ISBN: 978-3-95450-184-7 doi:10.18429/JACoW-IPAC2018-THPML116

AUTOTUNER: A GENERAL GRAPHICAL USER INTERFACE FOR AUTOMATED TUNING *

Xiaobiao Huang[1†], Tong Zhang[1,2]
[1]SLAC National Accelerator Laboratory, 2575 Sand Hill Road, Menlo Park, CA, USA 94025
[2]University of Science and Technology of China, 96 Jinzhai Rd, Hefei, Anhui, China

Abstract

AutoTuner is a general graphical user interface (GUI) that we developed for automated tuning or online optimization. The GUI provides a convenient interface to select tuning knobs, objectives, and optimization algorithms and to change the tuning control parameters. Tuning setup can be created and saved for reuse. The progress of the tuning processing is plotted in real time. The tuning process can be paused, aborted, or resumed. We have tested the program for real-life accelerator tuning problems.

INTRODUCTION

Tuning has been an indispensable approach for improving machine performance since the beginning of particle accelerators. Traditionally tuning is done manually - an operator turns a control knob at a time while watching the performance of the machine and using the performance response to guide the tuning direction. Manual tuning is usually slow and is not effective for complex problems where multiple inter-dependent knobs need to be tuned.

Modern accelerators are controlled through computers. This has made automated tuning not only possible, but also inevitable. Automated tuning is similar to manual tuning, except here a computer takes the role of the operator. Automated tuning has many advantages over manual tuning. The overhead of interfacing between the operator and the control computers in the knob changing action is eliminated, which saves time and reduces the risk of human errors. Multiple knobs can now be turned simultaneously in arbitrary combinations, which is useful for large scale problems. Most importantly, the computer can more efficiently and more accurately process the data accumulated in the tuning process and can potentially combine the data with some internal models in real time to make better decisions in the process which lead to the better solutions in less time.

Automated tuning is basically online optimization of performance functions. The algorithms used in online optimization are critical for its efficiency and its ability to find the real optimum. Iterative parameter scans and the downhill simplex method were tested in Ref. [1]. The robust conjugate direction search (RCDS) method was proposed and tested in Ref. [2]. Because RCDS is aware of the noise in online function evaluation and takes measures to reduce sensitivity to noise, it is very suitable for online applications. There have been successful applications of RCDS at many labs around the world [3–9]. Clearly, online optimization has become a popular trend and has great potential for future applications.

Today the application of online optimization is typically done by running an optimization script. This requires the user to be familiar with programming and the code setup for the particular optimization problem. Therefore, the use of online optimization is limited to experts and there is a certain learning curve and training requirement for accelerator operators . A graphical interface (GUI) would make using the online optimization tools substantially easier.

In Ref. [10] we reported on such a GUI called AutoTuner. Since then we have improved the design and the functionality of the program. In this report we describe the framework of the GUI and its present status, including the experimental tests we conducted on the SPEAR3 storage ring.

PROGRAM INTERFACE

The main purpose of the AutoTuner is to make online optimization an accessible tool in routine operation of accelerations. We try to make the interface easy to use and complete with the necessary functionality by the design of the program framework.

The main GUI panel is as shown in Figure 1. There are five areas on the panel. The upper left area is for optimization problem management. A problem definition includes the knobs, the target (objective function), the optimization type, the solver, and the working directory. The information is saved in internal data structures and can be saved and re-used. The optimization type can be minimization, optimization, or setting the target to a specific value. The solver refers to the optimization algorithm to be used. Presently we have implemented the RCDS method and the downhill simplex (Nelder-Mead) method. More algorithms can be added in the future. Problem descriptions or instructions can be shown in a text box at the bottom of this area.

The middle left area is for knob management. An optimization problem can have multiple knobs. The potential tuning knobs are grouped by categories. Knobs can be added to the categories using the GUI. A knob can be a process variable (PV) or a Matlab function. Each knob has a range, a unit, and a designated pause time after the knob value is changed before the performance evaluation begins. All of these fields can be changed with the GUI and can be saved to the problem definition. Internally the range of each knob is mapped to [0, 1].

The bottom left area is for the definition of the objective (i.e., target) function(s). Potentially multiple objective func-

* Work supported by U.S. Department of Energy, Office of Science, Office of Basic Energy Sciences, under Contract No. DE-AC02-76SF00515
† xiahuang@slac.stanford.edu

ISBN: 978-3-95450-184-7 doi:10.18429/JACoW-IPAC2018-THPML116

tions can be used for one optimization session, although we have only implemented single-objective algorithms so far. The objective function(s) can be chosen from a list of candidates. A target function is a measure of the machine performance. It can be a PV or a Matlab function. The noise level for a single evaluation is an input field for the target. Multiple evaluations of the target function can be performed to reduce the noise level. The pause time between consecutive evaluation is also an input field.

Controls for the optimization algorithm are provided in the bottom right area. Before online optimization begins, the problem setting can be checked in case the problem has not been previously tested. The "Start" button launches a new optimization run for the current problem. Unlike launching optimization with a script, in which case no intervention is possible after it is started, the GUI allows interruption. The optimization algorithm can be put on hold with the "Pause" button and resumed later. The "Stop" button terminate the optimization run. If improvement is made during the optimization run, the machine setting can be set to the best solution using the "To Best" button. If that is not desired, the original machine setting can be restored. Optimization history data will be saved to data files in the working directory.

The top right area of the GUI shows the optimization progress in two plots. The top plot shows the history of the objective function values. The bottom plot shows the history of the values of the normalized knob values. Both plots are updated in real time as each data point is evaluated. The real-time graphic display of the history of the optimization run is very helpful for the user to make informed decisions.

CODE AND DATA MANAGEMENT

The AutoTuner GUI is intended to be an open and expandable program. The code and data management are structured in a way to facilitate users to extend its functionality. The GUI essentially provides an interface for optimization problem definition, optimization algorithm control, and visualization of optimization data.

The program interface described in the previous section provides input fields to define optimization problems. The problem definition is saved as a data object internally and can be saved to a file to be re-used. Problems can also be defined by using Matlab scripts to construct the data objects with proper fields. A list of pre-defined problems are registered through a script, which is loaded to the GUI for user selection. Problem definition data are saved in a sub-directory along with the main source code.

An algorithm is an extension of the program; all files related to the algorithm are located in one sub-directory. Each optimization algorithm uses the same problem definition data. The algorithm functions calls a standard objective function for function evaluation. This function convert the normalized decision variables to actual knob values. It then calls the problem specific objective function that changes the machine setting and measures machine performances. The standard objective function updates the data plots on the GUI.

The optimization data are saved in the working directory, which is typically different from the GUI code directory. When an optimization run is started, a new data record object is created. Each data point is added to the data object as the optimization algorithm proceeds. The data object is saved with the start and end time stamp when the current run is stopped. The saved data contain all relevant information of the problem definition and machine setting which can be used in post processing.

ONLINE TESTS

AutoTuner support the simulation mode for off-line tests, which could be useful for the development stage of code expansion. This mode can potentially be used to optimize simulation problems using the same problem definition framework. As we plan to integrate many optimization algorithms into AutoTuner, it could become a general optimization software package. In this report we only discuss its online optimization capabilities.

The new AutoTuner has been experimentally tested on the SPEAR3 storage ring. Figure 1 shows the interface and test results when it was applied to the kicker bump matching problem. This problem has been previously reported in Ref. [2,11]. The goal of the problem is minimize the residual oscillation of the stored beam after the three injection kickers are fired. The knobs are the voltages, pulse widths, and pulse delays of two of the kickers, and two skew quadrupole magnets between the kickers. The skew quadrupoles are used to minimize oscillation in the vertical plane. The residual oscillation is measured by a turn-by-turn BPM and the objective function is the $f = \sigma_x + 3\sigma_y$, where $\sigma_{x,y}$ are rms of beam orbit in the first 256 turns.

Data shown in Figure 1 were for the RCDS algorithm. No initial conjugate direction set was provided to the algorithm. Therefore, the knobs were initially optimized sequentially. The downhill simplex method was also tested for this problem. The results are shown in Figure 2. The simplex method turns the knobs simultaneously. Both algorithms found the optimal setting with about 50 function evaluations.

We also tested the GUI with other online optimization problems, including injector beam intensity tuning with linac and linac-to-Booster (LTB) knobs and SPEAR3 coupling minimization. The Booster beam intensity at extraction was used as the objective function for the injector tuning. The K2 klystron phase and an LTB steering magnet (B3trim) were used as knobs. These knobs are frequently tuned by operators in operation. Coupling minimization has been used as a test case for the RCDS method [2]. In this test the objective was to maximize beam loss over a 6-second period with 500 mA beam current because small coupling leads to high loss rate for a Touschek lifetime dominated beam. The knobs are 13 skew quadrupoles.

ISBN: 978-3-95450-184-7 doi:10.18429/JACoW-IPAC2018-THPML116

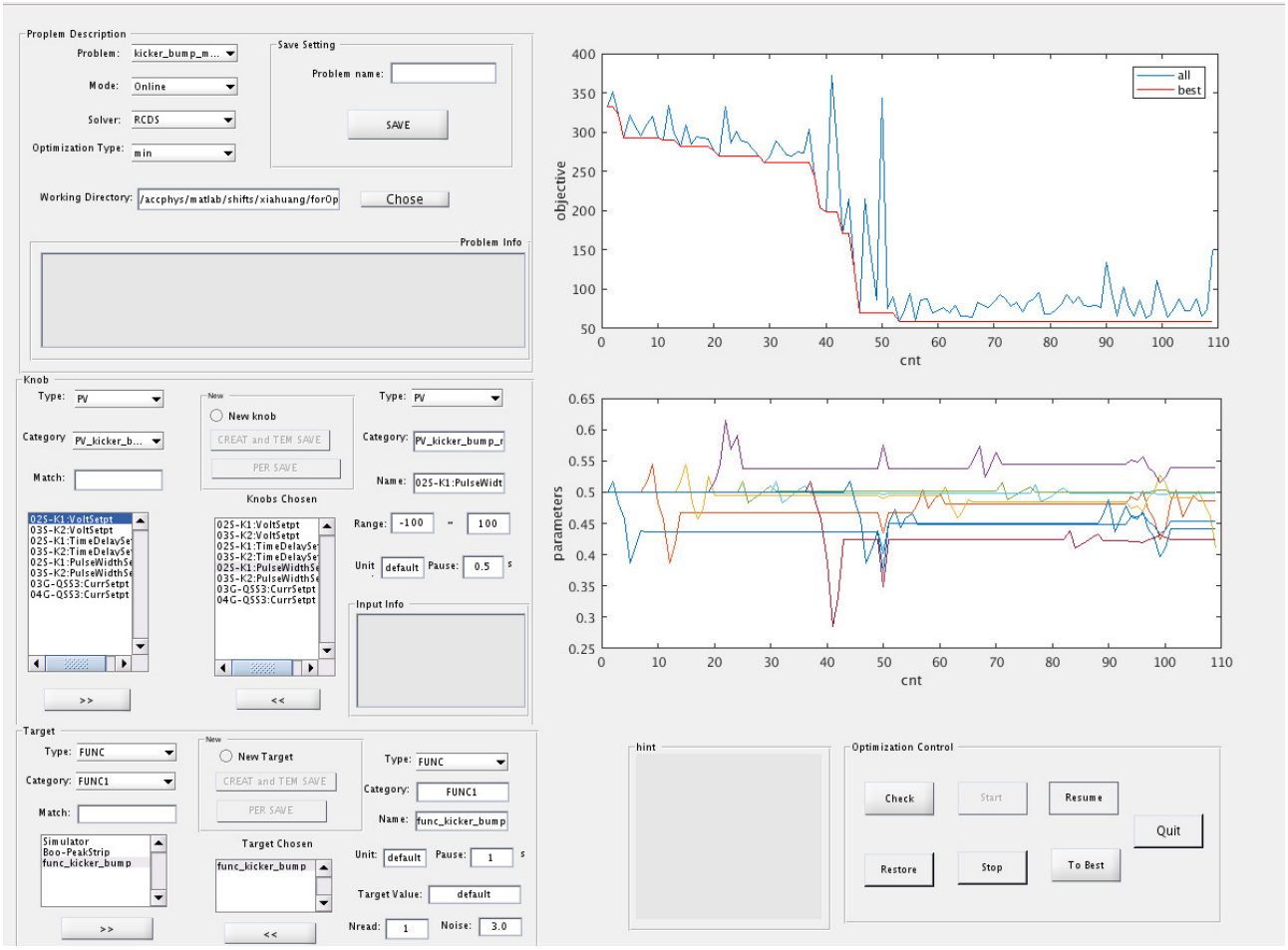

Figure 1: The AutoTuner GUI as it was used in the SPEAR3 kicker bump matching test experiment.

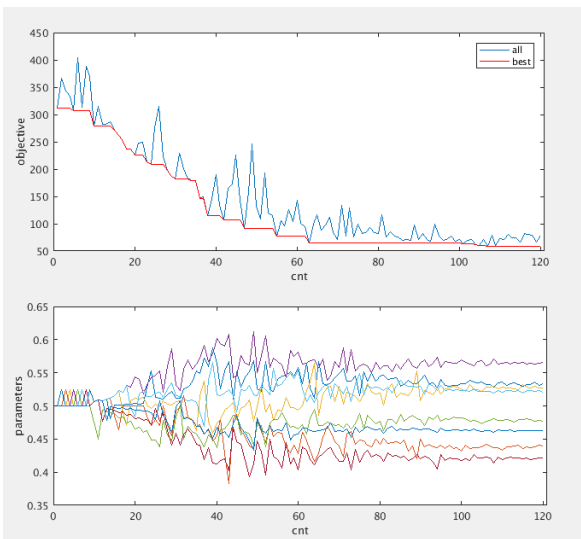

Figure 2: The history of knob variables and the objective function in kicker bump matching optimization with the Nelder-Mead simplex method.

CONCLUSION

We expanded the functionality of the AutoTuner program with the aim of making it a general online optimization tool. The user interface enables users to define and manage optimization problems more conveniently. The RCDS and the Nelder-Mead simplex optimization algorithms have been implemented and more algorithms can be incorporated. The GUI plots the history of the optimization process in real time and allows intervention during optimization. A clear scheme of program and user data management facilitates further expansion of the program and post processing of experimental data.

The GUI has been experimentally tested on the SPEAR3 storage ring with multiple test problems.

REFERENCES

[1] L. Emery, M. Borland, H. Shang, Proceedings of PAC03, Portland, Oregon, USA 2330-2332 (2003)

[2] X. Huang, J. Corbett, J. Safranek, J. Wu, Nucl. Instrum. Meths. A 726, 77 (2013)

[3] X. Huang, J. Safranek, Phys. Rev. ST Accel. Beams 18, 084001 (2015).

[4] H.-F. Ji, et al, Chinese Physics C 39, 127006 (2015)

[5] S. Liuzzo, et al, Proceedings of IPAC2016, Busan, Korea, 3420-3422 (2016)

[6] I. Martin, et al, Proceedings of IPAC2016, Busan, Korea, 3381-3383 (2016)

ISBN: 978-3-95450-184-7 doi:10.18429/JACoW-IPAC2018-THPML116

[7] G. Wang, et al, Proceedings of IPAC2017, Copenhagen, Denmark, 4683-4685 (2017)

[8] T. Pulampong, et al, Proceedings of IPAC2017, Copenhagen, Denmark, 4086-4088 (2017)

[9] W. F. Bergan, et al, Proceedings of IPAC2017, Copenhagen, Denmark, 2418-2420 (2017)

[10] X. Huang, Proceedings of NAPAC'16, Chicago, Ill, USA, 1287-1291 (2016)

[11] A. Sheinker, X. Huang, J. Wu, IEEE Trans. control systems, 26, 1, 336-343 (2018)