# ACCELERATOR OPTIMIZATION FOR LARGE PARAMETER NUMBERS EMBEDDED IN THE NATIVE FAIR CONTROL SYSTEM ENVIRONMENT

W. Geithner*[1], M. Boschert[2],
Z. Andelkovic[1], F. Herfurth[1], N. Kehl[1], J. Koedel[3],
A. Müller[2], N. Stallkamp[1], S. Trotsenko[1], D. Zisis[3]
[1]GSI Helmholtzentrum für Schwerionenforschung, Darmstadt, Germany
[2]Hochschule Darmstadt, Fachbereich Informatik, Darmstadt, Germany
[3]Technische Universität Darmstadt, Fachbereich Physik, Darmstadt, Germany

*Abstract*

We report on the latest developments of our end-user application "Device-Automator" for use in the main control room of GSI/FAIR. Recent progress includes a Bayesian optimizer engine ported from Python to Java in addition to the genetic algorithm engine we applied in earlier work. The engines were tested on the HITRAP beamline with up to 58 control parameters and the CRYRING@ESR injection system.

## INTRODUCTION

Particle accelerator operation is subject to continuous improvement in efficiency and effectiveness, with the goal of providing beam according to experimental requirements. While *technical* performance is advanced through new physical and technical concepts, machine *operation* must be improved by optimizing the beam provisioning processes.

Accelerator models are one important tool for minimizing setup times if these models are well-developed and matured. However, situations exist where well-established models are not available (new machines, parameter drifts over time, complex and hard-to-model subsystems. etc.) and "manual" tuning of the numerous accelerator parameters such as beam orbit, tune, transport beamline elements, injection systems, etc. – becomes one key aspect of machine operation binding significant personnel resources. Software tools assisting operators are therefore highly desirable. Ideally, such tools should (a) reduce time for machine setup and tuning, (b) reduce operator workload, and (c) achieve beam performance at least comparable to experienced human operators. The implementation of such tools has been facilitated by recent advances in computer science making available ready-to-use optimization libraries, and tailored implementations have been used successfully for optimizing particle beams and related sub-systems; see [1] for an overview.

This has to be framed within GSI Helmholtzentrum für Schwerionenforschung (GSI) and its extension, the Facility for Antiproton and Ion Research (FAIR) [2], currently under construction [3]. While GSI currently operates "only" 5 machines, FAIR will comprise several ion sources, 10 accelerators or decelerators, 2 fragment separators, and numerous experimental sites, operated as injectors to each other or in parallel. Typical experiments last a few days and often require entirely different setups regarding ion species and beam energy. For this future, it can be anticipated that operation will become more challenging and labor-intensive, increasing personnel workload. In addition to that, the FAIR complex must be commissioned before operation which will be a very time-consuming process.

The FAIR Control System (CS) built upon the CERN control system stack, offers two approaches for setting control parameters: device level parameters (magnet currents, voltages, etc.) can be set via the so-called JAPC API [4]. Physics-level parameters (kick angles, tunes, beam energies, etc.) are modeled in the so-called "LSA layer" providing a high-level API [5]. Ideally, applications should integrate as tightly as possible into this given environment.

In the context of this work, CRYRING@ESR and HITRAP are two FAIR-associated machines already in productive or commissioning states [6, 7], are serving as test environments for new concepts such as optimizer tools presented here.

## DEVICE-AUTOMATOR APPLICATION

In the context of GSI/FAIR, a series of machine studies employing software optimizers based on the programming language Python showed very promising results [8–10]. To provide an end-user application, the beam automation tool GeOFF, as well implemented in Python, was migrated from CERN to the FAIR control system stack [11].

The FAIR/GSI control system software stack is implemented in C++ and Java [12]. We therefore develop and maintain an application, named "Device-Automator", designed for use in the main control room by operators and accelerator physicists. Device-Automator incorporates optimization tools natively in Java – FAIR CS' native language – enabling direct integration without additional interfaces bridging Java and Python. One key challenge in this approach is the lack of ready-to-use optimization libraries in Java, which are abundant in Python. To address this, we implemented a Bayesian optimizer, and integrated the Jenetics genetic algorithm library [13] and the BOBYQA optimizer available in the Apache math3 library [14].

The GUI of Device-Automator is implemented in JavaFX 21 [15] following the standard technology stack used for FAIR control room applications. This design choice ensures seamless integration with existing tools and a consistent user experience for personnel already trained using other

* w.geithner@gsi.de

FAIR CS applications. Device-Automator allows loading accelerator parameters from a user-selected "beam production chain" (BPC) – a configuration defining the accelerator chain, parameters, and timing. Users can load "scan-templates" – pre-configured parameter sets for scanning and observing objectives, with each parameter individually configurable: BPC timing, scan range, step size, data processing, etc.. Furthermore, Device-Automator provides run options including scan-process algorithms, beam instrumentation (BI) timing, data aggregation and more.

Device-Automator uses FAIR CS' LSA/JAPC API to send settings to accelerator components and receive diagnostic data. The application is modularized into Java packages: "app" (GUI), "services" (scan logic and optimizer integration), "base" (common functionality), and "db" (database access via direct connection and devauto-rest-api).

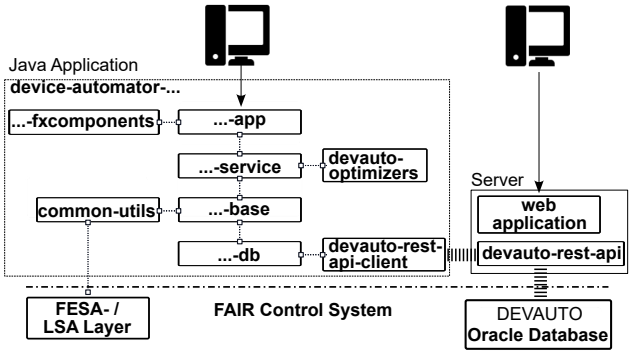


Figure 1: Architecture overview of the device-automator application.

Recently, we enhanced Device-Automator by integrating a database on the FAIR CS Oracle stack for storing configuration parameters, scan templates, and run data for post-run analysis.

For best maintainability, the Device-Automator Java application is structured into 8 software packages separating functionality (see Fig. 1). The database is isolated from the application logic via a REST api, running on a separate server, which also provides a newly built, user-friendly web interface for data access and export.

## DEVICE-AUTOMATOR OPTIMIZERS

Device-Automator provides several user-selectable algorithms: a brute-force/grid-search (constant step width), a genetic algorithm using the Jenetics library [13] (included since the initial version [16]), and self-built implementations of a Bayesian optimizer, a BOBYQA optimizer (Apache commons.math3 [14]), and a random walk optimizer.

### *Experimental Results*

The algorithms were tested at the HITRAP beamline and the CRYRING@ESR injection system – two representative optimization environments: a beamline with many free parameters, and a system with established parameters for a variety of ion species and energies. Hitrap is a low-energy beamline typically operated with its local ion source at repetition rates of approximately 1 Hz with mostly electrostatic elements, while CRYRING@ESR is a low energy ion storage ring and synchrotron with an mix of electrostatic elements in the injection system plus magnetic elements and more complex beam dynamics. Cryring is typically operated at repetition rates of approximately 0.1 Hz depending on deceleration or acceleration cycles.

**Experiments at the HITRAP beamline** The section of the HITRAP beamline utilized for testing connects the local EBIT ion source to the HITRAP cooler trap, comprising electrostatic transport and focusing elements, one dipole magnet, and several diagnostic devices. For our test, an $Ar^{16+}$ ion beam from the local EBIT ion source at an energy of 5 keV/q was used. The goal of the tests was to optimize beam transport efficiency through the beamline by varying the electrostatic elements. A bender magnet in the beamline (see Fig. 2) was operated at fixed current for the given ion species and energy during all the tests.

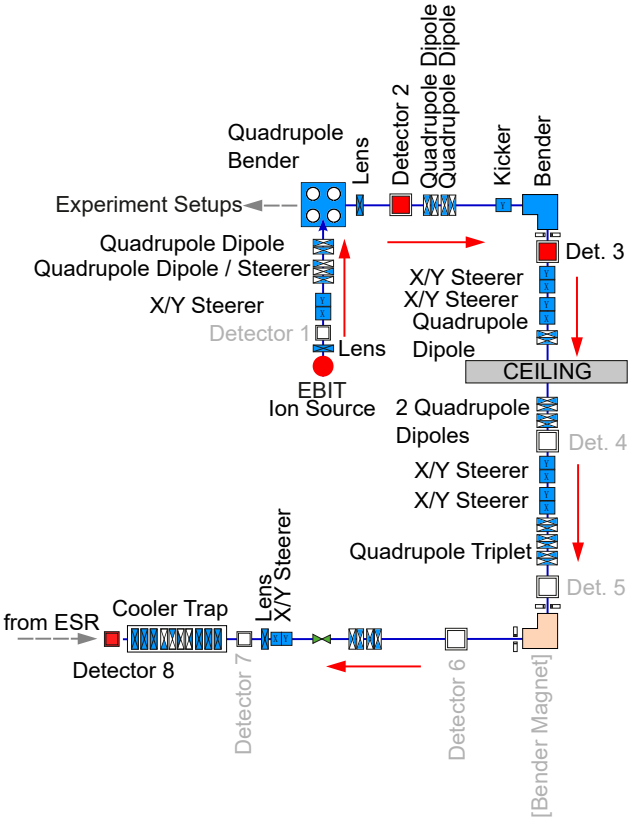


Figure 2: Schematic layout of the HITRAP beamline.

Three sets of tests were performed with 16, 24, and 58 parameters, respectively. The objective was to maximize beam current on the Faraday cups labelled "Detector 2, 3 and 8" in Fig. 2.

Figure 3 shows beam current evolution during Bayesian (16 parameters) and genetic (58 parameters) optimization. We found that the Bayesian optimizer is not well suited for large parameter numbers, as performance degrades significantly beyond ~250 steps. Nevertheless, it repeatedly found

16-parameter settings matching experienced human operators within 100 iterations (~10 min).

Given enough time, the genetic optimizer significantly exceeded human-operator results for 58 parameters, with continued improvements observed over 5000 iterations (~12 hours).

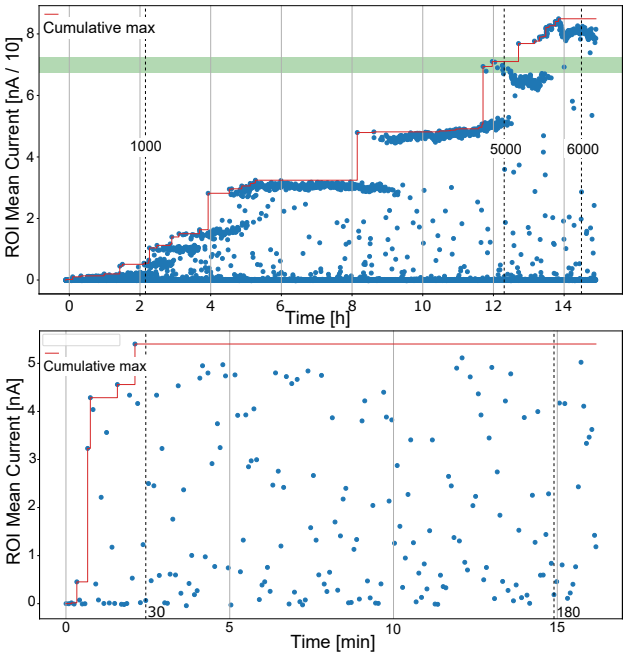


Figure 3: Beam current during genetic optimization of 58 beamline parameters (upper) and Bayesian optimization of 16 beamline parameters (lower). The red line indicates the cumulative maximum - the best value found up to that point. The green bars indicates beam intensities achieved by human operators. For the Bayesian optimization, reference values for the beam current were not available.

**Experiments at the CRYRING@ESR injection system** The CRYRING@ESR injection system transfers ions from a local ion source and RFQ accelerator-injector to the synchrotron ion storage ring. It comprises five electrostatic elements, several focusing magnets, and one dipole – nineteen parameters in total. The intensity of the eventually stored beam was monitored via the count rate of a vertical ionization profile monitor (IPM). The tests were performed using an $D^+$ beam at an injection energy of ≈300 MeV/u, with the objective of maximizing the IPM signal by optimizing the 19 parameters.

Figure 4 shows IPM signal evolution during genetic and BOBYQA optimization. Signal differences between runs are due to varying upstream beamline settings. The genetic optimizer reaches operator-comparable signals within 600 iterations (~3.5 hours); BOBYQA achieves similar values within ~100 iterations (~45 min).

## CONCLUSION

Software optimizers are a valuable tool for heavy-ion accelerator tuning and operation. While proven in machine

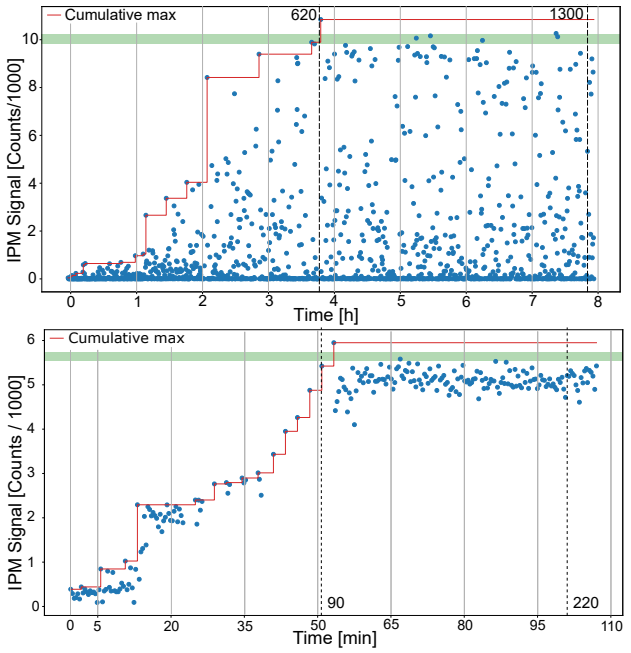


Figure 4: Cryring injection optimization: IPM signal during genetic (upper) and BOBYQA (lower) optimization. The red line indicates the cumulative maximum - the best value found up to that point. Dashed vertical lines with numbers indicate the number of steps. Note that the differences in signal values between runs are due to varying upstream beamline settings, which were not optimized during these tests. The green bars indicates the signal strengths before the optimization.

studies via Python-based approaches, a gap exists in bringing them to daily operations. Device-Automator closes this gap by providing an operations-grade control-room application with well-performing algorithms. Our tests show that optimizers yield operator-comparable results when application scenarios are carefully selected. The genetic algorithm is well suited for unexplored settings and large parameter spaces when sufficient time is available. For narrow parameter ranges with established starting points, the Bayesian and BOBYQA optimizers provide quick tuning support.

## REFERENCES

[1] A. Edelen *et al.*, "Machine learning for design and control of particle accelerators: A look backward and forward", *Annu. Rev. Nucl. Part. Sci.*, vol. 74, no. 1, pp. 557-581, 2024. doi:10.1146/annurev-nucl-121423-100719

[2] G. Rosner, "Future facility: FAIR at GSI", *Nucl. Phys. B, Proc. Suppl.*, vol. 167, pp. 77-81, 2007. doi:10.1016/j.nuclphysbps.2006.12.089

[3] S. Reimann *et al.*, "FAIR Commissioning-Towards First Science", in *Proc. 14th Int. Particle Accelerator Conf. (IPAC2025)*, Taipei, Taiwan, 2025, THBN2, JACoW (2025), doi:10.18429/JACoW-IPAC2025-THBN2

[4] V. Baggiolini *et al.*, "JAPC-the Java API for parameter control", in *Proceedings of ICALEPCS*, Geneva, Switzerland, 2005, pp. 1-3, 2005.

[5] J. Fitzek *et al.*, "Settings Management within the FAIR Control System based on the CERN LSA Framework", in *Proceedings of PCaPAC'10*, Saskatoon, Canada, pp. 41-43, 2010.

[6] M. Lestinsky *et al.* "First Experiments with CRYRINGESR", *Atoms*, vol. 10, no. 4, p. 141, 2022. doi:10.3390/atoms10040141

[7] S. Rausch *et al.*, "Commissioning of the HITRAP Cooling Trap with Offline Ions", Atoms, vol. 10, no. 4, p. 142, 2022. doi:10.3390/atoms10040142

[8] S. Appel *et al.*, "Injection optimization in a heavy-ion synchrotron using genetic algorithms", *Nucl. Instrum. Methods Phys. Res. A*, vol. 852, pp. 73-79, 2017. doi:10.1016/j.nima.2016.11.069

[9] S. Appel *et al.*, "Application of nature-inspired optimization algorithms and machine learning for heavy-ion synchrotrons", *International Journal of Modern Physics A*, vol. 34, no. 36, p. 1942019, 2019. doi:10.1142/S0217751X19420193

[10] S. Appel *et al.*, "Automated optimization of accelerator settings at GSI", in *15th International Particle Accelerator Conference, IPAC2024*, Nashville, Tennessee, USA, 19 May 2024 - 24 May 2024, pp. 882-885, 2024. doi:10.18429/JACOW-IPAC2024-MOPS68

[11] P. Madysa *et al.*, "Geoff: The generic optimization framework & frontend for particle accelerator controls", *SoftwareX*, vol. 32, p. 102335, 2025. doi:10.1016/j.softx.2025.102335

[12] M. Huhmann *et al.*, "The FAIR control system–system architecture and first implementations", in *Proceedings of the ICALEPCS*, San Francisco, CA, USA, 2013, pp. 329-331, 2013.

[13] F. Wilhelmstötter, " Jenetics, a modern genetic algorithm, evolutionary algorithm and genetic programming library for Java", version 7.2.0, 2025. [Online]. Available: https://jenetics.io/. [Accessed: 15-March-2026].

[14] Apache Software Foundation, "Apache Commons Math", version 3.6.1, 2016. [Online]. Available: https://commons.apache.org/proper/commons-math/. [Accessed: 05-December-2025].

[15] OpenJFX Community, "JavaFX open source, next generation client application platform", version 21, 2025. [Online]. Available: https://openjfx.io/. [Accessed: 15-March-2026].

[16] W. Geithner *et al.*, "Genetic Algorithms for Machine Optimization in the FAIR Control System Environment", in *9th International Particle Accelerator Conference (IPAC2018)*, Vancouver, Canada, 2018, pp. 2566-2569, 2018. doi:10.18429/JACoW-IPAC2018-THPML028