Multi-objective multi-generation Gaussian process optimizer for design optimization

Xiaobiao Huang

***Abstract*—We present a multi-objective optimization algorithm that uses Gaussian process (GP) regression-based models to generate or select trial solutions in a multi-generation iterative procedure. In each generation, a surrogate model is constructed for each objective function with the sample data. The models are used to evaluate solutions and to select the ones with a high potential before they are evaluated on the actual system. Since the trial solutions selected by the GP models tend to have better performance than other methods that only rely on random operations, the new algorithm has much better efficiency in exploring the parameter space. Simulations with multiple test cases show that the new algorithm has a substantially higher convergence speed that the NSGA-II and PSO algorithms.**

***Index Terms*—Gaussian process, optimization, multi-objective**

I. Introduction

THE design of a complex system often requires the search of the ideal solution among a multi-variable parameter space. The ideal solution may involve a trade-off of competing performance requirements. In recent years, multi-objective stochastic optimization has been widely adopted to discover the set of solutions with the best performances, i.e., the Pareto front. These include multi-objective genetic algorithms (MOGA) [1], [2] and multi-objective particle swarm optimization (MOPSO) [3], [4].

In an optimization many trial solutions will be evaluated. In a design study, typically an evaluation involves the numeric simulation of the physics processes that affect the system performance. Such a simulation could be computationally expensive, especially as the current trend is to build in as many details into the physics model as possible. Therefore, the efficiency of the optimization algorithm is very important.

In MOGA and MOPSO algorithms, an iterative process is executed to update a population of solutions. During each iteration, which may be referred to as a generation, new trial solutions are generated and evaluated. Both methods employ stochastic operations to produce new solutions with existing good solutions, although the details differ. These operations are heuristically effective, but are intrinsically inefficient as the new solutions are not based on any valid prediction.

Both MOGA [5]–[7] and MOPSO [8], [9] algorithms have found use in the accelerator design studies in recent years. There is a strong incentive to develop more powerful methods as the design of future accelerators is becoming more challenging.

xiahuang@slac.stanford.edu

X. Huang is a staff scientist at SLAC National Accelerator Laboratory, Menlo Park, CA 94025,

In this study, we propose to use posterior Gaussian process (GP) [10]–[15] models to generate or select new trial solutions. A posterior Gaussian process is a non-parametric, analytic model derived from a prior Gaussian process and the sample data, based on the Bayesian inference. It serves as a surrogate model of the actual physics model that governs the system and produces the sample data. The GP model can be used to predict the performance of solutions not yet evaluated, along with an uncertainty estimate. With the GP model, only solutions with a high likelihood of producing good performances will be selected for evaluation. Therefore, the efficiency of the algorithm will be substantially higher than algorithms that do not have the prediction ability.

In Section II we gives a brief introduction to the Gaussian process regression and optimization. The multi-generation GP optimzier is described in Section III. A test of the new optimizer with analytic functions is presented in Section IV. The conclusion is given in Section V.

II. Gaussian process regression and optimization

The Gaussian process regression is a type of Bayesian inference, in which one combines a prior statistical model and the observed evidences to deduce knowledge of the actual statistical model, based on Bayes' theorem of the conditional probabilities.

A Gaussian process is a statistical model of the distribution of a random function over space or time (distribution of a parameter space is assumed in the present context). The GP not only gives the probability distribution of the function at one location, but also its joint distribution with the function value at any other location. The joint distribution is a normal distribution. For an unknown function over a parameter space, a prior Gaussian process can be specified with the prior mean function $m(\mathbf{x})$ and the kernel function $k(\mathbf{x}, \mathbf{x}')$, where $\mathbf{x}$ and $\mathbf{x}'$ are vectors that represent points in the parameter space. Without any knowledge about the function, the prior mean is often assumed $m(\mathbf{x}) = 0$. The kernel function represents the covariance of the function values at two locations. It is often assumed to take the squared exponential form [14], [15],

$$k(\mathbf{x}, \mathbf{x}') = \Sigma_f^2 \exp(-\frac{1}{2}(\mathbf{x} - \mathbf{x}')^T \mathbf{\Theta}^{-2}(\mathbf{x} - \mathbf{x}')), \qquad (1)$$

where Σ_f is the estimated variance of the function, $\mathbf{\Theta} = \text{diag}(\theta_1, \theta_2, \cdots, \theta_n)$ is a diagonal matrix and the θ_i parameters specify the correlation of the function values at two points separated in space in the direction of x_i coordinate.

After a number of sample data points, given as $(\mathbf{x}_i, f_i = f(\mathbf{x}_i))$, $i = 1, 2, \cdots, t$, are taken from the parameter space,

we would like to know the function value at a new point $\mathbf{x}_{t+1}$. From the prior GP, the joint distribution of the sample data and the new point is given by a multi-variate normal distribution,

$$\mathcal{N}\left(\mathbf{0}, \begin{pmatrix} \mathbf{K} & \mathbf{k} \\ \mathbf{k}^T & k(\mathbf{x}_{t+1}, \mathbf{x}_{t+1}), \end{pmatrix}\right) \tag{2}$$

where $\mathbf{K}$ is the kernel matrix, whose elements are $K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$, and the kernel vector is given by $k_i = k(\mathbf{x}_i, \mathbf{x}_{t+1})$. The prior joint distribution function, Eq. 2, and the evidence by the sample data set allow us to calculate the conditional distribution of the function value at point $\mathbf{x}_{t+1}$, which is a normal distribution given by its mean and standard deviation [14],

$$\mu_{t+1} = \mathbf{k}^T \mathbf{K}^{-1} \mathbf{f}_t, \tag{3}$$

$$\sigma^2_{t+1} = k(\mathbf{x}_{t+1}, \mathbf{x}_{t+1}) - \mathbf{k}^T \mathbf{K}^{-1} \mathbf{k}. \tag{4}$$

The expected mean, μ_{t+1}, is an estimate of the function value and the standard deviation σ_{t+1} gives the uncertainty.

Eqs. 3-4 are the posterior model of the actual function. It is worth noting that this is a non-parametric model. The sample data enter the model directly. The posterior distribution not only can be used to predict the function values in the parameter space, but also can be used to optimize the function.

In a GP optimizer, the posterior model is used to choose the next trial solution. With the posterior GP, an optimization algorithm is used to look for a point $\mathbf{x}_{t+1}$ that the model predicts to yield the largest gain, which is then evaluated on the real system. After that, the new data point enters the sample data set and the GP model is updated accordingly. The measure of the gain is represented by the acquisition function, a popular choice of which is the upper confidence bound (UCB) for a maximization problem [16]. For a minimization problem, it is the lower confidence bound (LCB), given by

$$\text{GP-LCB}(\mathbf{x}) = \mu(\mathbf{x}) - \kappa\sigma(\mathbf{x}), \tag{5}$$

where $\kappa > 0$ is a constant. A suitable value of κ is used to balance the exploitary and the exploratory strategies - a small κ is exploitary and a large κ is exploratory. Taking a large κ is to take some risk by going into the less certain area in the parameter space in exchange for the opportunity to yield a big gain.

After every new data point is added, the GP model is updated, which requires the inversion of the kernel matrix. During the search for the trial solution, many matrix multiplications are performed. These calculation can be time consuming if the dimension of the matrix is large. Therefore, the size of the data set is often limited to the order of hundreds.

III. Multi-generation Gaussian process optimizer

The ability of the posterior GP model to approximate the actual model and to predict the performance of a new solution can be very useful in design optimization, where it is common to evaluate thousands or tens of thousands solutions in the search for the optimal design. A design study often has multiple objectives. In the following we propose a multi-objective, multi-generation Gaussian process optimization algorithm that would be ideal for design optimization.

Presently MOGA and MOPSO algorithms are widely used in the design optimization of accelerators. A popular MOGA algorithm is the NSGA-II [2]. It takes an iterative scheme to update a population of solutions. At each iteration, it generates new trial solutions based on the existing ones, using the crossover and mutation operations. In a crossover two solutions are combined to generate a pair of new solutions randomly distributed in between, while a mutation operation modifies a solution with random changes to the parameters. The new trial solutions are evaluated and compared to the existing solutions with a non-dominated sorting. Some solutions replace the existing ones and enter the next generation if they outperform the latter.

The MOPSO [3], [4] algorithm also manipulates a population of solutions iteratively. In this case, each solution is considered a particle in the parameter space. New solutions are generated by shifting the existing solutions in the parameter space by an offset called the velocity. The velocity consists of contributions from three terms: the previous velocity, a shift toward the best solution of the history of the particle (the personal best), and a shift toward a solution in the global best solutions. The velocity and the personal and global best solutions are updated at every iteration.

The MOGA and MOPSO algorithms work because the operations used to generate new solutions tend to produce solutions toward the direction with better performances, which are then selected and used for the next generation. However, there is no guarantee that the crossover and mutation operations or the shift by the velocity will yield better solutions. No information is extracted from the previous function evaluations other than the selection of the best solutions.

When we apply GP regression to model the existing solutions, we would be able to determine which new solutions have a high probability of yielding good performances. We can optimize with the posterior GP model to produce promising trial solutions. Or we can simply generate a large quantity of potential new solutions, evaluate them with the GP model, and use the outcome to select the solutions with a potential to yield a significant improvement. By selecting only these solutions for the computationally expensive function evaluation, we could substantially improve the efficiency of the algorithm.

The new algorithm, which may be referred to as the multi-objective, multi-generation Gaussian process optimizer (MG-GPO), also works iteratively. The initial population of solutions may be randomly generated, throughout the parameter space, or within a small region in the parameter space. The population of solutions, N, is fixed.

At each iteration, N new solutions will be generated and evaluated. The set of solutions evaluated on iteration n may be labeled $\mathcal{F}_n$. The set $\mathcal{F}_n$ is combined with the N best solutions from the last iteration, which form a set labeled $\mathcal{G}_{n-1}$, and the combined set is sorted with the non-dominated sorting [2], from which the population of N best solutions is updated.

A GP model is constructed for each objective, which has its own set of model parameters, $\mathbf{\Theta}^{(j)}$ and $\Sigma_f^{(j)}$. We also give the prior GP model a non-zero mean, $m_j(\mathbf{x}) = \bar{\mu}^{(j)}$. The value of $\bar{\mu}^{(j)}$ and $\Sigma_f^{(j)}$ are given by the mean and standard deviation

of the function values of the previous data set, respectively. With the non-zero mean, Eq. 3 is replaced with

$$\mu_{t+1} \quad = \mathbf{k}^T \mathbf{K}^{-1}(\mathbf{f}_t - \bar{\mu}) + \bar{\mu}. \tag{6}$$

The use of a non-zero mean helps avoid an abrupt change in the function value when searching in the transition region between the sampled area and the un-sampled areas. A wrong mean value could produce a bias that either pull the search into the unexplored territory or prevent the search into it.

While it is possible to use a multi-objective optimization algorithm to optimize the surrogate models, produce the Pareto front, and use the solutions in the Pareto front for the actual evaluation, we adopt a simple approach to sample the area around the existing best solutions. New solutions are generated through the mutation and crossover operations. For each solution in the previous population of best solutions, $\mathcal{G}_{n-1}$, m_1 new solutions are by mutation and another m_2 solutions are by crossover. Mutation is done by randomly shifting each individual parameter by up to a certain percentage of the parameter range. The variation is drawn from a uniform distribution. Crossover is done by paring the solution and another random solution in $\mathcal{G}_{n-1}$ and choosing a random intermediate point between the two solutions. Obviously, there are better ways to generate new solutions, for example, by using the gradient afforded by the posterior GP model. Nonetheless, the present simple approach is adequate to demonstrate the advantage of the GP method. Besides, we can always increase the number of new solutions to improve the sampling of the GP models as the cost of evaluating the GP models is usually negligible compared to the actual physics simulation.

The $(m_1 + m_2)N$ solutions are then evaluated with the GP models, which give the expected mean and standard deviation for each objective function. We choose the GP-LCB acquisition functions as the figure of merit for the solutions. A small κ value, such as $\kappa = 0.5$, was found to work better than a relatively larger value (such as $\kappa = 2$). A non-dominated sorting is then performed over the $(m_1 + m_2)N$ solutions, from which N solutions are selected for the actual design simulation. These N solutions form the set $\mathcal{F}_n$, which is then combined with $\mathcal{G}_{n-1}$ and another non-dominated sorting is used to updated the N best solutions, yielding $\mathcal{G}_n$.

The GP models are updated at the end of the iteration. The sample data used for the GP models are the combined set of $\mathcal{F}_n$ and $\mathcal{G}_n$. There will be some redundant data points, as some solutions in $\mathcal{F}_n$ has just entered $\mathcal{G}_n$. The duplicate points can be eliminated. It can also be left in, as it does not pose a difficulty.

The MG-GPO algorithm is summarized below (with G_{max} being the maximum number of generations)

$n \leftarrow 0$, Initialize the population, $\mathcal{G}_0$
Evaluate all solutions in $\mathcal{G}_0$
Construct Gaussian process models, $\mathcal{GP}_0$, with $\mathcal{G}_0$
while $n < G_{max}$ **do**
  $n \leftarrow n + 1$
  For each solution in $\mathcal{G}_{n-1}$, generate m_1 solutions with mutation and m_2 solutions with crossover.
  Evaluate the $(m_1 + m_2)N$ solutions with $\mathcal{GP}_{n-1}$
  Use non-dominated sorting to select N best solutions, which forms the set $\mathcal{F}_n$.
  Evaluate the solutions in $\mathcal{F}_n$.
  Use non-dominated sorting to select N best solutions from the combined set of $\mathcal{GP}_{n-1}$ and $\mathcal{F}_n$, the result of which form $\mathcal{G}_n$.
  Construct Gaussian process models, $\mathcal{GP}_n$, with solutions in $\mathcal{F}_n$ and $\mathcal{G}_n$.
end while

IV. Test with analytic functions

The MP-GPO algorithm has been implemented with a framework similar to that of the PSO described in Ref. [9]. Function minimization is assumed. The parameter range is normalized to $[0, 1]$. The correlation length can be set in the normalized coordinates. The GP-LCB acquisition function with $\kappa = 0.5$ is used. The multiplication factors are $m_1 = m_2 = 20$ by default. These algorithm parameters can be changed for different problems. Mutation of a solution is done by varying each normalized parameter with a random deviation drawn from the uniform distribution of $[-0.05, 0.05]$. The deviation range can also be changed.

Four test cases have been used to test the performance of the MG-GPO algorithm in comparison to the NSGA-II and PSO algorithms. The test cases are taken from Ref. [2]. All test cases have two objective functions.

The NSGA-II code used in the test was obtained online from Matlab Central File Exchange [17]. The crossover probability is set to 90%. The distribution indices for the simulated binary crossover (SBX) and mutation operations are $\eta_c = 20$ and $\eta_m = 20$, respectively [18]. For the PSO algorithm, the weight factors in the velocity composition are $w = 0.4$ and $r_1 = r_2 = 1$. The PSO algorithm also includes a mutation operation, with a probability rate of $1/P$, where P is the number of variables.

The population size is set to $N = 100$ for all algorithms and all test cases. The algorithms are run for 100 generations. In all test cases, the correlation length parameters of MG-GPO are set to $\theta_j^{(i)} = 0.4$ for all parameters and both objective functions ($i = 1, 2$).

The initial solutions are randomly distributed, with parameters drawn from a uniform distribution in the parameter range.

A. Test case 1 - FON

The first test case comes from Refs. [2], [19] and is referred to as FON. There are $P = 3$ variables. The two objective functions are defined as

$$f_1(\mathbf{x}) \quad = \quad 1 - \exp(-\sum_{i=1}^{3}(x_i - \frac{1}{\sqrt{3}})^2), \tag{7}$$

$$f_2(\mathbf{x}) \quad = \quad 1 - \exp(-\sum_{i=1}^{3}(x_i + \frac{1}{\sqrt{3}})^2). \tag{8}$$

The parameter ranges are $[-4, 4]$ for all three variables, x_i, $i = 1, 2$, and 3. The optimal solutions are with $x_1 = x_2 = x_3 \in [-\frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}}]$.

The evolution of the best solutions are shown in Fig. 1, where generation 0 represent the initial distribution. At generation 2, solutions by the MG-GPO algorithm has already

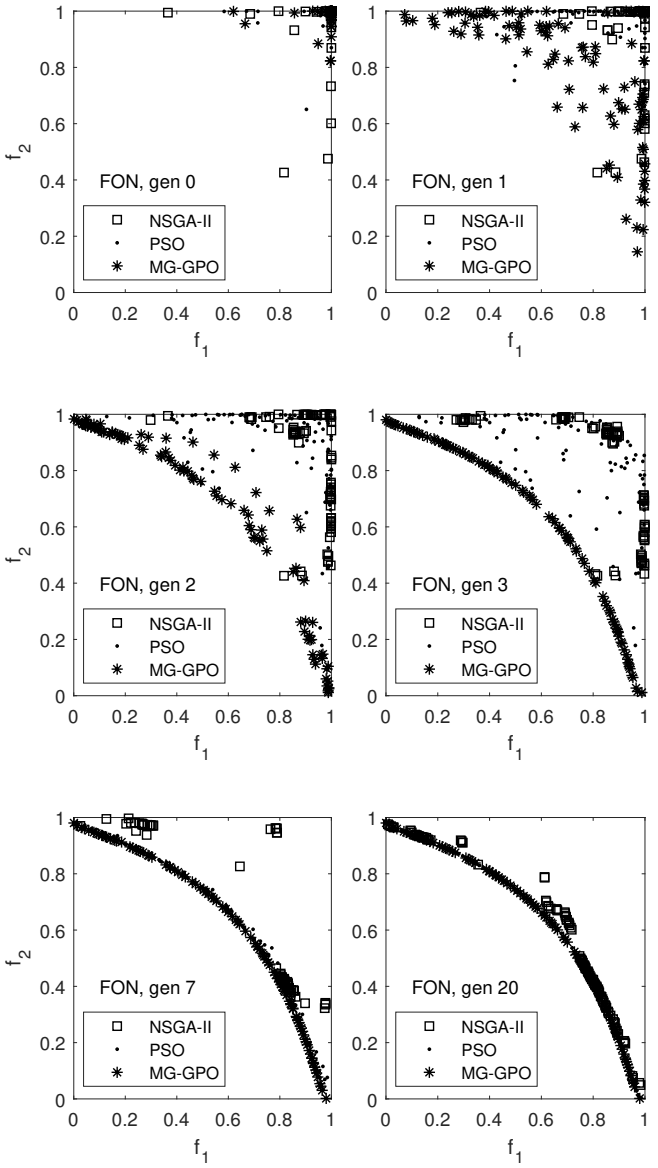

Fig. 1. The best solutions at a given generation during the optimization of the first test case (FON) with the NSGA-II, PSO, and MG-GPO algorithms. Generation 0 (initial distribution), 1, 2, 3, 7, and 20 are shown.

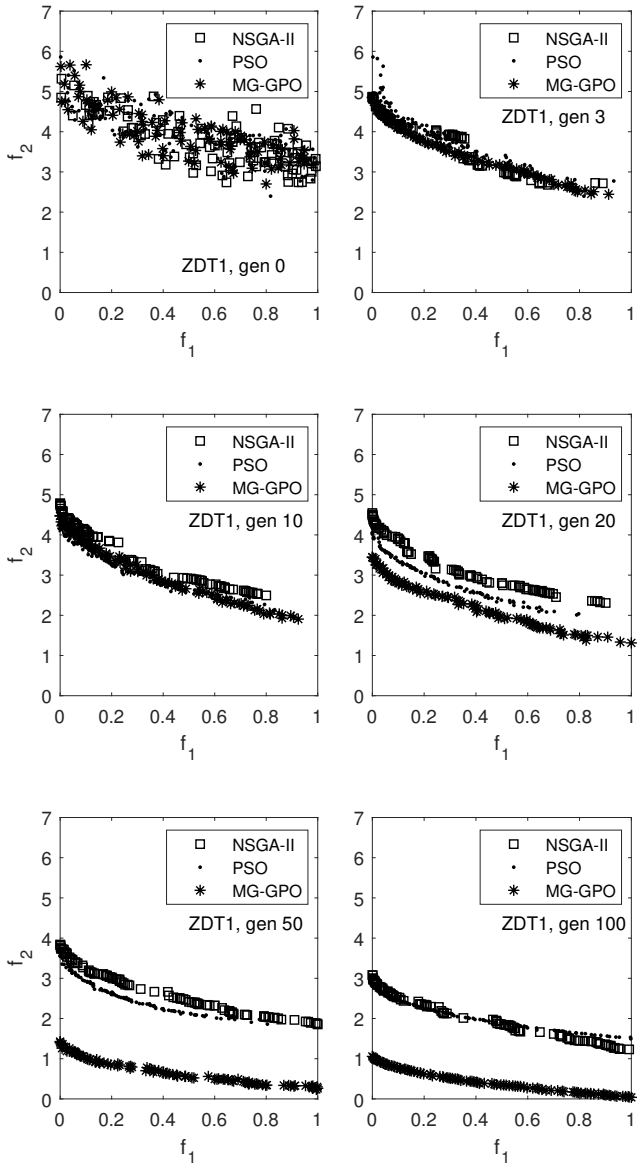

Fig. 2. The best solutions at a given generation during the optimization of the second test case (ZDT1) with the NSGA-II (GA), PSO, and MG-GPO algorithms. Generation 0 (initial distribution), 3, 10, 20, 50, and 100 are shown.

taken the shape of the Pareto front, while its solutions at generation 3 have converged. The PSO algorithm converges faster than NSGA-II in this case. But it does not converge to the Pareto front until generation 7. The solutions of the NSGA-II algorithms has not yet converged to the Pareto front at generation 20.

B. Test case 2 - ZDT1

The second test case uses two functions with $P = 30$ variables, defined by

$$f_1(\mathbf{x}) = x_1, \tag{9}$$

$$f_2(\mathbf{x}) = g(\mathbf{x})(1 - \sqrt{x_1/g(\mathbf{x})}), \tag{10}$$

(11)

with

$$g(\mathbf{x}) = 1 + \frac{9}{P-1}(\sum_{i=2}^{P} x_i). \tag{12}$$

This test case is referred to as ZDT1 [20]. The parameter ranges are [0, 1] for all variables. The optimal solutions are with $x_i = 0$ for $i = 2, 3, \cdots, n$ and $x_1 \in [0, 1]$.

Figure 2 shows the test results in 100 generations. The MG-GPO algorithm converges much faster than NSGA-II and the PSO algorithm. At generation 20, its best solution front is ahead of the other two algorithms. At generation 50, it has nearly converged to the Pareto front, while the other two are still at about the position of GP-MPO at its 20th generation.

C. Test case 3 - ZDT2

The third test case we tried is ZDT2 [20], which is defined similarly as ZDT1, except that the $f_2(\mathbf{x})$ function is redefined as

$$f_2(\mathbf{x}) = g(\mathbf{x})[]1 - (x_1/g(\mathbf{x}))^2]. \quad (13)$$

The number of variables and the ranges of the parameters are the same as ZDT1.

The test results for ZDT2 are shown in Fig. 3. Here we find that MG-GPO is in the leading position at generation 3. Its convergence speed is substantially faster than the other two algorithms. At generation 50, MG-GPOS has nearly converged to the Pareto front, while NSGA-II and PSO lags behind by a large distance.

D. Test case 4 - ZDT3

Test was also done for the case ZDT3 [20]. The definition is similar to ZDT1, except that the $f_2(\mathbf{x})$ function is redefined as

$$f_2(\mathbf{x}) = g(\mathbf{x})(1 - \sqrt{x_1/g(\mathbf{x})} - \frac{x_1}{g(\mathbf{x})} \sin 10\pi x_1). \quad (14)$$

The number of variables and the ranges of the parameters are the same as ZDT1. The Pareto front of this case consists of disconnected stripes.

Fig. 4 shows the test results for ZDT3. At generation 10, the front of the best solutions for the three algorithms almost overlap in the objective space. There is no one better than the others at this point. However, at generation 20, MG-GPO has taken the leading position. The gap between MG-GPO and NSGA-II, which is in the second place, becomes wider at generation 50, and later at generation 100. At generation 100, MG-GPO has nearly converged to the Pareto front.

The front of MG-GPO lost the two stripes at $f_1 > 0.4$ early on in the run as the solutions in the two stripes to the left produce many good candidates. It would be useful to introduce some measures in the GP-based selection operation to promote diversity in the trial solutions.

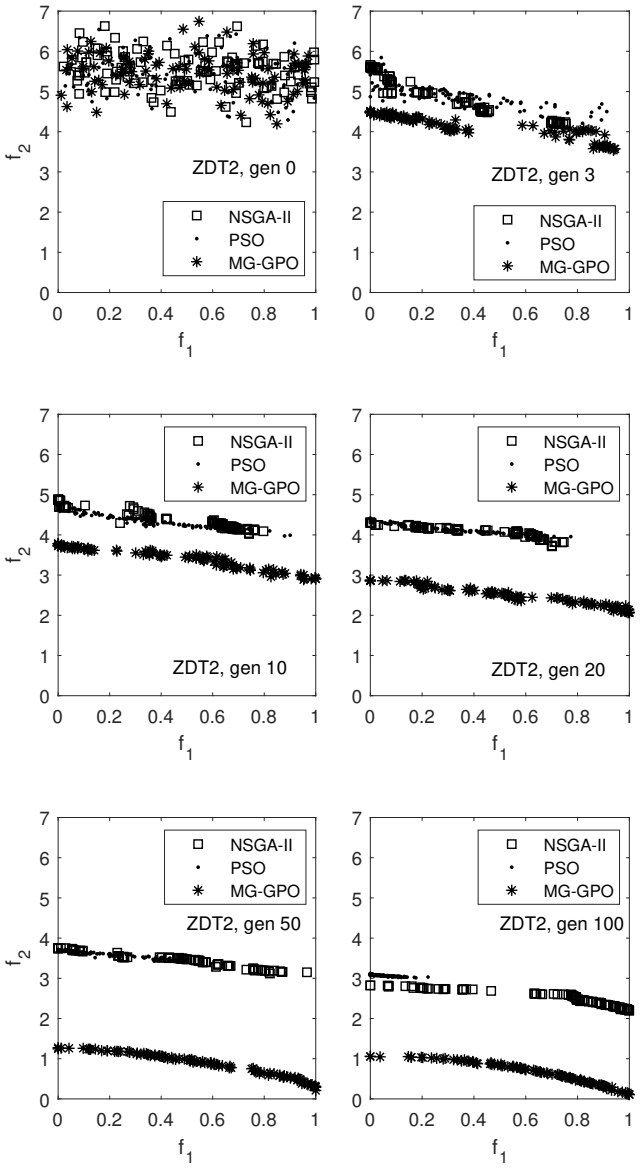

Fig. 3. The best solutions at a given generation during the optimization of the second test case (ZDT2) with the NSGA-II (GA), PSO, and MG-GPO algorithms. Generation 0 (initial distribution), 3, 10, 20, 50, and 100 are shown.

V. Conclusion

We proposed a new multi-objective stochastic optimization algorithm that is based on Gaussian process regression. The new algorithm update a population of solutions iteratively. At each iteration, it constructs a posterior Gaussian process and uses it as a surrogate model of the actual system to be optimized. A large number of candidate solutions are generated and evaluated with the surrogate model and the results are used to select a small number of promising solutions to be evaluated on the real system (e.g., by physics simulation).

The new algorithm, referred to as multi-generation Gaussian process optimizer (MG-GPO), has been tested with analytic functions. In all test cases, a substantially faster convergence speed is found than the NSGA-II and PSO algorithms. The new algorithm would be very useful for design optimization of large systems where a search for optimal solutions in a multi-dimensional parameter space is needed.

Acknowledgment

This work was supported by the U.S. Department of Energy, Office of Science, Office of Basic Energy Sciences, under Contract No. DE-AC02-76SF00515 and FWP 2018-SLAC-100469 and by Computing Science, Office of Advanced Scientific Computing Research under FWP 2018-SLAC-100469ASCR.

References

[1] K. Deb and D. Kalyanmoy, *Multi-Objective Optimization Using Evolutionary Algorithms*. New York, NY, USA: John Wiley & Sons, Inc., 2001.

[2] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, April 2002.

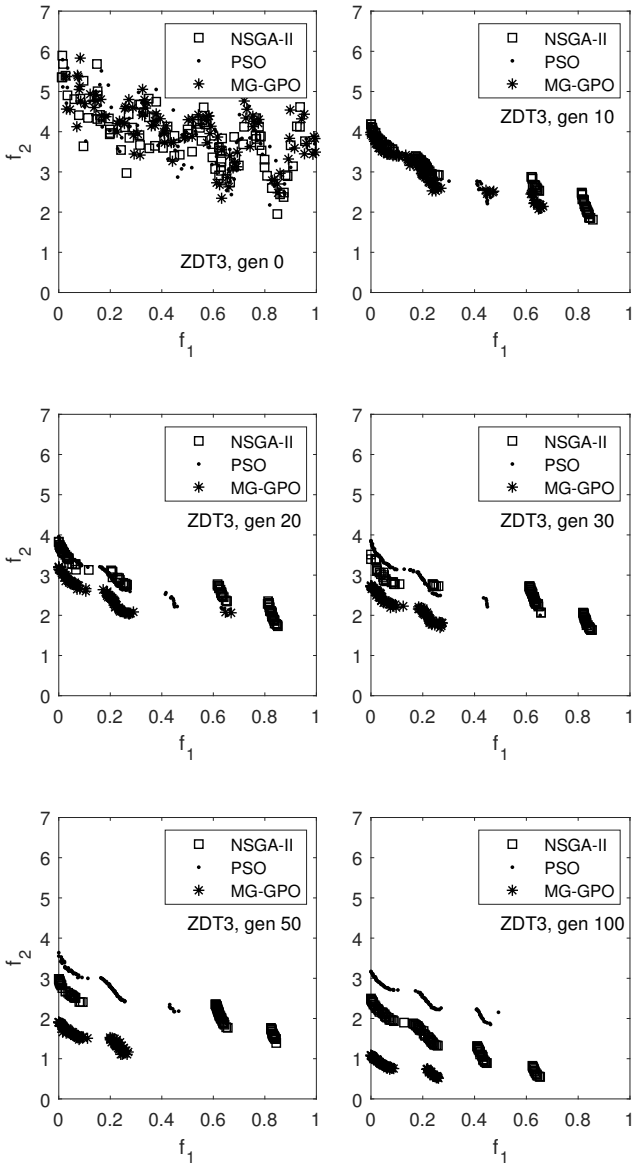

Fig. 4. The best solutions at a given generation during the optimization of the second test case (ZDT3) with the NSGA-II (GA), PSO, and MG-GPO algorithms. Generation 0 (initial distribution), 10, 20,30, 50, and 100 are shown.

[3] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of ICNN'95 - International Conference on Neural Networks*, vol. 4, Nov 1995, pp. 1942–1948 vol.4.

[4] M. Clerc and J. Kennedy, "The particle swarm - explosion, stability, and convergence in a multidimensional complex space," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 1, pp. 58–73, Feb 2002.

[5] I. V. Bazarov and C. K. Sinclair, "Multivariate optimization of a high brightness dc gun photoinjector," *Phys. Rev. ST Accel. Beams*, vol. 8, p. 034202, Mar 2005. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevSTAB.8.034202

[6] L. Yang, D. Robin, F. Sannibale, C. Steier, and W. Wan, "Global optimization of the magnetic lattice using genetic algorithms," in *Proceedings of EPAC'08*, 2008, pp. 3050–3052.

[7] M. Borland, V. Sajaev, L. Emery, and A. Xiao, "Direct methods of optimization of storage ring dynamic and momentum aperture," in *Proceedings of PAC'09*, 2009, pp. 3850–3852.

[8] X. Pang and L. Rybarcyk, "Multi-objective particle swarm and genetic algorithm for the optimization of the lansce linac operation," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 741, pp. 124 – 129, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0168900213017464

[9] X. Huang and J. Safranek, "Nonlinear dynamics optimization with particle swarm and genetic algorithms for spear3 emittance upgrade," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 757, pp. 48 – 53, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0168900214004914

[10] H. J. Kushner, "A new method of locating the maximum point of an arbitrary multipeak curve in the presence of noise," *Journal of Basic Engineering*, vol. 86, no. 1, p. 97, 1964. [Online]. Available: https://doi.org/10.11152F1.3653121

[11] A. G. Zhilinskas, "Single-step bayesian search method for an extremum of functions of a single variable," *Cybernetics and Systems Analysis - CYBERN SYST ANAL-ENGL TR*, vol. 11, pp. 160–166, 01 1975.

[12] J. Mockus, "On bayesian methods for seeking the extremum and their application." in *IFIP Congress*, 1977, pp. 195–200. [Online]. Available: http://dblp.uni-trier.de/db/conf/ifip/ifip1977.htmlMockus77

[13] 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, Dec 1998. [Online]. Available: https://doi.org/10.1023/A:1008306431147

[14] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. the MIT Press, 2006.

[15] E. Brochu, V. M. Cora, and N. de Freitas, "A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning," *CoRR*, vol. abs/1012.2599, 2010. [Online]. Available: http://arxiv.org/abs/1012.2599

[16] P. Auer, "Using confidence bounds forexploitation-exploration trade-offs," *ournal of Machine Learning Research*, vol. 3, pp. 397–422, 2002.

[17] A. Seshadri, "Nsga - ii: A multi-objective optimization algorithm," https://www.mathworks.com/matlabcentral/fileexchange/10429-nsga-ii-a-multi-objective-optimization-algorithm, Retrieved June 2019.

[18] K. Deb and R. B. Agrawal, "Simulated binary crossover for continuous search space," *Complex Systems*, vol. 9, pp. 115–148, 1995.

[19] C. M. Fonseca and P. J. Fleming, "Multiobjective optimization and multiple constraint handling with evolutionary algorithms. ii. application example," *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, vol. 28, no. 1, pp. 38–47, Jan 1998.

[20] E. Zitzler, K. Deb, and L. Thiele, "Comparison of multiobjective evolutionary algorithms: Empirical results," *Evol. Comput.*, vol. 8, no. 2, pp. 173–195, Jun. 2000. [Online]. Available: http://dx.doi.org/10.1162/106365600568202

Xiaobiao Huang Xiaobiao Huang graduated from Tsinghua University with a BS in physics and a BE in computer science. He obtained a PhD degree in Physics from Indiana University, Bloomington in 2005. He has been a staff scientist at SLAC National Accelerator Laboratory since 2006.