes as same as the original PSO algorithm with inertia weight set to w_{max} , position $\boldsymbol{x}$ and velocity $\boldsymbol{v}$ randomly within boundary, and records the real-function evaluated $(x, f(x))$ before the global database D_g meet the minimum requirement. In the main phase, along with the classic swarm, the surrogate swarm start to run bases the meta-model constructed using the local database D_l ,which is a subset of the global database D_g under some certain criteria. The initial position $\boldsymbol{x}$ and velocity $\boldsymbol{v}$ of the surrogate swarm are following distribution of the classic swarm. The dual swarms run parallelly and asynchronously. While the classic swarm iterates once, the surrogate swarm iterates multiple times, contributing its best swarms to the classic swarm. The validity of the meta-model can is verified by using the current classic swarm particles. Specifically, for various real-world application that priori knowledge or history data is available, the algorithm can start at second phase.

ISSN: 2664-9640

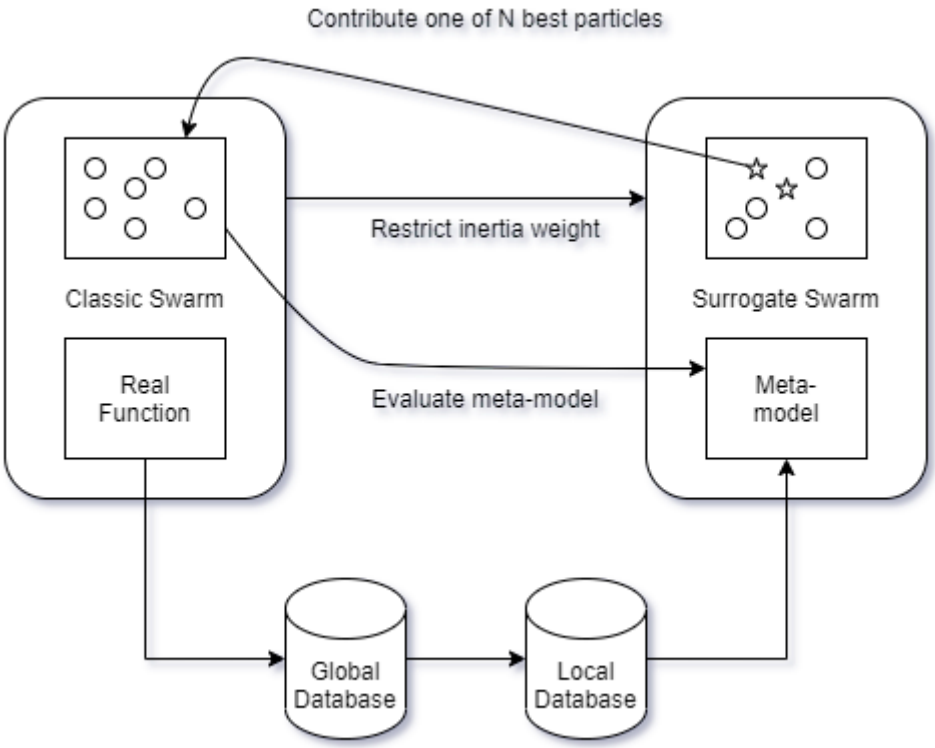

Figure 2: Structure of the MADSPSO

4.2. Database Management

All the real functions evaluated are supposed to be recorded in the global database. The meta-model constructed with global database is called global model. Though has the remarkable ability of exploring the decision space, the global model also has negative impacts. As the search proceeding, approximation accuracy of the global model could be an issue due to the data samples that the global model used distribute in a wide search space. Likewise, getting more data samples generally means more time to train meta-model, which could be geometrically expanding in many approximation strategies. Hence, a subset of the global database is selected in various researches. The meta-model using this more 'local' database is defined as local model. The proposed algorithm uses the following strategy:

$$xmin_d^t = \min_{i \in I}(x_{id}^t) \tag{12}$$

$$xmax_d^t = \max_{i \in I}(x_{id}^t) \tag{13}$$

Where I is the classic swarm set. The boundary of local database is determined by $xmin_d^t$ and $xmax_d^t$ of last m generations.

$$ldeta_d^t = \max_{t'=t-m+1,t-m+2,\dots,t}(xmax_d^{t'}) - \min_{t'=t-m+1,t-m+2,\dots,t}(xmin_d^{t'}) \tag{14}$$

$$lmin_d^t = \min_{t'=t-m+1,t-m+2,\dots,t}(xmin_d^{t'}) - \alpha(ldeta_d^t) \tag{15}$$

$$lmax_d^t = \max_{t'=t-m+1,t-m+2,\dots,t} xmax_d^{t'} + \alpha(ldeta_d^t) \tag{16}$$

Where α is a constant that normally in the range [0,2]. Finally, the local database D_l can be defined as:

$$D_l^t = \{(x, f(x)) | (x, f(x)) \in D_g^t, lmin_d^t < x_d < lmax_d^t\} \tag{17}$$

A larger α and a larger m mean more 'global' the local model is. Occasionally, a could be relax until the samples of D_l^t reach minimum requirement n_l.

$$a_{new} = \tau a_{old}$$

ISSN: 2664-9640

τ could be simply set to 1.1.

4.3. Meta-model Management

As the search progressing, the differences between particles distribution and train set distribution will be getting bigger. Approximation accuracy of the meta-model is supposed to be tested. Evaluating the mean square error (MSE) is the most commonly method. Nevertheless, as [30] suggested, for ranking-based algorithms like PSOs and EAs, sometimes, approximation model that has a large approximation error but are adequately good for evolutionary search. Because in these algorithms, evolutionary behaviors such as selection or gbest alternating, only compare the ranking of solutions. Therefore, in this paper, a ranking-based error function is proposed to evaluate accuracy of the current meta-model:

$$err = \frac{2}{N(N-1)} \sum_{1<i<N} \sum_{i<j<N} \left(\left(f(x_i) < f(x_j) \right) xor \left(\hat{f}(x_i) < \hat{f}(x_j) \right) \right) \tag{18}$$

$$a \; xor \; b = \begin{cases} 0 \; if \; (a = true \; and \; b = true) \; or \; (a = false \; and \; b = false) \\ 1 \; if (a = true \; and \; b = false) \; or \; (a = false \; and \; b = true) \end{cases} \tag{19}$$

x_i and x_j are different particles in the current classic swarm. xor is the exclusive-or operator. Obviously, $err = 0$ means that the order of $f(x)$ and $\hat{f}(x)$ are exactly the same on the test set, and the meta-model are most likely trustworthy. Instead, if $err = 1$ means $f(x)$ and $\hat{f}(x)$ are in reverse order on the test set. A hyper parameter err_s is supposed to be determined in purpose of deciding whether the meta-model is still accurate enough, or a new meta-model of a more local dataset is ought to be built.

Algorithm 2: Database and Meta-model Management

Triggered when the classic swarm finished one .
Record the $(x, f(x))$,that the classic swarm newly evaluated, to global database D_g.
Calculate err using (18)(19).
IF $err > err_s$
  Calculate D_l^t using (12)-(17).
  WHILE $crad(D_l^t) < n_l$ and $crad(D_l^t) \neq crad(D_g^t)$
    Update $a_{new} = \tau a_{old}$.
    Calculate D_l^t using (12)-(17).
  END WHILE
  Train new meta-model bases new D_l^t.
END IF

5. Benchmark Results

In this section, the Rosenbrock benchmark function is chosen to evaluate the proposed algorithm. The Rosenbrock benchmark function, also referred to as the Valley or Banana function, is one of the most popular test problems for both black-boxed or gradient-based optimization algorithms. The n dimensions Rosenbrock benchmark function can be described as:

$$f(x) = \sum_{i=1}^{d-1} (100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2) \tag{20}$$

ISSN: 2664-9640

This function is unimodal, and the global minimum lies in a narrow, parabolic valley. However, even though this valley is easy to find, convergence to the minimum is difficult. The global minimum is $f(x^*) = 0$ at $x^* = (1,1,\ldots,1)$.

Table.1 and Figure.3 show the comparison between the original CPSO and the proposed algorithm (PA). Both results are the average of 20 times computations.

Table 1: Results of the original CPSO and the proposed algorithm

	Best	Worst	Average	Std
PSO(t=5000)	1.49e00	2.31e02	6.22e01	5.82e01
PA (t=5000)	5.02e00	8.80e01	8.17e00	5.70e00
PSO(t=10000)	8.66e-1	9.95e01	2.27e01	2.20e01
PA (t=10000)	3.98e00	7.06e01	5.64e00	2.31e00

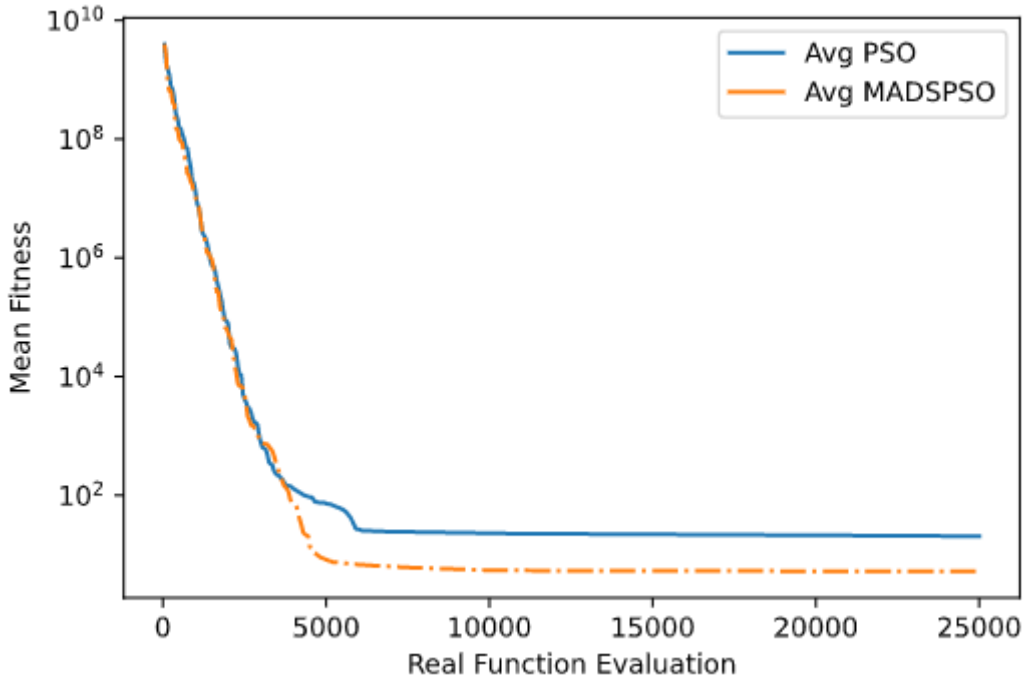

Figure 3: Average fitness comparison

The meta-model starts to work as soon as the global database has collected 2000 samples at least. However, it seems that the meta-model assisting does little influence before about 3500 real function evaluations. The reason is that at early stage of the algorithm, the particles spread in vast range of the decision space. It is difficult to collect enough samples to build accurate meta-model. The best particles that meta-model reporting, are with bigger bias than latter stage. In the proposed algorithm, those particles would be abandoned, and do no negative impact of the search process. With the algorithm progressing, the efficient local model could be built and continuously contribute solution to the main swarm. Those solutions could be used for replacing the particles in the main swarm and the search could be accelerated. Also, it is believed that the exploration ability is improved. The reason is every time when the meta-model reconstructs, a new surrogate swarm must be built basing the distribution of the main swarm particles. This process achieves the same effect as many PSO variants with restart strategy, which were proved to be effective methods to improve the exploration ability of algorithm.

ISSN: 2664-9640

6. Conclusion

The MADSPSO is proposed to solve real-world expensive optimization problems. The key feature is building a meta-model-based swarm, which continuously try to contribute good solutions to the main PSO swarm. This swarm runs separately and update itself using the computing results of the main PSO swarm, ensuring that the extra model does almost no negative impact to the searching progress. A ranking function is used to evaluate the quality of meta-model, that could avoid the negative impact of system bias, since for the separately running meta-model swarm the tendency of the fitness is more important than the exact function value. Experimental result shows the proposed algorithm could effectively accelerate the search speed and improve the exploration ability.

However, these are much work to do in the future research. The preparing stage is too long comparing the other model-based algorithm. The main swarm will not accept meta-model-based results unless they are good enough. Effective strategy is needed to do in the future work. Also, the resampling method of the surrogate swarm could be improved, many restart strategy based algorithms could be referred.

References

[1] Z. F. Wang, M. Yao, X. B. Zhang, and X. D. Wang, "Optimization Control for Solidification Process of Secondary Cooling in Continuous Casting Steel," *Appl. Mech. Mater.*, vol. 263–266, pp. 822–827, 2013.

[2] B. Xia and D.-W. Sun, "Applications of computational fluid dynamics (CFD) in the food industry: a review," *Comput. Electron. Agric.*, vol. 34, no. 1–3, pp. 5–24, 2002.

[3] X. C. Luo and C. Z. Na, "GA-CDFM Based Hybrid Optimization Method for Steelmaking Scheduling and Caster Operation," *Adv. Mater. Res.*, vol. 424–425, pp. 994–998, 2012.

[4] I. D. Kuntz, "Structure-Based Strategies for Drug Design and Discovery," *Science*, vol. 257, no. 5073, pp. 1078–1082, Aug. 1992.

[5] S. Shan and G. G. Wang, "Survey of modeling and optimization strategies to solve high-dimensional design problems with computationally-expensive black-box functions," *Struct. Multidiscip. Optim.*, vol. 41, no. 2, pp. 219–241, Mar. 2010.

[6] L. M. Rios and N. V. Sahinidis, "Derivative-free optimization: a review of algorithms and comparison of software implementations," *J. Glob. Optim.*, vol. 56, no. 3, pp. 1247–1293, 2013.

[7] A. Soman and Y. Shastri, "Optimization of novel photobioreactor design using computational fluid dynamics," *Appl. Energy*, vol. 140, pp. 246–255, Feb. 2015.

[8] A. Z. Dhunny, M. R. Lollchund, and S. D. D. V. Rughooputh, "Wind energy evaluation for a highly complex terrain using Computational Fluid Dynamics (CFD)," *Renew. Energy*, vol. 101, pp. 1–9, Feb. 2017.

[9] T. Norton, D.-W. Sun, J. Grant, R. Fallon, and V. Dodd, "Applications of computational fluid dynamics (CFD) in the modelling and design of ventilation systems in the agricultural industry: A review," *Bioresour. Technol.*, vol. 98, no. 12, pp. 2386–2414, Sep. 2007.

[10] Y. Jin, M. Olhofer, and B. Sendhoff, "A framework for evolutionary optimization with approximate fitness functions," *IEEE Trans. Evol. Comput.*, vol. 6, no. 5, pp. 481–494, Oct. 2002.

[11] Y. S. Ong, P. B. Nair, A. J. Keane, and K. W. Wong, "Surrogate-Assisted Evolutionary Optimization Frameworks for High-Fidelity Engineering Design Problems," in *Knowledge Incorporation in Evolutionary Computation*, Y. Jin, Ed. Berlin, Heidelberg: Springer, 2005, pp. 307–331. Accessed: May 20, 2021. [Online]. Available: https://doi.org/10.1007/978-3-540-44511-1_15.

[12] B. Liu, Q. Zhang, and G. G. Gielen, "A Gaussian process surrogate model assisted evolutionary algorithm for medium scale expensive optimization problems," *IEEE Trans. Evol. Comput.*, vol. 18, no. 2, pp. 180–192, 2013.

[13] L. G. Fonseca, A. C. Lemonge, and H. J. Barbosa, "A study on fitness inheritance for enhanced efficiency in real-coded genetic algorithms," in *2012 IEEE Congress on Evolutionary Computation*, 2012, pp. 1–8.

[14] M. D. Parno, T. Hemker, and K. R. Fowler, "Applicability of surrogates to improve efficiency of

particle swarm optimization for simulation-based problems," *Eng. Optim.*, vol. 44, no. 5, pp. 521–535, 2012,

[15] C. Sun, Y. Jin, J. Zeng, and Y. Yu, "A two-layer surrogate-assisted particle swarm optimization algorithm," *Soft Comput.*, vol. 19, no. 6, pp. 1461–1475, Jun. 2015.

[16] Z. Zhou, Y. S. Ong, P. B. Nair, A. J. Keane, and K. Y. Lum, "Combining Global and Local Surrogate Models to Accelerate Evolutionary Optimization," *IEEE Trans. Syst. Man Cybern. Part C Appl. Rev.*, vol. 37, no. 1, pp. 66–76, Jan. 2007.

[17] J. Poloczek and O. Kramer, "Local SVM Constraint Surrogate Models for Self-adaptive Evolution Strategies," in *KI 2013: Advances in Artificial Intelligence*, Berlin, Heidelberg, 2013, pp. 164–175.

[18] R. T. Haftka, D. Villanueva, and A. Chaudhuri, "Parallel surrogate-assisted global optimization with expensive functions – a survey," *Struct. Multidiscip. Optim.*, vol. 54, no. 1, pp. 3–13, Jul. 2016.

[19] D. Lim, Y. Jin, Y.-S. Ong, and B. Sendhoff, "Generalizing Surrogate-Assisted Evolutionary Computation," *IEEE Trans. Evol. Comput.*, vol. 14, no. 3, pp. 329–355, Jun. 2010.

[20] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, 1995, pp. 39–43.

[21] P. N. Suganthan, "Particle swarm optimiser with neighbourhood operator," in *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)*, Jul. 1999, vol. 3, pp. 1958-1962 Vol. 3.

[22] Y. Shi and R. Eberhart, "A modified particle swarm optimizer," in *1998 IEEE international conference on evolutionary computation proceedings. IEEE world congress on computational intelligence (Cat. No. 98TH8360)*, 1998, pp. 69–73.

[23] M. Clerc, "The swarm and the queen: towards a deterministic and adaptive particle swarm optimization," in *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)*, Jul. 1999, vol. 3, pp. 1951-1957 Vol. 3.

[24] R. C. Eberhart and Y. Shi, "Comparing inertia weights and constriction factors in particle swarm optimization," in *Proceedings of the 2000 congress on evolutionary computation. CEC00 (Cat. No. 00TH8512)*, 2000, vol. 1, pp. 84–88.

[25] Y. Shi and R. C. Eberhart, "Parameter selection in particle swarm optimization," in *Evolutionary Programming VII*, Berlin, Heidelberg, 1998, pp. 591–600.

[26] V. Vapnik, *The nature of statistical learning theory*. Springer science & business media, 2013.

[27] V. N. Vapnik, "An overview of statistical learning theory," *IEEE Trans. Neural Netw.*, vol. 10, no. 5, pp. 988–999, 1999.

[28] A. J. Smola and B. Schölkopf, "A tutorial on support vector regression," *Stat. Comput.*, vol. 14, no. 3, pp. 199–222, 2004.

[29] C.-C. Chang and C.-J. Lin, "LIBSVM: a library for support vector machines," *ACM Trans. Intell. Syst. Technol. TIST*, vol. 2, no. 3, pp. 1–27, 2011.

[30] Y. Jin, "Surrogate-assisted evolutionary computation: Recent advances and future challenges," *Swarm Evol. Comput.*, vol. 1, no. 2, pp. 61–70, 2011.