ISBN: 978-3-95450-214-1 ISSN: 2673-5490 doi:10.18429/JACoW-IPAC2021-MOPAB106

ENHANCING THE MOGA OPTIMIZATION PROCESS AT ALS-U WITH MACHINE LEARNING*

Y. Lu†, S.C. Leemann, C. Sun, M.P. Ehrlichman, T. Hellert, H. Nishimura, M. Venturini
Lawrence Berkeley National Laboratory, Berkeley, CA 94720, USA

Abstract

The bare lattice optimization for the linear and nonlinear ALS-U storage ring lattice, even without reverse bending, comprises 11 degrees of freedom (DoF) and is therefore a very complex and highly time-consuming process. This design process relies heavily on multi-objective genetic algorithms (MOGA), usually requiring many months of experienced scientists' time. The main problem lies in having to evaluate vast numbers of candidate lattices due to the stochastic process of MOGA. Although almost all of these candidates are eventually rejected, they nevertheless require extensive particle tracking to arrive at a Pareto front. We therefore propose a novel Machine Learning (ML) pipeline in which nonlinear tracking is replaced by two well-trained neural networks (NNs) to predict dynamic aperture (DA) and momentum aperture (MA) for any lattice candidate. Initial training of these models takes only several minutes on conventional CPUs while predictions are then rendered near instantaneously. We present this novel method and demonstrate the resulting orders of magnitude speedup of the ML-enhanced MOGA process on a 2-DoF problem as well as first results on a more complex 11-DoF problem.

INTRODUCTION

MOGA is the most commonly used algorithm in lattice optimization for ultra-high brightness storage rings [1–4]. Multiple variants of MOGA are available, among which the Pareto-based algorithm NSGA-II is the most popular [5, 6]. However, it still takes vast amounts of time for MOGA to arrive at a Pareto front due to costly evaluations of DA and MA based on many-turn tracking. As ML has proven its efficiency in building computational models to solve complex data-intensive problems compared to traditional statistical methods [7], several groups have attempted applying ML techniques to speed up MOGA [8–10].

MACHINE LEARNING APPROACH

In this study, we built 8-layer fully-connected NNs for prediction of DA[1] and MA since DA/MA tracking requires the majority of runtime spent by a MOGA run. With the application of ML, we can reduce overall runtime from months to days. Our proposed ML pipeline is very simple: we pre-process training data acquired from prior simulations (cf. next Section) and use this data to obtain two well-trained models using the NN depicted in Fig. 1. Each NN has 8 fully-connected layers of various sizes with the first 7 layers using ReLU as activation function [11]. Training requires from several to ≈ 30 min on a single core depending on the amount of training data. We then use these two NN models to replace DA/MA particle tracking in MOGA while the rest of the MOGA setup remains the same as in the original tracking-based MOGA (Tr-MOGA). We evaluate this ML-based MOGA (ML-MOGA) on a simpler 2-DoF problem and a more complex 11-DoF problem.

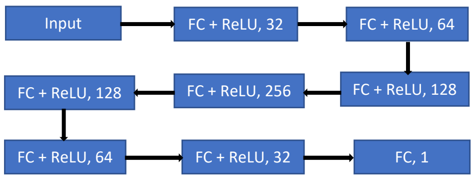

Figure 1: 8-layer fully-connected (FC) NN architecture for DA and MA prediction. Output dimension is indicated.

OPTIMIZATION IN 2 DoF

Lattice optimization at ALS-U—before introduction of reverse bending and high-field bends—consists of 11 DoF as well as several constraints and objectives. 9 DoF are linear (quadrupoles) while 2 DoF are nonlinear (2 harmonic sextupoles, the 2 chromatic sextupoles are constrained to correct linear chromaticity) [12]. In a first study, the 9 linear DoF were fixed, leaving only 2 harmonic sextupoles (SH1, SH2), for a data input size of 2. We explored primarily two methods to generate training data. The simplest method is to use some initial data from a previous Tr-MOGA run. This, however, requires at least a partial Tr-MOGA run completed first, which can easily takes days even on the NERSC cluster. Furthermore, this data is not necessarily transferable when machine settings are adapted throughout optimization. A superior method relies on generating random data which is uniformly distributed in input parameter space, in this case the two harmonic sextupoles: in units of b_2 [m^{-3}] each constrained to [-580, 580]. A random sampling of 115 × 115 was chosen which considerably shortens the initially required tracking runtime. This method does not require any prior Tr-MOGA and can quickly be re-applied for various machine settings.

After optimizing the NN architecture, lowest root mean square errors (RMSEs) for two ML models for DA and MA are 17.9 and 4.53×10^{-4}, respectively. Compared to target values around 1100 and 2.7%, this corresponds to relative errors well below 2%. The difference in runtime between

* This research is funded by the US Department of Energy (BES & ASCR Programs), and supported by the Director of the Office of Science of the US Department of Energy under Contract No. DEAC02-05CH11231.
† yupinglu@lbl.gov
[1] Throughout this study, we actually use total diffusion rate [3] as a proxy for DA since this ensures convergence towards more stable lattices.

tracking and NN lookup is dramatic: for each child in every generation, tracking DA and MA requires about 88 s, whereas the NN lookup takes only 16.3 ms when run on the same NERSC cluster (2048 cores) rendering a direct speedup beyond 5300. An example for the final solutions are visualized in Fig. 2 and related input variables are shown in Fig. 3. Blue dots represent the Tr-MOGA Pareto front

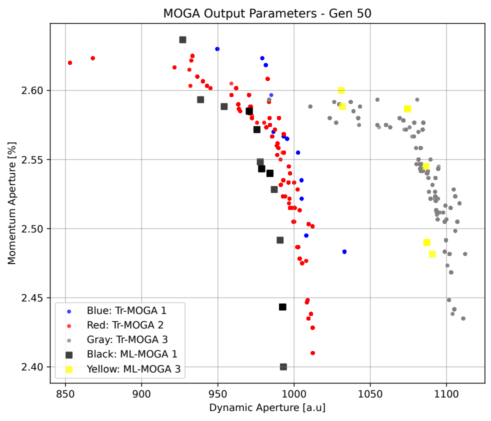

Figure 2: Solution space comparison between Tr-MOGA runs and ML-MOGA runs with different random seeds.

at generation 50[2]. Since NN predictions are never perfect, inputs of the final generation of ML-MOGA are tracked for validation. From these validation results, we can then extract the rank-1 children (black squares) which end up very close to the Tr-MOGA Pareto front. In input parameter space as well, Tr-MOGA and ML-MOGA children are very close, as shown in Fig. 3.

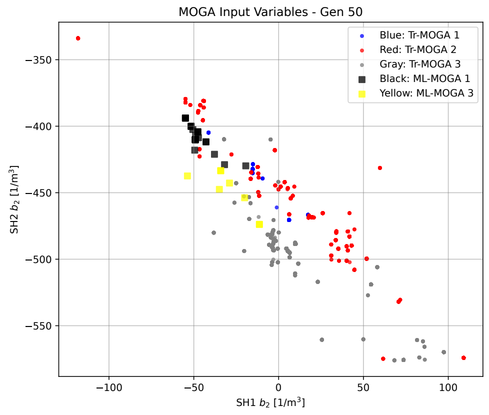

Figure 3: Input space comparison between Tr-MOGA runs and ML-MOGA runs with different random seeds.

To quantify how closely ML-MOGA results approach those of Tr-MOGA, we also ran Tr-MOGA using different random seeds. Red dots show the Pareto front of a Tr-MOGA run with a different MOGA random seed (affects breeding only). Conversely, gray dots show the Pareto front after changing both MOGA random seed and lattice error random seed (affects underlying physics). The effect of changing the MOGA random seed is much smaller than the lattice error random seed, and comparable to discrepancies between ML-MOGA and Tr-MOGA for the same underlying physics. For the Tr-MOGA run with both random seeds changed (gray dots), we re-ran the entire ML pipeline to arrive at corresponding ML-MOGA results (yellow squares). Again, the overlap is excellent in both solution space and input space confirming that the proposed ML pipeline can drastically speed up MOGA without sacrificing fidelity.[3]

We also studied the effect of training data sample size on the accuracy of our NN predictions. Figure 4 shows the RMSE percentage increase for smaller sampling sizes compared to the initially chosen 115 × 115. The RMSEs

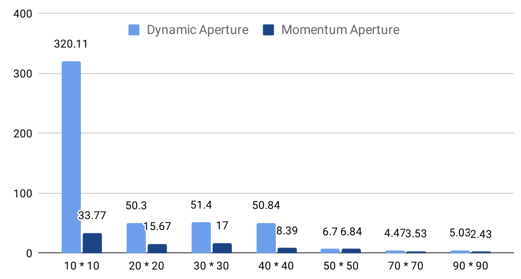

Figure 4: RMSE increase in percent compared to the initial sampling size choice of 115 × 115.

prove to be very robust allowing us to reduce the sampling to as low as 20 × 20 without loss of fidelity. Consequently, we can arrive at similar ML-MOGA results with much smaller sets of training data, effectively reducing tracking effort to 400 samples for ML-MOGA vs. 2.5 × 10^5 for Tr-MOGA.

OPTIMIZATION IN 11 DoF

Unlike the 2-DoF problem, the 11-DoF optimization includes all 9 linear and 2 nonlinear variables. The first step for both Tr-MOGA and ML-MOGA is to find reasonable ranges for the 9 linear DoF. Without narrowing their range, a vast parameter space needs to be searched wherein a large fraction of samples result in unstable solutions. This can significantly impede convergence of the MOGA run, particularly when combining linear and nonlinear lattice optimization. To overcome this problem, we first carry out only linear optimization using MOGA. This is a comparably fast optimization process since DA and MA do not have to be evaluated at this stage. From this linear optimization run, we are able to identify regions of stable solutions and narrow down the parameter range for the 9 linear DoF. Together with the previously introduced nonlinear DoF range, we then

[2] With 5000 children in every generation, convergence in Tr-MOGA thus requires tracking a total of 2.5 × 10^5 samples.

[3] We have investigated retraining of the NNs to improve overlap between ML-MOGA and Tr-MOGA. Although discrepancies can be reduced through retraining effort, we do not consider the additional computational effort justified in light of the stochastic nature of the MOGA breeding process as well as the much larger deviations resulting from variations in the underlying error distribution.

start a full Tr-MOGA or ML-MOGA run. The 9 additional variables make it impossible to randomly generate training data as the required time and computing resources become immense. Conversely, we also find that convergence of conventional Tr-MOGA is very slow, requiring over two weeks of runtime on the aforementioned NERSC cluster for a single lattice error seed. Therefore, with ML we pursue an alternate path: we use just the first 10 generations of Tr-MOGA data (with samples violating any constraints filtered out) as training data to build computational models (NNs for DA and MA) for use in subsequent ML-MOGA runs. Once the ML-MOGA run converges, we again perform one tracking run involving inputs from the final ML-MOGA generation for validation purposes. Finally, by combining this tracked generation with the previous training data we can retrain the NNs allowing us to iterate this ML pipeline until it fully converges.

To properly assess convergence among all MOGA runs, we introduce two Euclidean distance metrics for input and output space, respectively. Equation (1) shows the distance metric for input variables.

$$\delta_i(m) = \frac{1}{n(n-1)} \sum_{j=1}^{n} \sum_{k=1}^{n} \sqrt{\sum_{l=1}^{11} \left(\frac{a_{jl}^{(m)} - a_{kl}^{(m-1)}}{c_l} \right)^2} \quad (1)$$

Here, $a_{jl}^{(m)}$ is input variable l of child j in generation m. Similarly, $a_{kl}^{(m-1)}$ is input variable l of child k from previous generation $m-1$. c_l is the predetermined parameter range of variable l as mentioned above. Equation (2) shows the distance metric for emittance, DA, and MA compared to reference values $\{\varepsilon_0, D_0, M_0\}$, usually chosen as ideal target values.

$$\begin{aligned} \delta_o(m) &= \frac{1}{n}\sqrt{\sum_{j=1}^{n} \left(A_{mj}^2 + B_{mj}^2 + C_{mj}^2\right)}, \quad \text{where} \\ A_{mj} &= \frac{\varepsilon_{mj} - \varepsilon_0}{\varepsilon_0}, \quad B_{mj} = \frac{D_{mj} - D_0}{D_0}, \\ C_{mj} &= \frac{M_{mj} - M_0}{M_0} \end{aligned} \quad (2)$$

Here D_{mj} is the DA of child j in generation m and similarly for MA, M_{mj}, and emittance, ε_{mj}.

Both metrics have been defined as normalized quantities. Convergence in solution space is inferred from vanishing derivative of $\delta_o(m > m_0)$, but is independent of absolute values since that is determined by the choice of reference values. Figure 5 shows evolution of the distance metrics for two Tr-MOGA runs with different random seeds and one ML-MOGA run. In this example, Tr-MOGA run 2 (gray dots) converges at roughly generation 500 as both $\delta_i(m)$ and $\delta_o(m)$ become stable for $m \gtrsim 500$.

ML-MOGA runs complete much faster: 1500 generations can be produced within 48 hrs on the aforementioned NERSC cluster, compared to less than 100 generations with Tr-MOGA within the same time on the same cluster. Figure 6 shows a comparison in solution space between Tr-MOGA and ML-MOGA runs with different random seeds for the 11-DoF problem. Blue dots show Tr-MOGA run 1 at generation 643, while gray dots represent Tr-MOGA 2 (different MOGA random seed and lattice error random seed) at generation 635. Rank-1 children of the validated ML-MOGA results (for a first ML-MOGA iteration) are shown as black squares (same random seeds as Tr-MOGA 1) and yellow squares (same random seeds as Tr-MOGA 2). Note the good agreement for the same underlying physics. Further iterations of the ML-MOGA process are expected to reduce the remaining discrepancies. Note also, the ML-MOGA runs shown here require the tracking equivalent of just 11 generations, including initial training effort.

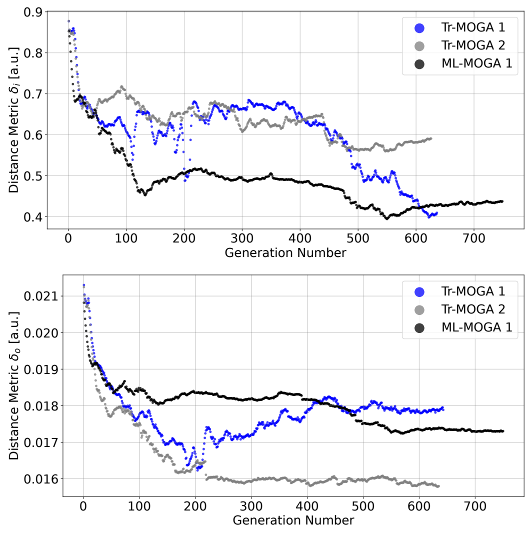

Figure 5: Distance metric for input variables (top) and solution space (bottom) for two Tr-MOGA runs with different random seeds and one ML-MOGA run.

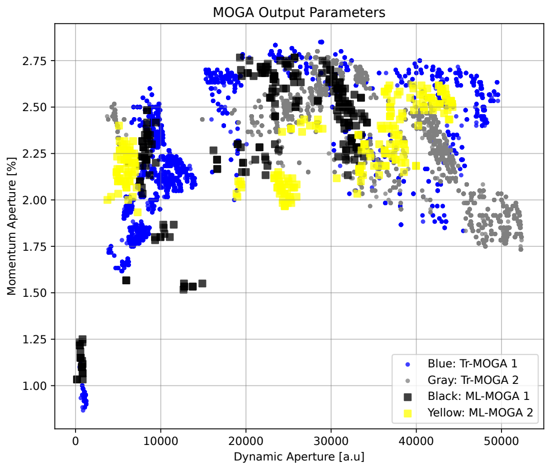

Figure 6: Comparison in reduced solution space (emittance omitted for clarity) between Tr-MOGA and ML-MOGA runs with different random seeds.

REFERENCES

[1] M. Tadahiko and I. Hisao, "MOGA: Multi-objective genetic algorithms", in *Proc. IEEE International Conference on Evolutionary Computation*, 1995, pp. 289–294. doi:10.1109/icec.1995.489161

[2] W. Gao, L. Wang, and W. Li, "Simultaneous optimization of beam emittance and dynamic aperture for electron storage ring using genetic algorithm", in *Phys. Rev. Spec. Top. Accel Beams*, vol. 14, no. 9, p. 094001, Sep. 2011. doi:10.1103/PhysRevSTAB.14.094001

[3] C. Sun, D. S. Robin, H. Nishimura, C. Steier, and W. Wan "Small-emittance and low-beta lattice designs and optimizations", in *Phys. Rev. Spec. Top. Accel Beams*, vol. 15, no. 5, p. 054001, May 2012. doi:10.1103/PhysRevSTAB.15.054001

[4] L. Yang, D. Robin, F. Sannibale, C. Steier, and W. Wan, "Global optimization of an accelerator lattice using multiobjective genetic algorithms", in *Nucl. Instrum. Methods Phys. Res., Sect. A*, vol. 609, no. 1, pp. 50–57, Oct. 2009. doi:10.1016/j.nima.2009.08.027

[5] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II", in *IEEE Trans. Evol. Comput.*, vol. 6, no. 2, pp. 182–197, Aug. 2002. doi:10.1109/4235.996017

[6] M. T. M. Emmerich and A. H. Deutz, "A tutorial on multiobjective optimization: fundamentals and evolutionary methods", in *Natural computing*, vol. 17, no. 3, pp. 585–609, Sep. 2018. doi:10.1007/s11047-018-9685-y

[7] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning", *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015. doi:10.1038/nature14539

[8] Y. Li, W. Cheng, L. H. Yu, and R. Rainer, "Genetic algorithm enhanced by machine learning in dynamic aperture optimization", in *Phys. Rev. Spec. Top. Accel Beams*, vol. 21, no. 5, p. 054601, May 2018. doi:10.1103/PhysRevAccelBeams.21.054601

[9] M. Kranjčević, B. Riemann, A. Adelmann, and A. Streun, "Multiobjective optimization of the dynamic aperture using surrogate models based on artificial neural networks", in *Phys. Rev. Spec. Top. Accel Beams*, vol. 24, no. 1, p. 014601, Jan. 2021. doi:10.1103/PhysRevAccelBeams.24.014601

[10] J. Wan, P. Chu, and Y. Jiao, "Neural network-based multiobjective optimization algorithm for nonlinear beam dynamics", in *Phys. Rev. Spec. Top. Accel Beams*, vol. 23, no. 8, p. 081601, Aug. 2020. doi:10.1103/PhysRevAccelBeams.23.081601

[11] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines", in *Proc. 27th International Conference on Machine Learning (ICML'10)*, Haifa, Israel, Jan. 2010, pp. 807–814. doi:10.5555/3104322.3104425

[12] C. Sun *et al.*, "Design of the ALS-U Storage Ring Lattice", in *Proc. 8th Int. Particle Accelerator Conf. (IPAC'17)*, Copenhagen, Denmark, May 2017, pp. 2827–2829. doi:10.18429/JACoW-IPAC2017-WEPAB105