ISBN: 978-3-95450-214-1 ISSN: 2673-5490 doi:10.18429/JACoW-IPAC2021-WEPAB304

MULTI-OBJECTIVE MULTI-GENERATION GAUSSIAN PROCESS OPTIMIZER*

X. Huang† [1], Z. Zhang, M. Song, SLAC National Accelerator Laboratory, Menlo Park, USA
[1]now at Argonne National Laboratory, Lemont, USA

Abstract

We present a multi-objective evolutionary optimization algorithm that uses Gaussian process (GP) regression-based models to 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 higher efficiency in exploring the parameter space. Simulations with multiple test cases show that the new algorithm has a substantially higher convergence speed and stability than NSGA-II, MOPSO, and some other recent preselection-assisted algorithms.

INTRODUCTION

In the particle accelerator field, there are many challenging design optimization problems, such as lattice design for synchrotron light sources [1–3], beamline design for photoinjectors [4], and cavity design for superconducting Radio Frequency (SRF) components. The design of such complex systems 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 evolutionary algorithms (MOEAs) have 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) [5–7] and multi-objective particle swarm optimization (MOPSO) [8–11].

Because the design problems often require time-consuming simulations. Therefore, high efficiency of the optimization algorithm is crucial. Both MOGA and PSO methods employ stochastic operations to produce new solutions with existing good solutions, although the details differ. The stochastic operations increase the ability of the algorithms in searching for global optima. However, they are not very efficient for fast convergence.

In this study, we propose a multi-objective multi-generation Gaussian process optimizer (MG-GPO) for design optimization. The method uses Gaussian process (GP) regression to build surrogate models for the selection of trial solutions. Similar to MOGA and MOPSO, it generates and manipulates a population of solutions with stochastic operations in an iterative manner. The difference is that posterior GP models are constructed and updated in each iteration and are used to select the trial solutions for the actual evaluation. The model-based selection substantially boosts the efficiency of the algorithm.

This algorithm has been successfully applied to the storage ring light source lattice design optimization [12] as well as online problems [13].

GAUSSIAN PROCESS REGRESSION

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 the distribution of a random function in space through the mean function $m(\mathbf{x})$ and the kernel function $k(\mathbf{x}, \mathbf{x}')$. 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],

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

where Σ_f is the estimated variance of the function, $\mathbf{\Theta} = \mathrm{diag}(\theta_1, \theta_2, \cdots, \theta_n)$ 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 can obtain a posterior distribution of the function at a new point $\mathbf{x}_{t+1}$, which is a Gaussian distribution with the mean and standard deviation given by [14],

$$\mu_{t+1} = \mathbf{k}^T \mathbf{K}^{-1} \mathbf{f}_t, \quad (2)$$

$$\sigma_{t+1}^2 = k(\mathbf{x}_{t+1}, \mathbf{x}_{t+1}) - \mathbf{k}^T \mathbf{K}^{-1} \mathbf{k}. \quad (3)$$

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

The GP model can be used for optimization. For a minimization problem, the lower confidence bound (LCB), given by

$$\text{GP-LCB}(\mathbf{x}) = \mu(\mathbf{x}) - \kappa \sigma(\mathbf{x}), \quad (4)$$

is often used as the target function, where $\kappa \geq 0$ is a constant. A suitable value of κ is used to balance the exploitation and the exploration strategies - a small κ favors exploitation and a large κ favors exploration.

* Work supported by DOE, Office of Science, Office of Basic Energy Sciences, DE-AC02-76SF00515 and FWP 2018-SLAC-100469 Computing Science, Office of Advanced Scientific Computing Research, FWP 2018-SLAC-100469ASCR.
† xiahuang@slac.stanford.edu

ISBN: 978-3-95450-214-1 ISSN: 2673-5490 doi:10.18429/JACoW-IPAC2021-WEPAB304

MULTI-GENERATION GAUSSIAN PROCESS OPTIMIZER

By combining the iterative, population-based framework of MOGA and PSO with GP regression, we developed a new algorithm, the multi-generation Gaussian process optimizer (MG-GPO). 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 [6], 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, $\boldsymbol{\Theta}^{(j)}$ 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, the prediction is given by:

$$\mu_{t+1} \quad = \mathbf{k}^T \mathbf{K}^{-1} (\mathbf{f}_t - \bar{\mu}) + \bar{\mu}. \tag{5}$$

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.

New solutions are generated through the mutation, crossover, and flocking 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 [16] and crossover [17] techniques adopted are the same as in NSGA-II.

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 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 κ parameter in GP-LCB can have a significant impact to the behavior of the algorithm. A large κ value encourages exploration of the parameter space but in the same time may not take full advantage of the learned model. Conversely, a small κ value better exploits the model but may not sufficiently explore the parameter space. It could be argued that at the beginning of an optimization a large κ is preferred as more exploration is needed in order to discover the area in the parameter space with good solutions. A small κ would be preferred in the later stage as the algorithm converges to a relatively small area where a refined search is needed. We adopted an adaptive scheme to vary κ exponentially generation by generation and found it to yield faster convergence.

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 MG-GPO algorithm is summarized in Algorithm 1 (with G_{max} being the maximum number of generations, ρ the decreasing factor for κ).

Algorithm 1: MG-GPO

```
n ← 0, Initialize the population, G_0. Initialize
  κ ← 2;
Evaluate all solutions in G_0;
Construct Gaussian process models, GP_0, with G_0;
while n < G_max do
    n ← n + 1;
    Update κ with κ ← ρκ;
    For each solution in G_{n-1}, generate m_1
      solutions with mutation and m_2 solutions with
      crossover;
    Evaluate the (m_1 + m_2)N solutions with
      GP_{n-1};
    Use non-dominated sorting to select N best
      solutions, which forms the set F_n;
    Evaluate the solutions in F_n in the actual
      system;
    Use non-dominated sorting to select N best
      solutions from the combined set of G_{n-1} and
      F_n, the results of which form G_n;
    Construct Gaussian process models, GP_n, with
      solutions in F_n and G_n;
end
```

BENCHMARKING STUDY

Benchmarking studies were conducted to demonstrate the fast convergence of the MG-GPO algorithm in comparison with the two classic MOGAs: NSGA-II [6] and MOPSO [3, 18], that MG-GPO based on. Two recent PS algorithms, CPS-MOEA [19] and MOEA/D-SVM [20], were also selected for comparison.

Test Instances

Eight test problems from two test suites, the ZDT [21] series and the UF [22] series: ZDT1, ZDT2, ZDT3, ZDT6, UF1, UF2, UF3 and UF4 have been used to benchmark the performance of the MG-GPO algorithm in comparison to the NSGA-II, MOPSO, CPS-MOEA and MOEA/D-SVM algorithms. These test cases are commonly used for algorithm performance comparison, for example, in Ref. [6]. All test cases have two objective functions, 30 dimension decision space, and assumed to be minimization problems.

ISBN: 978-3-95450-214-1 ISSN: 2673-5490 doi:10.18429/JACoW-IPAC2021-WEPAB304

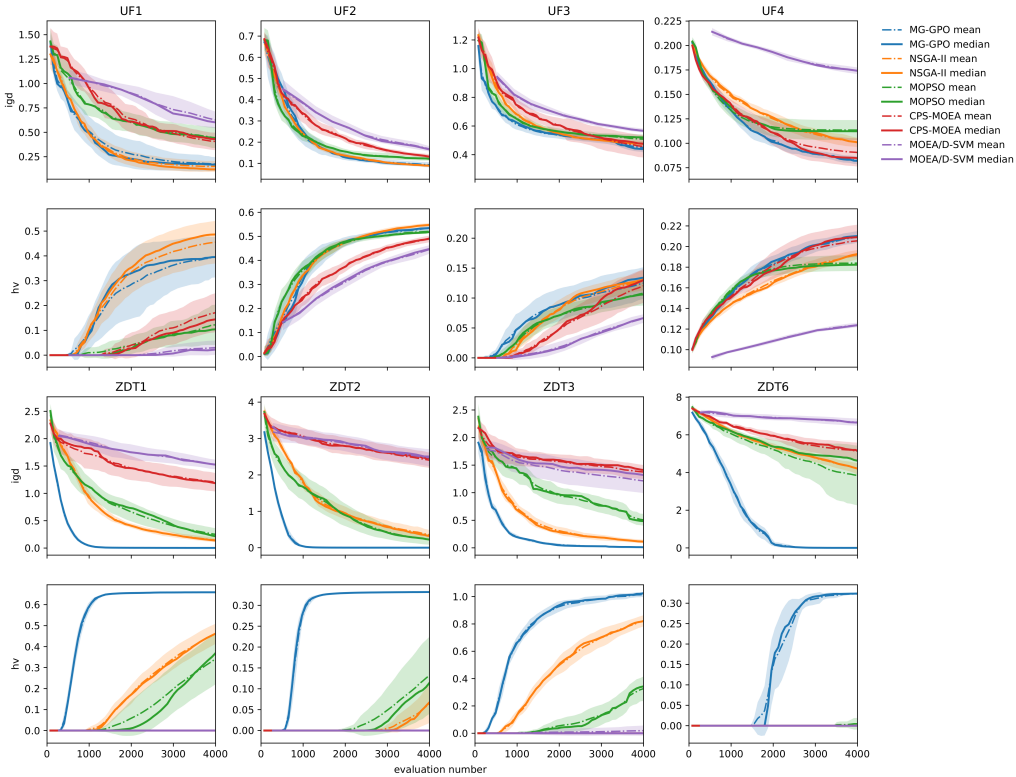

Figure 1: Comparison of HV and IGD between the five algorithms. The odd rows for IGD and the even rows for HV. The columns stand for eight test problems. The filled area around each mean curve indicates the standard deviation of the performance metric for each algorithm. The reference point in the HV calculation is set to $(1, 1)$.

Experimental Settings

For the MG-GPO implementation, the parameter range is normalized to $[0, 1]$. The polynomial mutation (PLM) and simulated binary crossover (SBX) control parameters are set to $\eta_m = 20$ and $\eta_c = 20$, respectively. The initial value $\kappa = 2$ is used and it is scaled down by the factor $\rho = 0.85$ in each generation. The multiplication factors are set to $m_1 = m_2 = 20$. In all test cases, the correlation length parameters of MG-GPO are optimized in each generation with the GPy package [23]. The initial solutions are randomly distributed, with parameters drawn from a uniform distribution in the parameter range.

The population size is set to $N = 80$ for all the algorithms if applicable. The algorithms are run for 4080 evaluations. Each test instance is repeated 10 times.

We chose hypervolume (HV) [24] and inverted generational distance (IGD) [25, 26] as optimization algorithm performance indicators (PI).

Benchmarking Results

Figure 1 shows the evolution of HV and IGD metrics for the algorithms. For each algorithm, the median and the mean values for the 10 runs are shown. The spread of the metrics among the 10 tests for each algorithm is shown with shaded areas. Clearly, for ZDT series test problems, MG-GPO converges faster than the other algorithms. In addition, the performance of MG-GPO is very stable. The spread of the metrics for MG-GPO is considerably smaller than the other algorithms. Tables that summarize the IGD and HV metrics can be found in Ref. [27].

CONCLUSION

We proposed a machine-learning based multi-objective stochastic optimization algorithm, multi-generation Gaussian process optimizer (MG-GPO), for accelerator optimization. It operates on a population of solutions and updates the population generation by generation as is done in MOGA and PSO. In addition, it uses Gaussian process regression models to select trial solutions, which substantially improve its efficiency. The algorithm was tested against a few popular optimization algorithms with commonly used benchmarking problems and was found to have higher convergence speed.

REFERENCES

[1] D. Robin, F. Sannibale, C. Steier, W. Wan, and L. Yang, "Global Optimization of the Magnetic Lattice Using Genetic Algoritihms", in *Proc. 11th European Particle Accelerator Conf. (EPAC'08)*, Genoa, Italy, Jun. 2008, paper THPC033, pp. 3050-3052.

[2] M. Borland, L. Emery, V. Sajaev, and A. Xiao, "Direct Methods of Optimization of Storage Ring Dynamic and Momentum Aperture", in *Proc. 23rd Particle Accelerator Conf. (PAC'09)*, Vancouver, Canada, May 2009, paper TH6PFP062, pp. 3850-3852.

[3] X. Huang and J. Safranek, "Nonlinear dynamics optimization with particle swarm and genetic algorithms for spear3

ISBN: 978-3-95450-214-1 ISSN: 2673-5490 doi:10.18429/JACoW-IPAC2021-WEPAB304

emittance upgrade", *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 757, pp. 48–53, 2014. doi:10.1016/j.nima.2014.04.078

[4] I. V. Bazarov and C. K. Sinclair, "Multivariate optimization of a high brightness dc gun photoinjector", *Physical Review Special Topics - Accelerators and Beams*, vol. 8, p. 034202, Mar. 2005. doi:10.1103/physrevstab.8.034202

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

[6] 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, Apr. 2002. doi:10.1109/4235.996017

[7] Q. Zhang and H. Li, "MOEA/D: A multiobjective evolutionary algorithm based on decomposition", *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 6, pp. 712–731, Dec. 2007. doi:10.1109/tevc.2007.892759

[8] J. Kennedy and R. Eberhart, "Particle swarm optimization", in *Proc. Int. Conf. Neural Networks (ICNN'95)*, Perth, Australia, Nov. 1995, vol. 4, pp. 1942–1948. doi:10.1109/icnn.1995.488968

[9] 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. doi:10.1109/4235.985692

[10] Q. Lin, J. Li, Z. Du, J. Chen, and Z. Ming, "A novel multiobjective particle swarm optimization with multiple search strategies", *European Journal of Operational Research*, vol. 247, no. 3, pp. 732–744, 2015. doi:10.1016/j.ejor.2015.06.071

[11] H. Zille, H. Ishibuchi, S. Mostaghim, and Y. Nojima, "A framework for large-scale multiobjective optimization based on problem transformation", *IEEE Transactions on Evolutionary Computation*, vol. 22, no. 2, pp. 260–275, 2018. doi:10.1109/tevc.2017.2704782

[12] M. Song, X. Huang, L. Spentzouris, and Z. Zhang, "Storage ring nonlinear dynamics optimization with multi-objective multi-generation gaussian process optimizer", *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 976, p. 164273, Oct. 2020. doi:10.1016/j.nima.2020.164273

[13] Z. Zhang, M. Song, and X. Huang, "Online accelerator optimization with a machine learning-based stochastic algorithm", *Machine Learning: Science and Technology*, vol. 2, no. 1, p. 015014, Dec. 2020. doi:10.1088/2632-2153/abc81e

[14] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*, Cambridge, MA, USA: 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", 2010. arXiv:1012.2599

[16] K. Liagkouras and K. Metaxiotis, "An elitist polynomial mutation operator for improved performance of moeas in computer networks", in *Proc. 22nd Int. Conf. on Computer Communication and Networks (ICCCN)*, Nassau, Bahamas, Jul.-Aug. 2013, pp. 1–5. doi:10.1109/icccn.2013.6614105

[17] K. Deb and R. B. Agrawal, "Simulated binary crossover for continuous search space", *Complex Systems*, vol. 9, no. 2, pp. 115– 148, 1995.

[18] 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. doi:10.1016/j.nima.2013.12.042

[19] J. Zhang, A. Zhou, and G. Zhang, "A multiobjective evolutionary algorithm based on decomposition and preselection", in *Bio-inspired Computing - Theories and Applications*, Berlin, Germany: Springer, 2015, pp. 631–642.

[20] X. Lin, Q. Zhang, and S. Kwong, "A decomposition based multiobjective evolutionary algorithm with classification", in *Proc. IEEE Congress on Evolutionary Computation (CEC)*, Vancouver, Canada, Jul. 2016, pp. 3292–3299. doi:10.1109/cec.2016.7744206

[21] E. Zitzler, K. Deb, and L. Thiele, "Comparison of Multiobjective Evolutionary Algorithms: Empirical Results", *Evolutionary Computation*, vol. 8, no. 2, pp. 173–195, Jun. 2000. doi:10.1162/106365600568202

[22] Q. Zhang, A. Zhou, S. Zhao, P. N. Suganthan, W. Liu, S. Tiwari, *et al.*, "Multiobjective optimization test instances for the CEC 2009 special session and competition", University of Essex, Essex, UK, Rep. CES-487, 2009.

[23] GPy: A gaussian process framework in python, http://github.com/SheffieldML/GPy

[24] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. G. Da Fonseca, "Performance assessment of multiobjective optimizers: An analysis and review", *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 2, pp. 117–132, 2003. doi:10.1109/tevc.2003.810758

[25] C. A. C. Coello and M. R. Sierra, "A study of the parallelization of a coevolutionary multi-objective evolutionary algorithm", in *Mexican Int. Conf. on Artificial Intelligence*, Mexico City, Mexico, Apr. 2004, pp. 688–697. doi:10.1007/978-3-540-24694-7_71

[26] M. R. Sierra and C. A. C. Coello, "A new multi-objective particle swarm optimizer with improved selection and diversity mechanisms", CINVESTAV-IPN, Mexico City, Mexico, 2004.

[27] X. Huang, M. Song, and Z. Zhang, "Multi-objective multigeneration gaussian processoptimizer for design optimization", 2019. arXiv:1907.00250