Neural network-based multiobjective optimization algorithm for nonlinear beam dynamics

Jinyu Wan,[1,2] Paul Chu,[1,2,*] and Yi Jiao[1,2,†]

[1]*Key Laboratory of Particle Acceleration Physics and Technology, Institute of High Energy Physics, Chinese Academy of Sciences, Beijing 100049, China*

[2]*University of Chinese Academy of Sciences, Beijing 100049, China*

(Received 14 January 2020; accepted 6 August 2020; published 17 August 2020)

Multiobjective genetic algorithms (MOGAs) have proven to be powerful in solving multiobjective problems in the accelerator field. Nevertheless, for explorative problems that have many variables and local optima, the performance of MOGAs is not always satisfactory, especially when a small population size is used due to practical limitations, e.g., limited computing resources. To deal with this challenge, in this paper an enhanced MOGA, neural network-based MOGA (NBMOGA), is proposed. In this method, the data produced with the standard MOGA are used to train a neural network. The neural network is fast to produce a large pool of objective function estimates, with sufficiently high accuracy. A subset of the most competitive estimates is selected to form a population (matching MOGA population size), which is then evaluated with the MOGA evaluator. By taking three classic multiobjective problems as examples, we demonstrate that the proposed method promises a faster convergence and a higher degree of diversity than that available with the standard MOGA and other three optimization methods that have been applied in the accelerator field, i.e., the multiobjective particle swarm optimization (MOPSO), the combination of MOPSO and MOGA, and the clustering enhanced MOGA. And then this method is applied to a time-consuming optimization problem, the dynamic aperture and Touschek lifetime optimization of the high energy photon source. It turns out that, within the same optimization time, a better set of solutions in the objective space can be obtained with the NBMOGA than using other methods. The Touschek lifetime can be improved by about 10% compared with using the standard MOGA, with approximately the same dynamic aperture area. Besides, a higher degree of diversity among solutions is observed with the NBMOGA than using other tested methods.

DOI: 10.1103/PhysRevAccelBeams.23.081601

I. INTRODUCTION

Multiobjective genetic algorithms (MOGAs) [1] have proven to be powerful in exploring optimal solutions of multiobjective problems in many research areas [2–9]. They are widely used in the accelerator field, especially for global optimization of particle accelerator lattices, including optimizations of linear optics and nonlinear dynamics [2–6]. For an optimization problem with V input variables, referred to as genes and denoted with x_v ($v = 1, 2, \ldots, V$), and M performance parameters to be optimized, referred to as objectives and denoted with y_m ($m = 1, 2, \ldots, M$), the goal of a MOGA is to optimize the values of objectives, so as to explore optimal combinations of variables in their parameter space.

Taking a typical MOGA, the nondominated sorting genetic algorithm II (NSGA-II) [10], as an example, its procedure can be described as follows. (i) An initial population consisting of N random seeds is generated, where N is the population size. One seed (sometimes referred to as an individual) $X_n = [x_{n,1}, x_{n,2}, \ldots, x_{n,V}]$ ($n = 1, 2, \ldots, N$) denotes a set of variable values. (ii) These seeds are evaluated via an evaluator (a numerical simulation code, or an experiment), to calculate the corresponding objective values denoted with $Y_n = [y_{n,1}, y_{n,2}, \ldots, y_{n,M}]$ ($n = 1, 2, \ldots, N$). (iii) Then the evaluated seeds are sorted with the nondominated sorting method (for more details of this method, we recommend Ref. [11]). The nondominated individuals are assigned a rank of 1, and the rest of the solutions are assigned ranks according to their domination levels. (iv) The seeds with better objective performance are weighted to increase the probability of being selected to generate offspring by

*Corresponding author.
chuzm@ihep.ac.cn
†Corresponding author.
jiaoyi@ihep.ac.cn

crossover and mutation. (v) The offspring individuals are then evaluated and combined with parent individuals, from which N individuals with better objective performance are selected to form a new generation. (vi) Repeat steps (ii)–(v) until the criterion of convergence is met or the maximum of iterations is reached.

It has been noted that the optimization performance of MOGAs depends highly on the distribution of the initial population. Experiences (e.g., [6,12]) indicate that, if the initial population is not generated with enough diversity, the algorithm may converge to local optima rather than global optima, especially when a MOGA is applied to a complicated problem with many variables and many local optima. To overcome this problem, the most direct way is to use a large size of initial population, so as to ensure enough diversity. This, however, implies a requirement of a long computing time and/or many computing resources. In an actual scenario, the computing resources are usually limited, and one always wants to obtain solutions in as short a time as possible. Thus, the problem becomes, with limited computing resources, how to achieve better optimization performance, e.g., faster convergence and/or more diversity, in a specific time.

In the past few years, many measures have been proposed to enhance the MOGA optimization performance. For instance, multiobjective particle swarm optimization (MOPSO) [13] was proven to be able to promise faster convergence than MOGA in ring lattice optimization [6]. And, a combination of MOGA and MOPSO has more potential of avoiding local optima than using the MOGA or MOPSO alone [12]. Moreover, accelerator scientists are now exploring different machine learning enhanced MOGAs [14–18]. The basic consideration is that the dataset $\{X_n, Y_n\}$ continuously produced and accumulated by the MOGA can be used as the training data of machine learning, so as to reveal some hidden properties of the data which, in turn, helps to speed up the convergence and/or increase the diversity among solutions.

One approach is to combine MOGA with data clustering (e.g., [14,15]), namely, using K-means clustering for each generation to find the “elite” variable range covered by the individuals that have high objective performance. The new competitive offspring individuals are then generated within the elite variable range and used to replace the original data in the population, which can result in faster convergence. Another approach is to use the data from MOGA to train a machine learning surrogate model (e.g., [16,17]), which can predict the objective values in fractions of a second, much faster than the actual evaluator. To take advantage of such a surrogate model, one can use it to replace the actual MOGA evaluator as in Ref. [16]. Such an approach can greatly reduce the optimization time. However, one problem is that there is an inevitable error between the prediction of the surrogate model and the result of the actual evaluator. It is empirically found that, for some problems with advancing steps of the objectives for each generation similar to or even smaller than the prediction error, the individuals with actually better objective performance may be underestimated to have worse objective values and accordingly assigned a lower probability of being selected to form the new generation. In the worst scenario, the prediction error may cause the algorithm to suggest a different optimizing direction from what it actually would be.

Actually, the problem can be avoided. As shown in Ref. [18], for each generation of a MOGA, a Gaussian process regression-based model was constructed and used to predict the objective values of a large pool of candidate individuals ($\gg N$). Then N individuals with high probability of producing better objective values were selected and evaluated with the actual evaluator. It has been demonstrated that this approach promises much faster convergence than the standard MOGA. Nevertheless, due to the fact that Gaussian process regression has a time complexity of $O(n^3)$ [19], this approach is especially suitable for problems when the dataset is not very large. In the case of using a large dataset, the training time for Gaussian process regression will be significantly increased. To avoid the training time being too long, one could use part of (instead of all) the obtained samples, e.g., by setting an upper limit for the number of the training data samples, to build the Gaussian process model.

In the presented study, we adopt a similar approach to that in Ref. [18], while replacing the Gaussian process regression-based model with a neural network-based model and using the data of the first few generations of MOGA for training. For simplicity, hereafter this approach is referred to as neural network-based MOGA (NBMOGA). Since the training time for neural network increases only linearly with the number of samples, it is thought the NBMOGA allows for greater flexibility and adaptability.

To demonstrate the effectiveness of NBMOGA, taking three classic optimization problems as test problems, the convergence rate and diversity available with the NBMOGA, the standard MOGA, and three other multiobjective optimization methods that have been proposed and used in the accelerator field, i.e., the MOPSO, the combination of MOPSO and MOGA, and the clustering enhanced MOGA (CEMOGA) [15], are quantitatively compared in Sec. II. The impact of population size on optimization performance is especially investigated. The results suggest that, with the NBMOGA, the available minimum population size can be greatly reduced, while ensuring a fast convergence rate and high degree of diversity. In addition, we trace the origin of each individual in a generation to reveal how NBMOGA improves the diversity. Then in Sec. III, the NBMOGA is applied to the Touschek lifetime and dynamic aperture optimization of the high energy photon source (HEPS). In this optimization, it is very time consuming for evaluating the objective values of individuals, and the population size cannot be

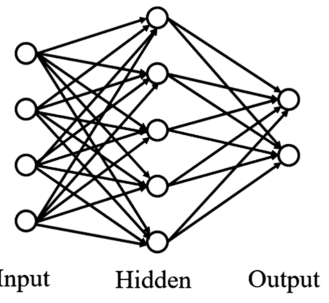

FIG. 1. A schematic of a typical fully connected neural network.

set to a large value due to the limited computing resources. The results indicate that the NBMOGA results in a better set of solutions having higher objective performance and a more continuous nondominated front in the objective space than using the other four methods. Within almost the same optimizing time as the standard MOGA, the NBMOGA allows a further improvement of the Touschek lifetime by about 10%, with almost the same dynamic aperture area.

II. METHOD

A. Neural network

Thanks to the astonishing development of computing devices, the neural network, one of the classic machine learning algorithms that has existed for more than 70 years, has gotten a second wind in the age of the 21st century. In recent years, the developments of supervised learning and reinforcement learning are remarkable in the areas of pattern recognition and automatic control, signaling that the machine learning field moves a step toward the expected general artificial intelligence. Neural networks play a central role in these developments, as almost all of those inspiring achievements are based on deep neural networks. Neural networks are becoming the most popular machine learning algorithms.

The neural network is a relevant method for the purpose of this paper, and we briefly describe it here. For further reading, we recommend Lippmann [20], Widrow and Lehr [21], and Girosi and Poggio [22]. A schematic of a typical fully connected neural network is shown in Fig. 1. It contains at least three layers: the input layer, the hidden layer, and the output layer. In either of the hidden and output layers, there is at least one node called a neuron, which represents a nonlinear algebraic function called an activation function. The input data are fed to the hidden layer from the input layer and then processed by the activation functions. The computation results are then weighted and summed as the inputs of the next layer, as described in Eq. (1):

$$O_{j,k} = f_A\left(\sum_{i=1}^{N_{k-1}} W_{i,k-1} O_{i,k-1} + b_{i,k-1}\right), \tag{1}$$

where f_A is the activation function; $O_{i,k}$, $W_{i,k}$, and $b_{i,k}$ are the output, weight, and bias of the ith neuron in the kth layer, respectively; and N_k is the number of neurons in the kth layer.

This process is repeated until the output layer is reached. Backpropagation learning [23] is usually used to adjust the weights and biases in each neuron to train the network to fully model the mapping between the inputs and outputs.

B. Neural network-based MOGA

With the NBMOGA, the processing for the first few generations follows a standard MOGA, i.e., NSGA-II in this study. As the data pool becomes large enough, e.g., the number of samples becomes larger than $100V$ (V is the number of variables), a neural network is trained with the data produced in the evolution of the MOGA. The optimized variables x_v ($v = 1, 2, \ldots, V$) and objectives y_m ($m = 1, 2, \ldots, M$) in the optimization, where M is the number of the optimized objectives, are treated as inputs and outputs of the network, respectively.

In the later optimization, the number of generated offspring in each generation is increased to K times that of a standard MOGA. Specifically, when the population size is N, $K \times N$ offspring will be generated by crossover and mutation. The objective values of the offspring are estimated with the trained neural network in fractions of a second. Based on the estimated results, those offspring are ranked with the nondominated sorting method. To alleviate the influence of the prediction error on the ranking of the offspring, $1.1 \times N$ top-ranked individuals are first retained, from which N solutions are randomly selected and used as the evolutionary candidates. With their objective performance evaluated on the actual evaluator, these N candidates are then combined with their parent individuals and ranked with the nondominated sorting method again. The N top-ranked solutions are selected from the combination to form a new generation. In addition, the prediction accuracy of the neural network is also validated in each generation, with the newly evaluated solutions as validation data.

To summarize the NBMOGA processing, let us denote the initial population as $\mathcal{P}_0$; the parents and children in the tth generation as $\mathcal{P}_t$ and $\mathcal{Q}_t$; the combination of $\mathcal{P}_t$ and $\mathcal{Q}_t$ as $\mathcal{R}_t$; the generation index when the neural network training starts as t_a; and the maximum number of generations as $t_{\max}$. The constructed neural network is called $\mathcal{NN}$ for short. The main loop of the NBMOGA can be described as follows:

```
initialize $t \leftarrow 0$, $\mathcal{P}_0$
while $t < t_{\max}$
  if $t < t_a$
    Perform standard NSGA-II to generate $\mathcal{P}_t + 1$.
  else if $t = t_a$
    Perform standard NSGA-II to generate $\mathcal{P}_t + 1$.
    Construct a neural network $\mathcal{NN}$.
    Train $\mathcal{NN}$ with the data produced from the first to
    the $t_a$th generations.
```

```
        Validate accuracy of $\mathcal{NN}$ with the new data in
        $\mathcal{P}_t + 1$ as testing data.
    else if $t > t_a$
        Generate $K \times N$ offspring from $\mathcal{P}_t$ by mutation and
        crossover.
        Estimate performance of the $K \times N$ offspring
        with $\mathcal{NN}$.
        Rank these offspring with the nondominated sorting
        method.
        Randomly select $N$ solutions denoted as $\mathcal{Q}_t$ from
        the $1.1 \times N$ top-ranked solutions.
        Evaluate solutions in $\mathcal{Q}_t$ on the actual evaluator.
        Combine $\mathcal{P}_t$ and $\mathcal{Q}_t$ as $\mathcal{R}_t$.
        Rank $\mathcal{R}_t$ according to the actual performance with
        the nondominated sorting method.
        Select $N$ top-ranked solutions in $\mathcal{R}_t$ as $\mathcal{P}_t + 1$.
        Validate accuracy of $\mathcal{NN}$ with the new data in $\mathcal{Q}_t$
        as testing data.
    end if
        $t = t + 1$.
end while
```

C. Optimizations of three test problems

To investigate the performance of NBMOGA, we use three classic multiobjective problems as test problems, the so-called ZDT1, ZDT2, and ZDT6 referenced in Ref. [24], with 10, 30, and 10 input variables, respectively. Details of their definitions are described in the Appendix A. All the test problems have known optimal solution sets, the so-called Pareto front, which helps to quantify the performance of optimization algorithms. In addition to the NBMOGA and the standard MOGA, three other evolutionary algorithms showing faster convergence than MOGA in accelerator optimizations, i.e., the MOPSO, the combination of MOGA and MOPSO, and the CEMOGA, are also tested for further comparison. For fair comparison, the optimizations with different algorithms are seeded with the same initial population.

For the NBMOGA, the parameter t_a, i.e., the index of the generation when training starts, is set to 10, and a simple neural network that has only one hidden layer with 32 hidden neurons is adopted to build the surrogate model. It is worth mentioning here that, although it is enough to use such a simple network to demonstrate the effectiveness of NBMOGA, even better optimization performance can be achieved if using a more complicated network. The cases with parameter K of 5 and 15 are both tested. For the CEMOGA, the ratio of the individual replacement is set to be the same as in Ref. [15], 20%. For the MOPSO, the same parameter settings as in Ref. [6] are adopted, with the velocity weight factor w in Eq. (2) being set to 0.4 and the acceleration coefficients of group best experience (*gbest*) and personal experience (*pbest*), i.e., c_1 and c_2 in Eq. (2), being set to 1. It is worth mentioning that, different from typical parameter settings of MOPSO, the two random coefficients, i.e., r_1 and r_2 in Eq. (2), are fixed to 1. To test the performance of the combination of MOGA and MOPSO, the solutions of the 20th generation of the MOPSO evolution is used as the initial population of a further optimization with the standard MOGA. In addition, considering that the population size is an important control parameter, all the test problems are optimized with different population sizes, from 50 to 1000, to study how population size affects the optimization performance.

$$v_t^{t+1} = wv_i^t + c_1 r_1 (p_i^t - x_i^t) + c_1 r_1 (g_i - x_i^t). \qquad (2)$$

The convergence rate and the diversity among solutions are treated as two performance parameters of optimization. To quantitatively measure the degree of convergence, the

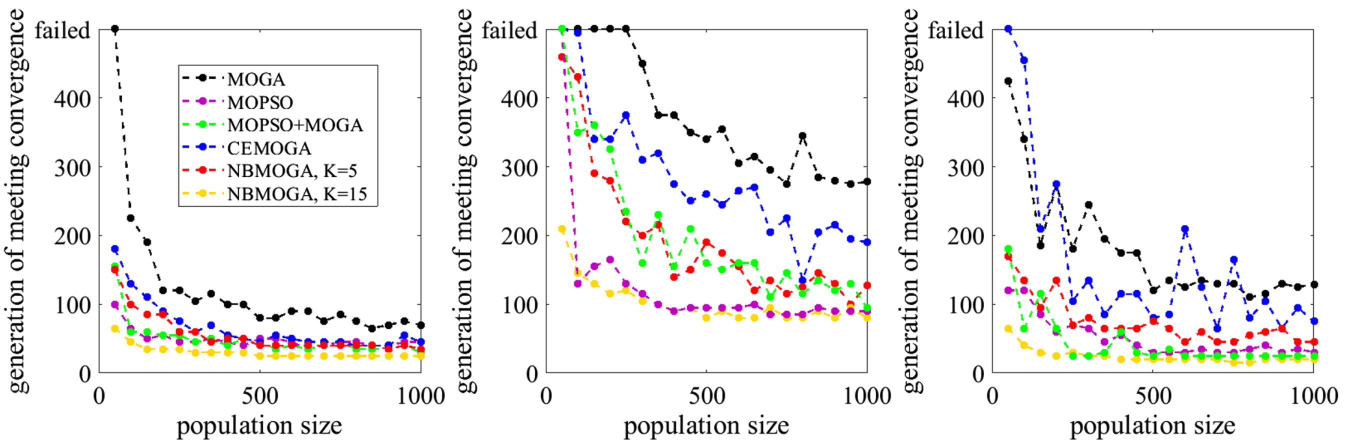

FIG. 2. Comparison of the convergence rate versus population size for the three test problems. The figures from left to right represent ZDT1, ZDT2, and ZDT6, respectively. The X axis indicates the population size, and the Y axis is the generation index when the population first reaches to the Pareto front. The "failed" on the Y axis indicates the failure of reaching the Pareto front after 500 generations of optimization. The black, purple, green, blue, red, and yellow points represent the results obtained with the standard MOGA, MOPSO, combination of MOPSO and MOGA, CEMOGA, NBMOGA ($K = 5$), and NBMOGA ($K = 15$), respectively.

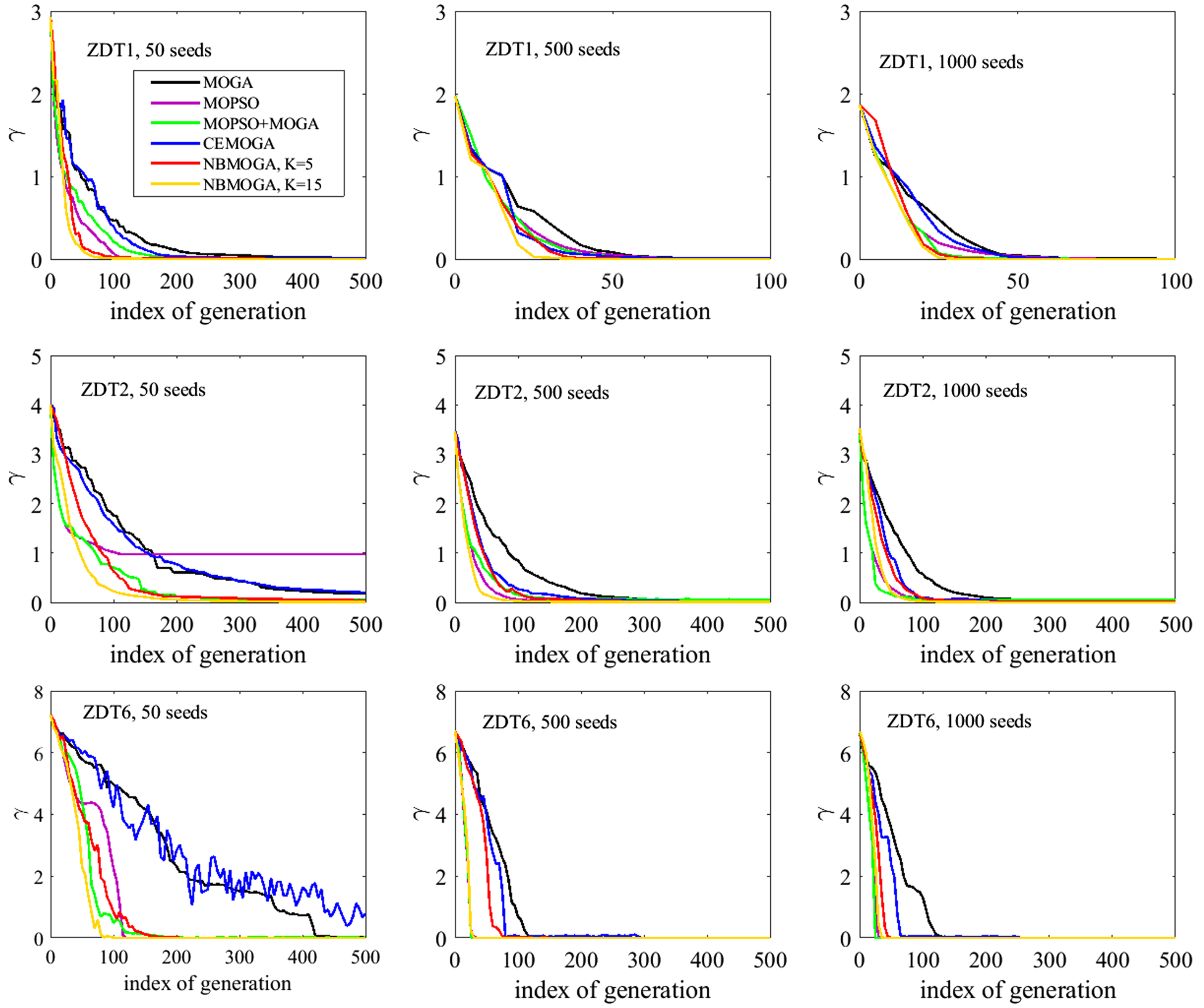

FIG. 3. The evolutions of γ over generations for a few cases. For the combination of MOPSO and MOGA, the data of the first 20 generations are obtained from the MOPSO optimizations—i.e., the purple and green curves are the same in the first 20 generations—and the data of the 21st to the last generations are obtained with the standard MOGA. In the cases of optimizing ZDT1 with population sizes of 500 and 1000, the convergence is fast, and only the first 100 generations are illustrated for a clear comparison.

convergence metric γ proposed in Ref. [10] (with its definition presented in Appendix B) is used, which measures the average distance of the obtained nondominated solutions and the Pareto front. The smaller the γ is, the closer the obtained solutions are to the Pareto front. For all test problems, the criterion marking successful convergence to the Pareto front is set to $\gamma < 0.05$, which can guarantee that almost all obtained solutions are located on the Pareto front. The index of the first generation to meet this criterion is used to characterize the convergence rate.

The required minimum numbers of generations to meet the Pareto front for different methods are shown in Fig. 2. In addition, to better compare the convergence rates of different methods, the evolutions of γ for a few cases are illustrated in Fig. 3.

For the presented optimization methods, the convergence rate roughly improves with the population size. Once the population size is large enough, e.g., $N \geq 10V$ for the standard MOGA, the minimum number of generations to reach the Pareto front stabilizes. In such a case, if the population size continues to increase, the gain in the convergence rate may be inconsequential compared with the extra required computing resources or evaluation time. For all three problems, with the same population size, the NBMOGA with $K = 5$ requires a smaller number of generations to reach the Pareto front than the standard MOGA and CEMOGA, while it needs more generations than the MOPSO and combination of MOPSO and MOGA in most cases. When K is increased to 15, the required minimum number of generations to reach the Pareto front for the NBMOGA becomes less than that of the MOPSO and combination of MOPSO and MOGA. Namely, it is feasible for the NBMOGA with a large K (e.g., 15) to use a smaller population size than the other four methods for a similar convergence rate. Even in the cases with population sizes as small as 50, the NBMOGA can speed up the evolution smoothly and effectively.

The results also indicate that in very few cases, e.g., optimizing ZDT2 with a population size of 50, the MOPSO fails to reach the Pareto front even after 500 generations of evolution. Nevertheless, the results in Fig. 3 indicate that the MOPSO does evolve fast in the early state of the optimization. The problem is that after a few tens of generations the convergence may slow down or even stop while the Pareto front is not reached yet. This phenomenon has been reported (e.g., in Ref, [25]) and referred to as the "premature convergence" problem. From Fig. 3, one can see that, by combining the MOPSO and MOGA, the γ can be further reduced in the case of the MOPSO optimization being trapped in premature convergence. However, for many other cases, the combination of MOPSO and MOGA does not show faster convergence than using MOPSO alone. It may be because the chosen generation to use the standard MOGA (i.e., 20th generation) is not the best for these cases.

However, note that a smaller number of generations of evolution does not definitely correspond to a shorter optimization time, which is related to both the number of generations and the running time for each generation. Taking ZDT2 as an example, we calculate the average running time for each generation for the presented methods, with the results shown in Fig. 4. For this problem, the NBMOGA requires a longer average time of up to 15 ($K = 5$) and 40 s ($K = 15$) for each generation, which is a few times larger than that of the other four methods, due to the training of the neural network surrogate model, the prediction of the objective values, and the sorting of a large number of candidates. Nevertheless, as will be shown in the next section, for a more complex and practical problem, more time will be taken for evaluating the objectives while the time of training and prediction of the surrogate model and sorting of the candidates will not grow as the complexity of the problem increases. The running times for each generation of the NBMOGA, the standard MOGA, the MOPSO, and the combination of MOPSO and MOGA are approximately equal, while the CEMOGA needs more running time than the other four methods due to the required evaluation of the additional solutions in each generation. For such a problem, the NBMOGA, requiring fewer generations with smaller population sizes and providing comparable or improved results, can potentially use computer resources more efficiently.

Another parameter describing the optimization performance is the diversity among the obtained nondominated solutions. We use the diversity metric Δ proposed in Ref. [10] to measure the degree of diversity (see the definition in Appendix B). The larger the Δ is, the lower the degree of diversity is.

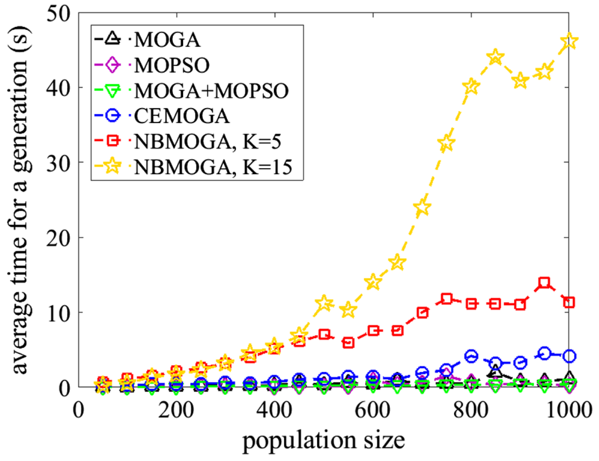

FIG. 4. The average running time for each generation of optimizing ZDT2 with different optimization methods and different population sizes. The average time is defined as the total optimizing time divided by the number of generations.

A stable nondominated front is required to calculate Δ. Based on observation, we empirically find that the nondominated front usually stabilizes after dozens of generations, and henceforth we start to calculate Δ. In addition, it is found that the randomness in the evolution of a MOGA sometimes heavily affects the diversity among solutions. To reduce the influence of randomness on Δ, the optimizations mentioned above are repeated 5 times. And, the average value of Δ, denoted as $\bar{\Delta}$, is used to measure the performance of optimization algorithms in terms of diversity.

Figure 5 presents the $\bar{\Delta}$ in the 100th and 300th generations of optimizing ZDT1 and the 300th and 500th generations of optimizing ZDT2 and ZDT6. Such two generations can represent the cases when the population is close to the Pareto front and at the Pareto front, respectively. With the NBMOGA, a higher degree of diversity than that of the other four methods is observed, especially for the cases with as a low population size as 50. For the CEMOGA, the resulting diversity is similar to, and sometimes lower than, that of the standard MOGA. This is probably because the replacement of original data with the individuals repopulated in a reduced variable range may decrease the diversity [15]. For the combination of the MOPSO and MOGA, the obtained diversity is slightly higher than that with MOPSO in most cases of optimizing ZDT2 but becomes lower than that of MOPSO when optimizing ZDT1 and ZDT6. It may be because the chosen generation to use the standard MOGA is not the best for the two problems. Figure 6 shows the obtained solutions of the cases with a population size of 50. The nondominated front obtained with the NBMOGA is more continuous in the objective space than that of other methods, especially at a

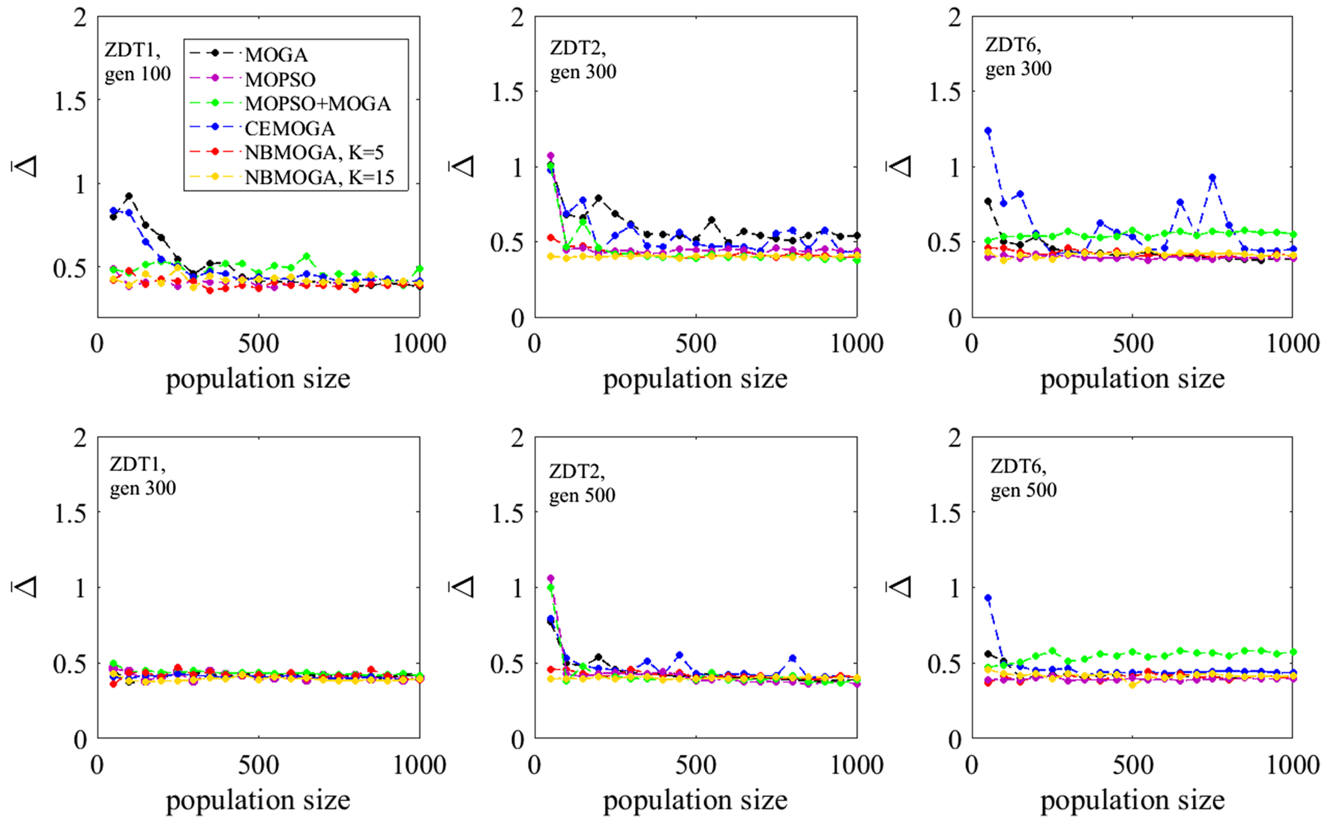

FIG. 5. Variations of the average diversity metric $\bar{\Delta}$ with the population size, where $\bar{\Delta}$ represents the average of Δ for five repeated optimizations.

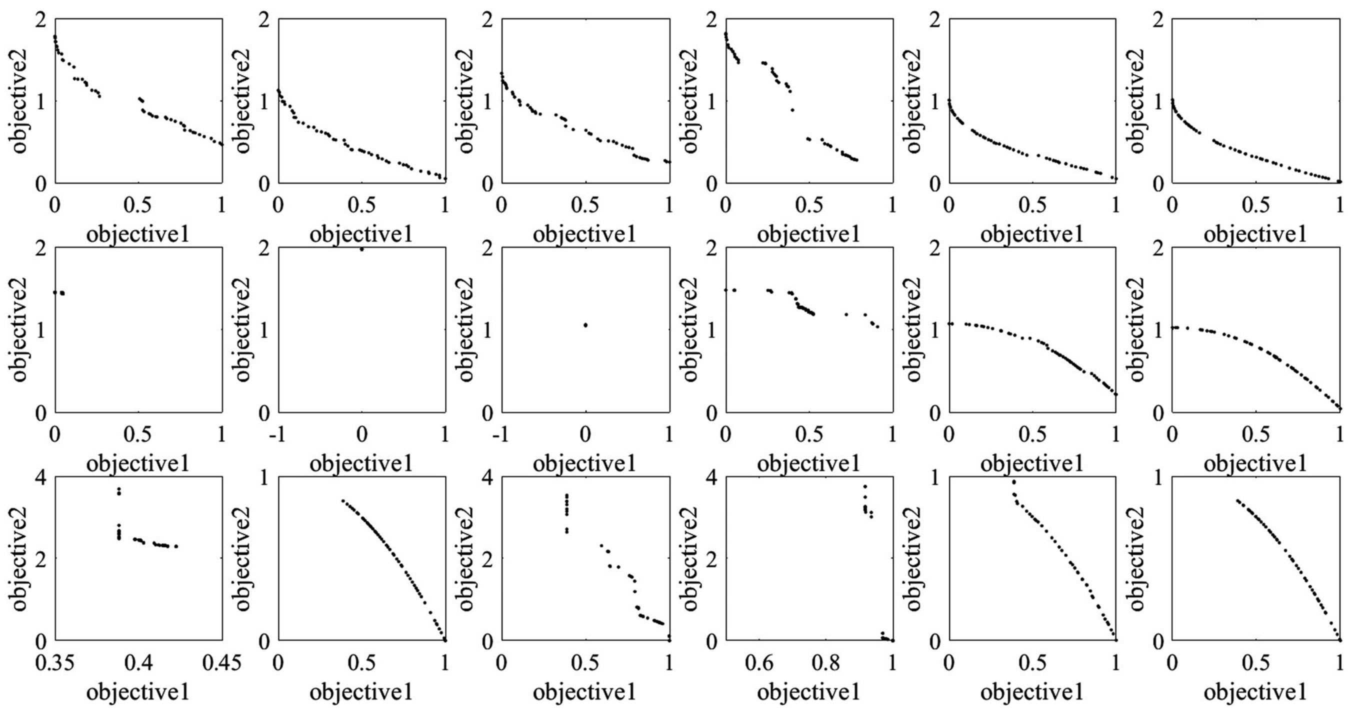

FIG. 6. The obtained solutions of the cases with a population size of 50. The rows from top to bottom represent the results of ZDT1 (100th generation), ZDT2 (300th generation), and ZDT6 (300th generation), respectively. The columns from left to right represent the results obtained with the standard MOGA, MOPSO, combination of MOPSO and MOGA, CEMOGA, NBMOGA ($K = 5$), and NBMOGA ($K = 15$), respectively.

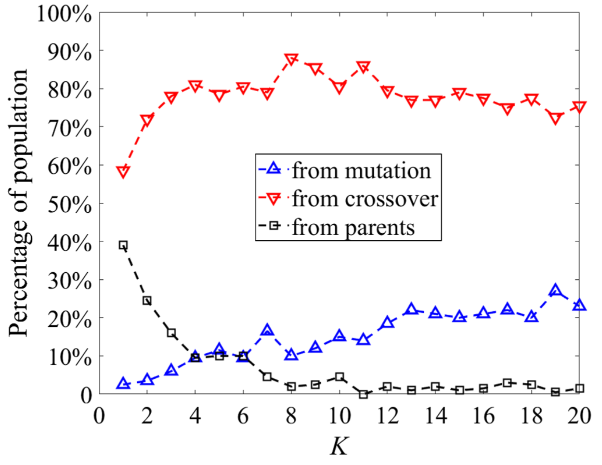

FIG. 7. The origins of individuals in the 11th generation of optimizing ZDT2 with the NBMOGA. The K parameter in the X axis represents different numbers of offspring generated with the NBMOGA. The blue, red, and black markers represent individuals originated from mutation, crossover, and copies of parents based on the solutions of the 10th generation, respectively.

low population size, corresponding to a higher degree of diversity.

In a further step, we try to investigate why and how NBMOGA improves the diversity. As we know, for the standard MOGA, the diversity is mainly fed by mutations in each generation of evolution. However, due to the random nature of mutation operation, the mutation individuals are more likely to have a lower objective performance than those generated from crossover or even copies of parents. A high probability of mutation could decrease the convergence rate [26]. Therefore, the nominal value of the mutation probability for a standard MOGA is usually set to a small number, e.g., 0.1. In contrast, thanks to the quick prediction of objective values with the surrogate model, it is feasible for the NBMOGA to generate a larger group of offspring (with the same probability of mutation), from which one can select more competitive candidates generated from mutation. To verify this understanding, taking ZDT2 as an example, we count the ratios of the individuals generated from crossover, mutation, and copies of parents in the 11th generation (the training of the neural network starts at the 10th generation), with different K parameters. The case with $K = 1$ can be regarded as the standard MOGA. One can clearly see that in Fig. 7, as the K parameter increases, the ratio of the mutants relative to the population size increases, from just a few percent to about 20%, providing a higher degree of diversity among solutions.

III. OPTIMIZATION OF NONLINEAR BEAM DYNAMICS OF THE HEPS

The HEPS is a 6-GeV, 1.3-km, ultralow-emittance storage ring light source being built in Beijing, China [27]. The lattice of the HEPS storage ring has been iteratively evolved and optimized for more than 10 years [28]. Currently, the storage ring lattice consists of 48 hybrid seven bend achromats (7BAs), which are grouped in 24 periods [29]. To explore the ultimate potential of the lattice, a global parameter scan of the lattice with stochastic algorithms has been extensively used to simultaneously optimize the linear and nonlinear beam dynamics [30], aiming to find an optimal balance between the available maximum brightness (or minimum emittance) and the largest possible dynamic aperture (DA).

There is another important performance parameter of the nonlinear beam dynamics of a ring light source, the Touschek lifetime (TL), which dominates the electron beam lifetime in a high brightness and low emittance light source [31]. Note that the combination of multipole strengths (i.e., strengths of sextupoles and octupoles) resulting in a large DA does not always correspond to a high TL. Thus, after the lattice linear parameters were fixed based on the brightness and DA optimization, optimization of the TL is necessary and has been performed for HEPS.

In this optimization, the goal is to simultaneously obtain a large DA with a long TL. Both of them are computed with the particle tracking simulation code Accelerator Toolbox (AT) [32]. Among 16 families of multipoles, except two families of sextupoles used for chromaticity correction, the strengths of the other 14 families of multipoles are used as optimizing variables and varied within specific ranges determined by practical limits (see the ranges in Table I). Two optimization objectives are constructed based on the values of the DA area and TL, which are denoted as f_1 and f_2, respectively:

$$f_1 = -\frac{\mathrm{DA}_x \times \mathrm{DA}_y}{\sqrt{\beta_x[m]\beta_y[m]}} \frac{\mathrm{MA}}{0.03} \prod w_i, \tag{3}$$

$$f_2 = -\tau \prod w_i, \tag{4}$$

where DA_x and DA_y are the horizontal and vertical DA size, respectively; β_x and β_y are the values of the horizontal and vertical beta functions, respectively, at the location where a particle is launched for DA tracking; MA is the

TABLE I. Variable ranges of the DA and TL optimization.

Variables	Scanning range
Strengths of sextupoles (KSD1, KSD5)	$[-126, 0]$ m^{-2}
Strengths of sextupoles (KSD2, KSD3, KSD4, KSD6, KSD7)	$[-192, 0]$ m^{-2}
Strengths of sextupoles (KSF1, KSF2, KSF3)	$[0, 192]$ m^{-2}
Strengths of octupoles (KOCT1, KOCT2, KOCT3, KOCT4)	$[-5592, 5592]$ m^{-3}

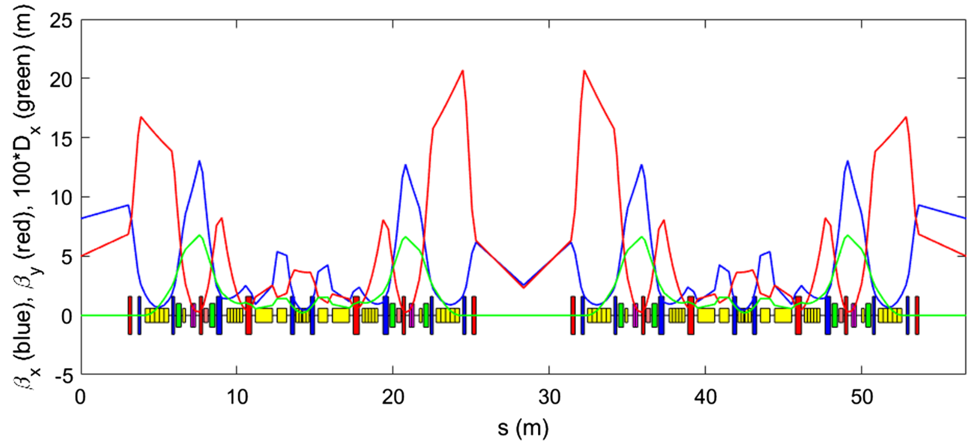

FIG. 8. Layout and optical functions of a period with two 7BAs in the lattice of the HEPS.

effective momentum acceptance [33] at the middle of the long straight section that is evaluated taking into account the limitation of dangerous low order resonances [34]; the weight w_i measures the degree of the solutions satisfying the ith constraint; and τ is the Touschek lifetime, which is determined by the local momentum acceptance, namely, the momentum acceptance of different locations along the ring. The constraints include limitations on the fractional part of tune, the maximum or minimum value of the beta function, the momentum compaction factor, the energy loss per turn, and the longitudinal damping partition number. Note that the objectives are assigned negative values, because the optimization is implemented for minimization.

This problem is optimized with the standard MOGA, MOPSO, combination of MOPSO and MOGA, CEMOGA, and NBMOGA, by adopting the same initial population that is generated by introducing random fluctuations on one initial solution, i.e., the lattice obtained from the brightness and DA optimization. The layout and optical functions of one period of the lattice corresponding to the initial solution are shown in Fig. 8. For the NBMOGA, after the tenth generation, a neural network is constructed to model the most time-consuming objective, the TL. A Python deep learning library, Keras [35], is used to train the network. After tests of different neural network structures, a neural network showing the highest prediction accuracy is adopted. It consists of three hidden layers with 64, 32, and 64 neurons in each hidden layer. As shown in Fig. 9, the coefficient of determination, R^2, is 0.9893 for this neural network, representing high prediction accuracy. The time taken for predicting the objectives of one individual with the trained model is about 0.04 s, which is 5 orders of magnitude shorter than that of particle tracking. For the other four methods, the control parameter settings are the same as that used in Sec. II.

One significant feature of this optimization problem is that it is very time consuming. To evaluate the linear parameters and the DA of a lattice, it will take less than one second and a few minutes, respectively. In contrast, it will take a few hours (3 h for the HEPS) to evaluate the TL for one set of multipole strengths, since it is necessary to calculate the momentum acceptances of many positions rather than that of only one specific location. For this problem, the available computing device is a parallel cluster equipped with 512 Intel® Xeon® 2.3 GHz CPU cores that allows us to compute the f_1 and f_2 for at most 512 sets of multipole strengths at the same time. As a compromise between better optimization performance and faster optimization, a population size of 500 is chosen for this optimization. In this way, the running time for evaluating each generation is approximately the same as the time taken for evaluating a single individual. In each generation, the time for training the neural network and for predicting the objective values of offspring is about 10 s, which is much less than the running time for each generation and can be

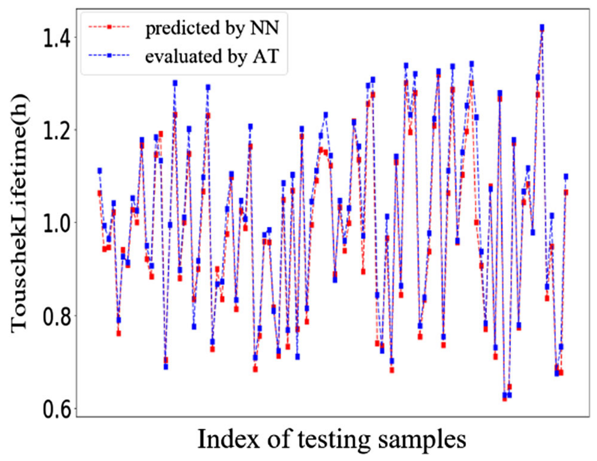

FIG. 9. Comparison of the Touschek lifetimes of 100 testing samples predicted by neural network (red) and particle tracking (blue). The coefficient of determination, R^2, is 0.9893.

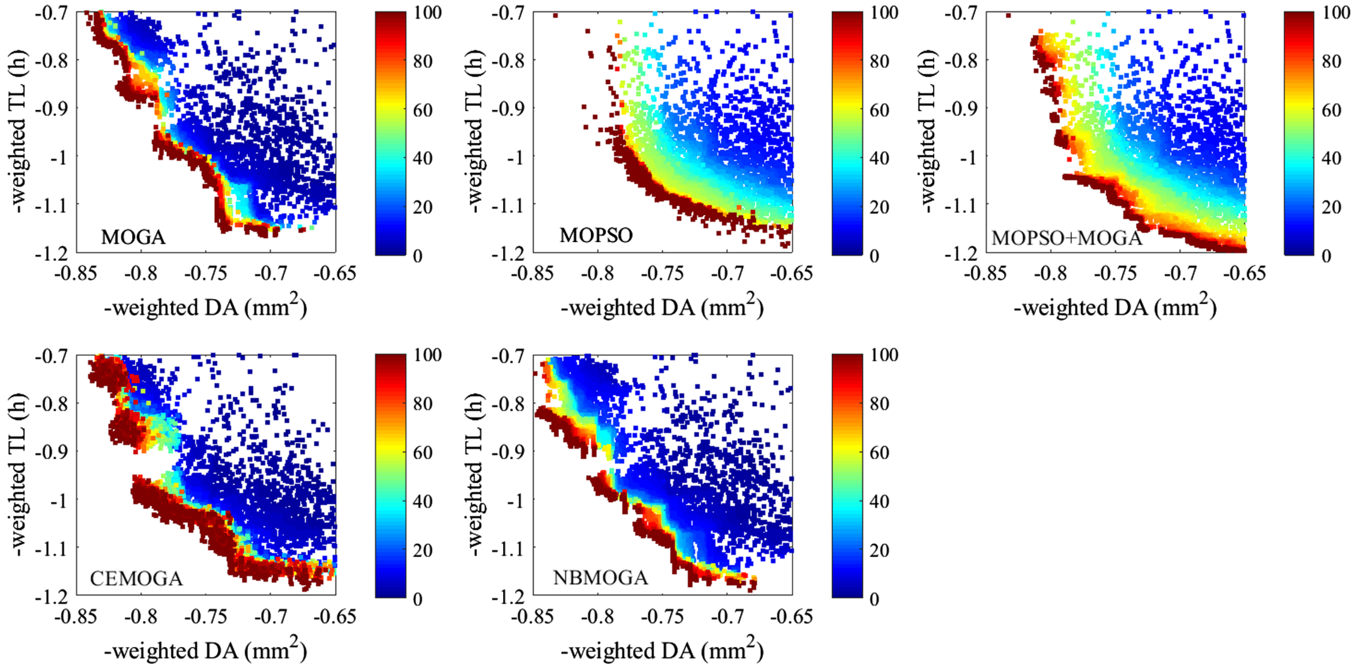

FIG. 10. Evolutions of population for optimizing DA and TL. The color from blue to red represents the index of generation from 1 to 100.

ignored. Therefore, the running time required for a generation of evolution with the standard MOGA, MOPSO, combination of MOPSO and MOGA, and NBMOGA is considered to be the same. However, for the CEMOGA, it is necessary to evaluate some extra "elite solutions" when the MOGA offspring are evaluated. In each generation, the CEMOGA program will successively call the parallel program twice to evaluate the MOGA offspring and the elite solutions, respectively, which doubles the evaluation time for this problem.

The evolutions of population over 100 generations (~600 h for the CEMOGA and ~300 h for the other four methods) are illustrated in Fig. 10, and the nondominated solutions obtained in the 100th generation are shown in Fig. 11. To better compare the solutions of different methods, here we combine the final solutions of all methods and sort them with the nondominated sorting method. Then a "best front," representing the nondominated ones among the combined solutions, is defined and shown in Fig. 11. The comparison of the nondominated front shows that the NBMOGA results in a better nondominated front with higher objective performance and more continuous distribution in the objective space than using other four methods. For the standard MOGA, even after completing 50 additional generations, the nondominated front found by the standard MOGA is still inferior to that of the NBMOGA. In addition, the distribution of the MOGA solutions becomes more discrete in the objective space, implying a lower degree of the diversity. For the MOPSO, a fast evolution is observed during the 30th–60th generations in Fig. 10. The evolution, however, slows down in the subsequent generations, and many solutions with larger DA are phased out. By combining MOPSO and MOGA, a wider and more continuous nondominated front can be observed than using MOPSO alone. For the CEMOGA, the obtained nondominated front is close to the best front, but the nondominated front is still discrete, especially for the solutions having large DA.

For this problem, the Pareto front is actually unknown, so the convergence criterion defined in the last section is not applicable to this case. However, to quantitatively compare the convergence rates between the standard MOGA and the NBMOGA, we define a γ^* similar to the convergence metric γ (see the definition in Appendix B). Different from γ, γ^* measures the average distance between the obtained nondominated solutions and the best front shown in Fig. 11. As shown in Fig. 12, with the NBMOGA, the γ^* evolves faster over generations than that with the standard MOGA, MOPSO, and combination of MOPSO and MOGA. For the CEMOGA, the dropping speed of γ^* is similar to (and a bit slower at the final stage than) that of the NBMOGA. However, note that the running time of CEMOGA is twice longer than that of the other four methods for this problem. The convergence rate of NBMOGA in terms of time is actually much faster than that of CEMOGA.

From Fig. 11, if keeping the DA area approximately the same as the initial solution, the TL can be further improved by about 10% with the NBMOGA compared with the standard MOGA within the same optimization time (see

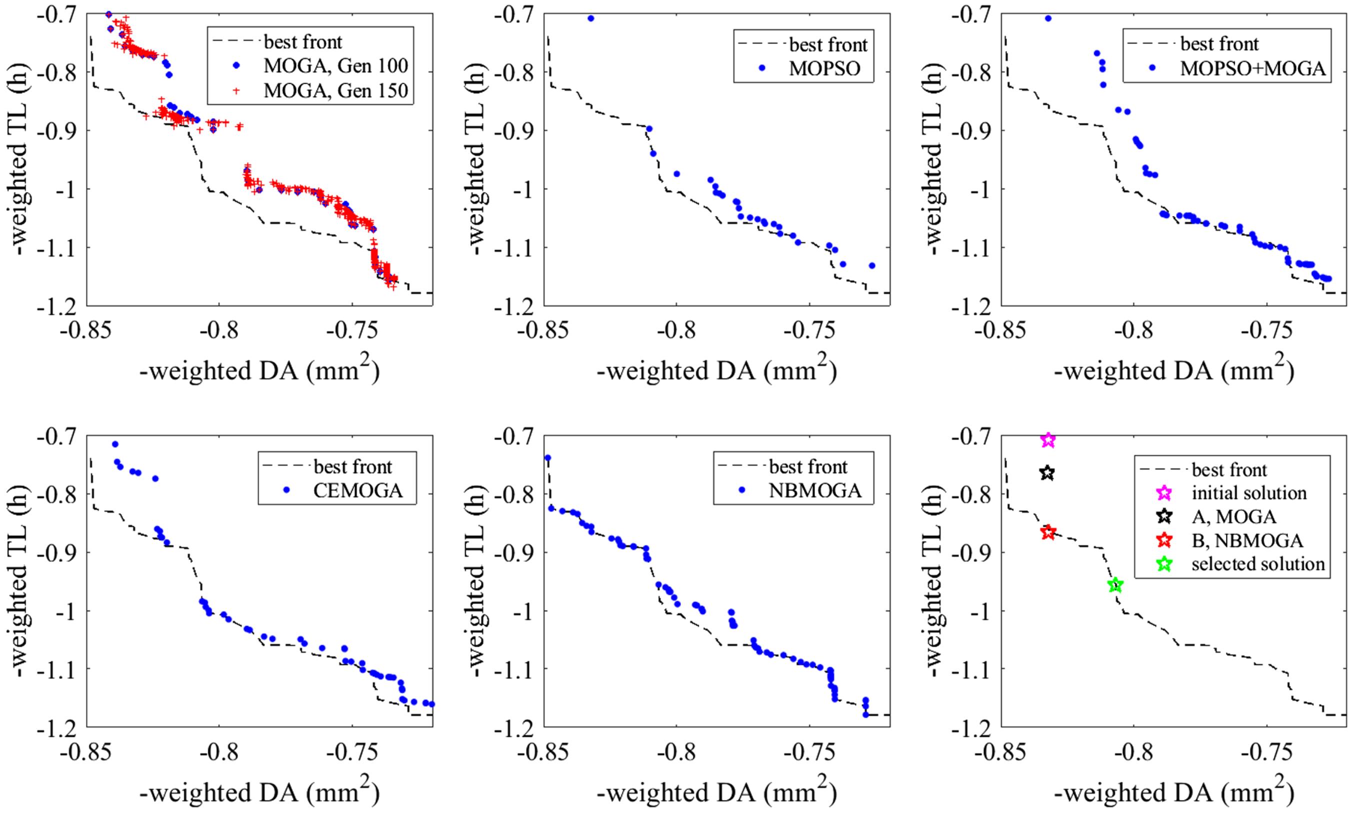

FIG. 11. Comparison of the nondominated solutions in the 100th generation obtained with different optimization methods. The black dashed line represents the best front. The pink star is the initial solution. The black and red stars, A and B, are two nondominated solutions of standard MOGA and NBMOGA, respectively, which have almost the same DA area as the initial solution. The green star is the selected solution for the multipole strengths of the HEPS.

solutions marked as A and B in Fig. 11). The green star selected for HEPS in Fig. 11 has a longer TL with large enough DA. Its variable values are listed in Table II. For this solution, the DA at the center of low-beta section center and the local momentum acceptance along one period of the lattice are plotted in Fig. 13 and compared with that of the initial solution. It can be seen that the selected solution just sacrifices a little DA in the horizontal plane, while it gains an obvious increase in local momentum acceptance that is beneficial to the beam lifetime.

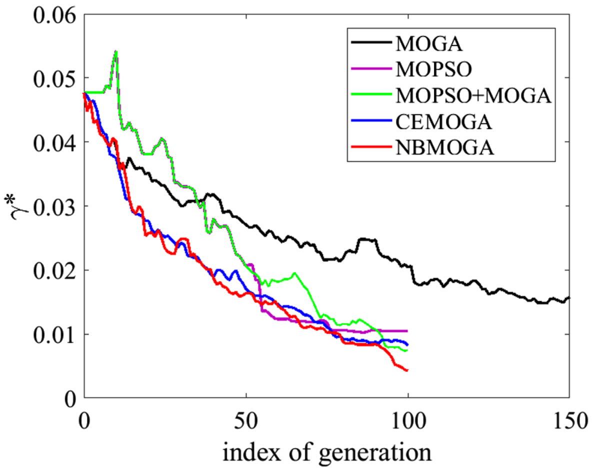

FIG. 12. Evolution of γ^* over generations (1–150 generations for the standard MOGA and 1–100 generations for the other four methods) for the DA and TL optimization of the HEPS. γ^* measures the average distance between the obtained nondominated solutions and the best front.

TABLE II. Comparison of the initial solution and the selected solution.

	KSD1 (m^{-2})	KSD2 (m^{-2})	KSD3 (m^{-2})	KSD4 (m^{-2})	KSD5 (m^{-2})
Initial	−116.52	−146.15	−124.53	−54.25	−111.21
Selected	−118.88	−148.44	−127.93	−58.10	−108.10
	KSD6 (m^{-2})	KSD7 (m^{-2})	KSF1 (m^{-2})	KSF2 (m^{-2})	KSF3 (m^{-2})
Initial	−71.73	−191.97	171.32	182.77	61.61
Selected	−69.04	−189.79	167.62	183.89	62.14
	KOCT1 (m^{-3})	KOCT2 (m^{-3})	KOCT3 (m^{-3})	KOCT4 (m^{-3})	
Initial	−5592	−4981	−3876	−5529	
Selected	−5498	−5166	−4087	−5563	

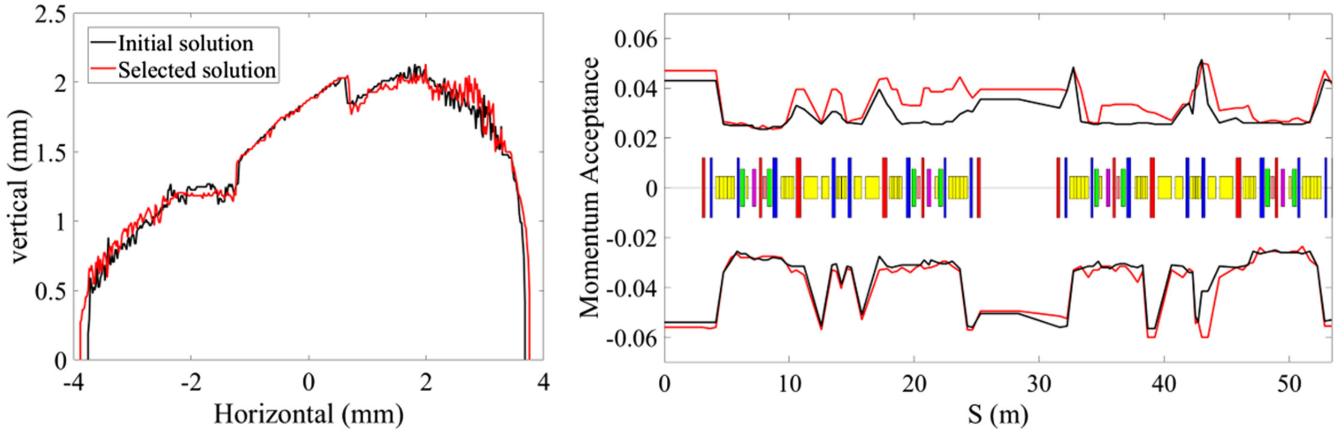

FIG. 13. The numerical tracking results of the DA (left) and local momentum acceptance (right) for the initial solution (black curve) and the selected solution (red curve).

IV. CONCLUSION

An enhanced MOGA algorithm, called neural network-based MOGA (NBMOGA), has been proposed. By taking three classic test problems as examples, it has been demonstrated that, if using the same population size, the NBMOGA is able to reach the Pareto front with not only a smaller number of generations but also a higher degree of diversity, compared with the other four optimization methods that have been used in an accelerator field, i.e., the standard MOGA, the MOPSO, combination of MOPSO and MOGA, and the CEMOGA. Moreover, with fewer generations and smaller population sizes, the NBMOGA can provide comparable or improved results. Additionally, it needs an extra time of a few seconds on average for each generation to train the surrogate model and to predict the objective values with the trained model. Thus, the NBMOGA method is especially applicable to time-consuming optimizing problems where the running time for each generation is on the minute or even hour scale. For such problems, the proposed method can provide more efficient use of computer resources.

With implementation of the NBMOGA in the dynamic aperture and Touschek lifetime optimization of the high energy photon source, within the same optimization time, a better set of solutions with a larger dynamic aperture area and longer Touschek lifetime is obtained than using the other four methods. A further improvement of the Touschek lifetime by 10% with almost the same dynamic aperture area is obtained compared with the standard MOGA. In addition, a higher degree of diversity among solutions is also observed.

ACKNOWLEDGMENTS

This work is supported by National Natural Science Foundation of China (No. 11922512), Youth Innovation Promotion Association of Chinese Academy of Sciences (No. Y201904), and Bureau of Frontier Sciences and Education, Chinese Academy of Sciences (No. QYZDJ-SSW-SLH001).

APPENDIX A: ZITZLER-DEB-THIELE PROBLEMS

In order to test performance of multiobjective optimization algorithms, Zitzler, Deb, and Thiele suggested six test problems [24], the so-called ZDT1–ZDT6. In this paper, ZDT1, ZDT2, and ZDT6 are chosen as test problems

TABLE III. Details of three test problems. All the objective functions are to be minimized.

Problem	Variable bounds	Objective functions	Optimal solutions	Number of variables
ZDT1	[0,1]	$f_1(X) = x_1$, $f_2(X) = g(X)[1 - \sqrt{x_1/g(X)}]$, $g(X) = 1 + 9(\sum_{i=2}^{n} x_i)/(n-1)$	$x_1 \in [0,1]$, $x_i = 0$, $i = 2, ..., 10$	10
ZDT2	[0,1]	$f_1(X) = x_1$, $f_2(X) = g(X)[1 - (x_1/g(X))^2]$, $g(X) = 1 + 9(\sum_{i=2}^{n} x_i)/(n-1)$	$x_1 \in [0,1]$, $x_i = 0$, $i = 2, ..., 30$	30
ZDT6	[0,1]	$f_1(X) = 1 - \exp(-4x_1)\sin^6(4\pi x_1)$, $f_2(X) = g(X)[1 - (f_1(X)/g(X))^2]$, $g(X) = 1 + 9[(\sum_{i=2}^{n} x_i)/(n-1)]^{0.25}$	$x_1 \in [0,1]$, $x_i = 0$, $i = 2, ..., 10$	10

to test performance of the NBMOGA. All of these problems have only two objectives. Details of them are described as in Table III.

APPENDIX B: THE CONVERGENCE METRIC γ AND THE DIVERSITY METRIC Δ

To measure the performance of different optimization methods, Deb proposed two performance metrics, γ and Δ, in Ref. [10]. The γ is defined as Eq. (B1), where $X_{O,i}$ denotes an obtained solution, $X_{P,i}$ denotes the Pareto-optimal solution with the smallest Euclidean distance to $X_{O,i}$, and N is the number of obtained solutions:

$$\gamma = \frac{\sum_{i=1}^{N} \sqrt{(X_{O,i} - X_{P,i})^2}}{N}. \tag{B1}$$

The Δ is formulated in Eq. (B2), where d_i is the Euclidean distance between consecutive solutions in the obtained nondominated set, $\bar{d}$ is the average distance of d_i, and d_f and d_l are the Euclidean distances between the boundary of the Pareto optimal set and the boundary of the N obtained nondominated solutions:

$$\Delta = \frac{d_f + d_l + \sum_{i=1}^{N-1} |d_i - \bar{d}|}{d_f + d_l + (N-1)\bar{d}}. \tag{B2}$$

Similar to Eq. (4), another metric, γ^*, is defined for the problems with an unknown Pareto front. As defined in Eq. (B3), $X_{B,i}$ denotes the solution with the smallest Euclidean distance to $X_{O,i}$ in the obtained best nondominated front:

$$\gamma^* = \frac{\sum_{i=1}^{N} \sqrt{(X_{O,i} - X_{B,i})^2}}{N}. \tag{B3}$$

[1] T. Murata and H. Ishibuchi, MOGA: Multiobjective genetic algorithms, in *Proceedings of IEEE International Conference on Evolutionary Computation* (IEEE, New York, 1995), Vol. 1, pp. 289–294.

[2] I. Bazarov and C. Sinclair, Multivariate optimization of a high brightness dc gun photoinjector, Phys. Rev. Accel. Beams **8**, 034202 (2005).

[3] L. Emery, Global optimization of damping ring designs using a multiobjective evolutionary algorithm, in *Proceedings of the 21st Particle Accelerator Conference, Knoxville, TN, 2005* (IEEE, Piscataway, NJ, 2005).

[4] L. Yang, D. Robin, F. Sannibale, C. Steier, and W. Wan, Global optimization of an accelerator lattice using multiobjective genetic algorithms, Nucl. Instrum. Methods Phys. Res., Sect. A **609**, 50 (2009).

[5] A. Hofler, B. Terzić, M. Kramer, A. Zvezdin, V. Morozov, Y. Roblin, F. Lin, and C. Jarvis, Innovative applications of genetic algorithms to problems in accelerator physics, Phys. Rev. Accel. Beams **16**, 010101 (2013).

[6] X. Huang and J. Safranek, Nonlinear dynamics optimization with particle swarm and genetic algorithms for SPEAR3 emittance upgrade, Nucl. Instrum. Methods Phys. Res., Sect. A **757**, 48 (2014).

[7] Y. F. Jin, Z. Y. Yin, S. L. Shen, and P. Y. Hicher, Investigation into MOGA for identifying parameters of a critical-state-based sand model and parameters correlation by factor analysis, Acta Geotechnica **11**, 1131 (2016).

[8] J. Wen, H. Yang, G. Jian, X. Tong, K. Li, and S. Wang, Energy and cost optimization of shell and tube heat exchanger with helical baffles using Kriging metamodel based on MOGA, Int. J. Heat Mass Transfer **98**, 29 (2016).

[9] L. Pu, D. Qi, L. Xu, and Y. Li, Optimization on the performance of ground heat exchangers for GSHP using Kriging model based on MOGA, Appl. Therm. Eng. **118**, 480 (2017).

[10] K. Deb, A. Pratap, S. Agrawal, and T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA-II, IEEE Trans. Evol. Comput. **6**, 182 (2002).

[11] N. Srinivas and K. Deb, Muiltiobjective optimization using nondominated sorting in genetic algorithms, Evolutionary Computation **2**, 221 (1994).

[12] Y. Jiao and G. Xu, Optimizing the lattice design of a diffraction-limited storage ring with a rational combination of particle swarm and genetic algorithms, Chin. Phys. C **41**, 027001 (2017).

[13] C. A. C. Coello and M. S. Lechuga, MOPSO: A proposal for multiple objective particle swarm optimization, in *Proceedings of the 2002 Congress on Evolutionary Computation* (IEEE, New York, 2002), Vol. 2, pp. 1051–1056.

[14] Y. Li, W. Cheng, L. H. Yu, and R. Rainer, Genetic algorithm enhanced by machine learning in dynamic aperture optimization, Phys. Rev. Accel. Beams **21**, 054601 (2018).

[15] J. Wan, P. Chu, Y. Jiao, and Y. Li, Improvement of machine learning enhanced genetic algorithm for nonlinear beam dynamics optimization, Nucl. Instrum. Methods Phys. Res., Sect. A **946**, 162683 (2019).

[16] A. Edelen, N. Neveu, M. Frey, Y. Huber, C. Mayes, and A. Adelmann, Machine learning for orders of magnitude speedup in multi-objective optimization of particle accelerator systems, arXiv:1903.07759.

[17] F. Wang, M. Song, A. Edelen, and X. Huang, Machine learning for design optimization of storage ring nonlinear dynamics, arXiv:1910.14220.

[18] X. Huang, Multi-objective multi-generation Gaussian process optimizer for design optimization, arXiv:1907.00250.

[19] C. K. I. Williams and C. E. Rasmussen, Gaussian processes for regression, Adv. Neural Inf. Process. Syst. **8**, 514 (1996), http://papers.nips.cc/paper/1048-gaussian-processes-for-regression.pdf.

[20] R. P. Lippmann, An introduction to computing with neural nets, IEEE Assp Mag. **4**, 4 (1987).

[21] B. Widrow and M. A. Lehr, 30 years of adaptive neural networks: Perceptron, madaline, and backpropagation, Proc. IEEE **78**, 1415 (1990).

[22] F. Girosi and T. Poggio, Networks and the best approximation property, Biol. Cybern. **63**, 169 (1990).

[23] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, Learning representations by back-propagating errors, Nature (London) **323**, 533 (1986).

[24] E. Zitzler, K. Deb, and L. Thiele, Comparison of multiobjective evolutionary algorithms: Empirical results, Evolutionary computation **8**, 173 (2000).

[25] Y. Zhao, W. Zu, and H. Zeng, A modified particle swarm optimization via particle visual modeling analysis, Comput. Math. Appl. **57**, 11 (2009).

[26] K. Q. Zhu and Z. Liu, Population diversity in permutation-based genetic algorithm, in *Proceedings of the European Conference on Machine Learning* (Springer, Berlin, 2004), pp. 537–547.

[27] Y. Jiao, G. Xu, X. H. Cui *et al.*, The HEPS project, J. Synchrotron Radiat. **25**, 1611 (2018).

[28] Y. Jiao, G. Xu, Y. M. Peng, S. Y. Chen, J. Y. Li, Q. Qin, J. Q. Wang, and C. H. Yu, Evolution of the lattice design for the High Energy Photon Source, in *Proceedings of the 9th International Particle Accelerator Conference (IPAC'18), Vancouver, Canada, 2018* (JACoW Publishing, Geneva, Switzerland, 2018), pp. 1363–1366.

[29] Y. Jiao, X. Cui, Z. Duan *et al.*, Beam dynamics study in the HEPS storage ring, in *Proceedings of the 10th International Particle Accelerator Conference (IPAC'19), Melbourne, Australia, 2019* (JACoW Publishing, Geneva, Switzerland, 2019), pp. 1203–1207.

[30] Y. Jiao and G. Xu, DA optimization experiences in the HEPS lattice design, *J. Phys. Conf. Ser.* 1067, 032003 (2018).

[31] A. Streun, Momentum acceptance and Touschek lifetime, Synchrotron Light Source **18** (1997), http://ados.web.psi.ch/slsnotes/sls1897a.pdf.

[32] A. Terebilo, Accelerator modeling with MATLAB accelerator toolbox, in *Proceedings of the 19th Particle Accelerator Conference, Chicago, IL, 2001* (IEEE, Piscataway, NJ, 2001).

[33] Y. Jiao, Z. Duan, and G. Xu, Characterizing the nonlinear performance of a DLSR with the effective acceptance of the bare lattice, in *Proceedings of the 8th International Particle Accelerator Conference (IPAC'17), Copenhagen, Denmark, 2017* (JACoW Publishing, Geneva, Switzerland, 2017), pp. 2706–2708.

[34] Y. Jiao and Z. Duan, Statistical analysis of the limitation of half integer resonances on the available momentum acceptance of a diffraction-limited storage ring, Nucl. Instrum. Methods Phys. Res., Sect. A **841**, 97 (2017).

[35] http://keras.io.