AUTOMATIZED OPTIMIZATION OF BEAM LINES USING EVOLUTIONARY ALGORITHMS

S. Appel*, S. Reimann†, V. Chetvertkova, W. Geithner, F. Herfurth, U. Krause,
M. Sapinski, P. Schütt, GSI, Darmstadt, Germany
D. Österle, KIT, Karlsruhe, Germany

Abstract

Due to the massive parallel operation modes at the GSI accelerators, a lot of accelerator setup and re-adjustment has to be made during a beam time. This is typically done manually and is very time-consuming. With the FAIR project the complexity of the facility increases furthermore and for efficiency reasons it is recommended to establish a high level of automation. Modern Accelerator Control Systems allow a fast access to both, accelerator settings and beam diagnostics data. This provides the opportunity together with the fast-switching magnets in GSI-beamlines to implement evolutionary algorithms for automated adjustment. A lightweight python interface to CERN Front-End Software Architecture (FESA) gave the opportunity to try this novel idea, fast and easy at the CRYRING@ESR injector. Furthermore, the python interface facilitates the work flow significantly as the evolutionary algorithms python package DEAP could be used. DEAP has been applied already in external optimization studies with particle tracking codes [1]. The first results and gained experience of an automatized optimization at the CRYRING@ESR injector are presented here.

INTRODUCTION

FAIR – the Facility for Antiproton and Ion Research – will constitute an international center of heavy ion accelerators that will drive forefront heavy ion and antimatter research. The goal of the FAIR facility is to provide antiproton and ion beams of unprecedented intensities as well as qualities. As a special feature, the facility will provide a broad range of high-intensity ion, antiproton and rare-isotope beams parallel to multiple experiments.

The High Energy Beam Transport System of FAIR, with a total length of more than 2350 meters, forms a complex system connecting seven accelerator- and storage rings, the experiment caves, beam dumps, stripping stations, the antiproton target and the Super Fragment Separator. The variety of beams to be transported is considerable, ranging from slow extracted beams with long spills of up to 100 s to short intense bunches with lengths of a few nanoseconds and momentum spreads of up to ±1%. The range of beam intensity covers more than six orders of magnitude [2]. The complexity of the FAIR facility demands a high level of automation for future operation, because otherwise the anticipated manpower requirements for operators would be

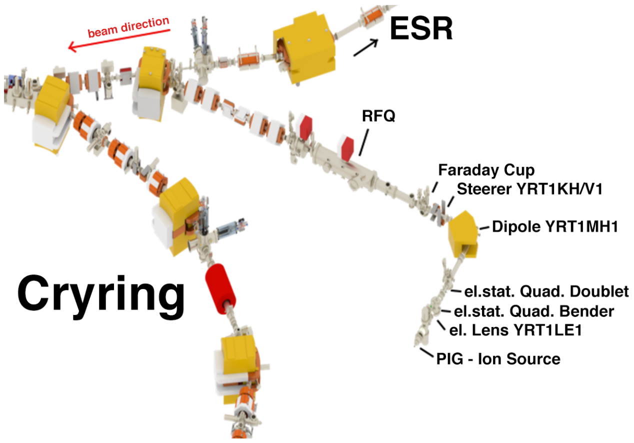

Figure 1: The settings of the steerers and electrostatic quadrupoles between ion source and Farady Cup after the dipol of CRYRING@ESR injector at GSI have been automatized optimized with evolutionary algorithm to maximize the beam transmission.

excessive, as shown in [3]. Modern accelerator control systems allow a fast access to both, accelerator settings and beam diagnostics data. This provides the opportunity to implement algorithms for an automated adjustment. Therefore, the Parameter Evolution Project (PEP) has been launched for automatized online parameter optimization in beam lines. An automatized machine based optimization using genetic algorithms for a storage ring has been already successfully demonstrated experimentally [4].

In the frame of the Swedish in-kind contribution to the FAIR project the storage ring CRYRING@ESR is planned to be used for experiments with low-energy ions and antiprotons. The ring is already installed in the existing GSI target hall and commissioning has started in 2015 [5, 6]. Since CRYRING@ESR has its own local injector it can be used stand-alone for testing novel technical developments like automatized configuration of beam line devices. Figure 1 shows the part of the CRYRING@ESR injector (from ion source to Faraday Cup), which has been used for testing automatized online evolutionary algorithm optimization. A semi-automatized optimization has been already preformed at the CRYRING in Sweden [7].

PYTHON INTERFACE TO FESA AND LSA

Currently, most of the GSI facility is undergoing heavy construction work or large up-grade measure and is therefore not available for beam time until 2018. An exception is the

* s.appel@gsi.de
† s.reimann@gsi.de

CRYRING@ESR and its local injector beam line. Since the CRYRING@ESR has just recently been installed at GSI, the infrastructure follows the guidelines of a modern accelerator control system and allows fast access to both accelerator settings and beam diagnostics data. GSI has selected the Front-End Software Architecture (FESA) to operate accelerator devices and the LHC Software Architecture (LSA) for the new settings management system. FESA is a comprehensive framework used to design, develop and deploy front-end device software. It abstracts from hardware-specific differences by allowing for the creation of device models where the equipment's interface towards the control system is represented as a consistent set of properties. LSA is a settings management framework that supports offline generation of machine settings, sending these settings to all involved devices and programming the schedule of the timing system. Settings calculation is based on a physics model for accelerator optics (twiss, machine layout), parameter space and overall relations between equipment settings and beam parameters. The LSA GUI applications support the operators in setting, optimizing and monitoring the accelerators for parallel beam operation. Both FESA and LSA originated at CERN and are now developed further in collaboration with GSI [8, 9].

Development of new operation techniques to increase the accelerator performance is carried out during dedicated machine development time, often by experienced machine physicists. During machine experiments, the machine physicists usually operate the accelerators in a non-standard way, including non-save operation. Often, the available GUI applications do not support or even prevent such operation modes. In order to ensure success for the machine experiment, new functionality must be added for a short period of use. In the physics community, Python is a widespread computer language as it allows programmers to express ideas in a shorter syntax than C++ or Java. A Python interface to FESA/LSA would significantly facilitate the work flow for the machine physicists. As an example, during machine development time one could easily access measurement data that is not available on the Java GUI in the required form. For the automatized project, the evolutionary algorithms Python package DEAP has been used [10]. DEAP has been applied already in external optimization studies with particle tracking codes. A lightweight Python interface to FESA for the development of novel ideas, fast and easy, is available at GSI. The Python module provides access to FESA devices including reading and writing data. The downside of FESA-Python access is that it is bypassing the settings management of LSA, including the physics model and settings database. A Python bridge to LSA would enable the use of the same framework as in an accelerator simulation code. One would change during the optimization global physics parameters like tune or magnet strength and not, as with the FESA-interface, local magnet currents. Furthermore, the history of settings during optimization would be accessible. A first test bridge to allow for the modification of physics parameters through LSA with Py4J as been implemented recently. Py4J enables Python programs running in a Python interpreter to dynamically access Java objects in a Java Virtual Machine [11].

OPTIMIZATION ALGORITHMS

As a result of the promising GA optimization simulation outcome of the multi-turn injection presented in [1] the Parameter Evolution Project (PEP) has been launched for automatized online parameter optimization in beam lines. The aim of the project is to identify a fast and reliable evolutionary optimization. The genetic algorithms, which is inspired by natural evolution and particle swarm optimization, inspired by movement of organisms in a bird flock or fish school, are widespread evolutionary algorithm. To overcome the evolutionary optimization slowness in the simulation (many different parameter settings need to be evaluated), parallel computing techniques are used. For an automatic optimization of real machine this advantage is not available. On the other hand an automatized configuration would instantly adapt to errors sensitive to the adjustment parameters as well as other technical influences. An automatic optimization of the transmission is maybe feasibly if a pre-conditioned initial generation, a smaller number of individuals and generations can be used to shorten the optimization time. Very important for this kind of an automatic optimization is a fast reaction of the beam line devices as well as a fast and accurate beam diagnostic. For a short optimization within a few minutes a single cycle of setting beam line parameters and reading out detector values should not be longer than two seconds.

Genetic Algorithms

Genetic algorithms (GA) search for solutions using techniques such as *selection*, *mutation* and *crossover*. By employing a wide range of different algorithms, GA are very flexible and can be adapted to a large range of different problems.

In GA terminology, a solution vector is called an *individual* and represents a set of variables; one variable is a *gene*. A group of individuals form a *population*, the following child populations are counted in *generations*. The first population is created randomly. The crossover operator exchanges variables between two individuals - the parents - to discover with their offspring promising areas in the solution space (*exploration*). For the optimization within a promising area, the mutation operator changes randomly the characteristics of individuals on the gene level (*exploitation*). Reproduction of individuals for the next generation involves selection. The *fitness* of an individual reflects how well an individual is adapted to the optimization problem and determines the probability of its survival for the next generation. The fitness is evaluated by an objective function, by a simulation code or by a real running system. During the single-objective optimization the most promising individuals are chosen to create the next generation. By allowing individuals with poor fitness to take part in the creation process the population is prevented to be dominated by a single individual. The most popular techniques are proportional selection, ranking and tournament selection [12–14].

A careful choice of the algorithm and operator is necessary to get the best performance of GA algorithms. The optimal choice of the offspring production probability through crossover or mutation is important for a proper balance between exploration and exploitation. The mutation operators are mostly used to explore, which is preferred at beginning of the search process. On the other hand, at the end of search process more exploitation through the crossover operators is needed to ensure convergence of the population. According to these facts, an incorrect production probability can lead to local optimum convergence.

Particle Swarm Optimization

The initial inspiration for the Particle Swarm optimization (PSO) came form the "graceful but unpredictable choreography of a bird flock". The key to the swarm success liens in social influence and learning. Each individual's behavior is influenced by its own personal experience and the social standard [14]. Within a swarm, each individual refers to a point in the variable space. It is updated by adding a velocity depending on the personal experience and the socially swarm influenced. The "nostalgia" in the individual tends to return to a place it encountered in the past that best fulfilled the objectives reflected by the personal best *pbest*. Simultaneous, the individuals seek to attain publicized knowledge or social norms, reflected by the best position ever for the entire swarm *gbest*. The movements of the swarm a guide by improved positions, which are updated during the optimization. Including in addition stochastic elements in the algorithm allows to search widely and hopefully finding a satisfactory solution.

SIMULATION

The CRYRING@ESR injector model has been implemented in the particle tracking code pyORBIT - the python implementation of ORBIT (Objective Ring Beam Injection and Tracking) [15]. For the GA optimization the Distributed Evolutionary Algorithms in Python (DEAP) together with pyORBIT has been used. DEAP includes evolution strategies, multi-objective optimization, and allows the development of new genetic algorithms. DEAP decouples the GA operators like crossover from the evolutionary algorithms, which allows for example to easily exchange the selection operator and leave the remaining algorithm unchanged. The implantation of PSO algorithms is also easy possible. Simulations indicated the $(\mu + \lambda)$-evolutionary algorithm from the DEAP python package would be the perfect candidate for a rapid automatic optimization. In the $(\mu + \lambda)$-algorithms as first step the individual fitness of μ-individuals are evaluated. μ is the population size and λ the offspring's size. Secondly, the evolutionary loop begins by producing $\lambda < \mu$-offspring's from the population through crossover and mutation. The offspring's are then evaluated and the next generations population is selected from both the offspring's and the current population. Finally, when a given number of generations has been evaluated, the algorithm returns the final population including the best solution [10]. In simulation the $(\mu + \lambda)$-algorithm could sufficient optimize the transmission of a beam line with four quadrupoles after 10 generations using a small population of 100 individuals and offspring size of 50, presented in Figure 2. Assuming a single beam line cycle is two seconds long, an automatic optimization would last 15 minutes.

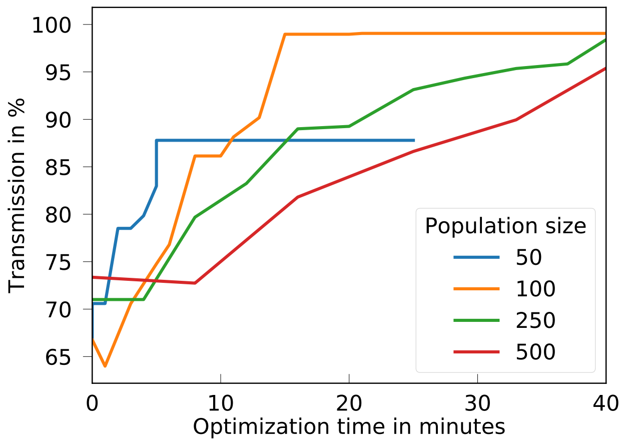

Figure 2: Simulated transmission evolution with the $(\mu + \lambda)$-algorithm for different population sizes for a beam line with four quadrupoles. It has been assumed a single beam line cycle is two seconds long.

EXPERIMENT

The aim of the automatized optimization was to maximize the beam transmission through the beam line using the python package DEAP (e.g. minimization of particle loss along beam line). In the time of machine experiment of one week the require short python syntax has be implemented and improved by few machine physicist without support by the FESA/LSA development team. The DEAP genetic algorithm has altered parameters on which the beam transmission depends to optimize the transmission. The algorithm allows independent variation of the steerer strengths and electrostatic quadrupoles voltages, in total nine different parameters. The 90°-Dipole shown in Figure 1 has been excluded from the genetic algorithm optimization due to is low transmission influence. The beam current has been measured at the current transformer behind the ion source and the Faraday cup after the dipole. Unfortunately, due to lack of time the current transformer has not been calibrated. Still without calibration the measurements from the current transformer could be used as reference in the transmission optimization process. Because of the slow response of the electrostatic devices, a holding period of five seconds before the readout of the beam diagnostic devices has been included. The

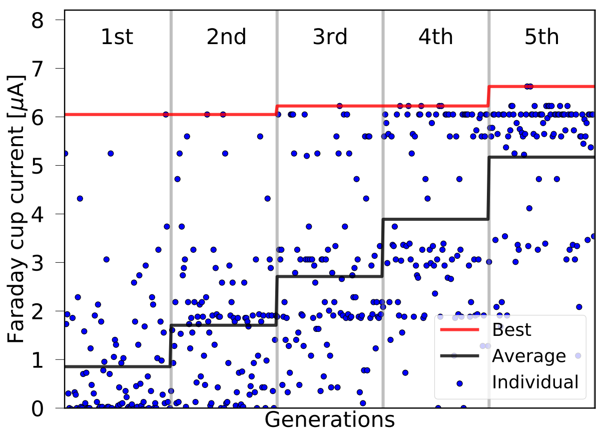

Figure 3: Evolution of the population fitness represented through beam current along generations. As the next population is selected from both the offspring's and current population the number of fitter individuals grows with generations.

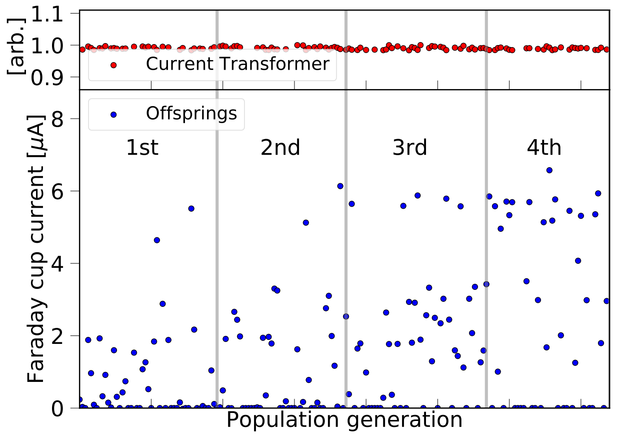

Figure 4: The offspring's fitness represented through beam current along generations. Only the best offspring's replace parents in the next population. The measurements from the uncalibrated current transformer are in arbitrary units, still show the low ion source fluctuations.

slow response of the CRYRING@ESR injector electrostatic quadrupoles of a few seconds is a disadvantage for testing evolutionary algorithm optimization. An enhancement of electrostatic devices response is maybe possible in the future. The result of the first successful evolutionary algorithm's optimization performed at GSI is presented in Figures 3 and 4. The population evolution has been limited to five generation in order not to exceed an optimization time of 30 minutes. Fortunately, during the optimization beam current fluctuation from the CRYRING@ESR source has been low. Even in the first generation a similar transmission as with a manual optimization could be reached, since the parameter space of the first generation has been limited to ±10% of the known optimal settings. As the next population is selected from both the offspring's and current population the number of fitter individuals grows with generations. Nevertheless, the generated and evaluated offspring covers a large parameter space indicated through different beam currents.

CONCLUSION AND OUTLOOK

As a result of the promising simulation outcome of optimizing the multi-turn injection as well as beam lines, the PEP Project has been launched. The first automatic PEP version at the CRYRING@ESR injector has been implemented and tested. A good transmission could be reached in half an hour of optimization time. Still, the PEP Project is at its beginning and many improvements as well as detailed studies have to be made. The influence of population, generation size, crossover and mutation should be studied as well as other genetic algorithms, particle swarm algorithms or machine learning algorithms should be tried. Before the parameter space can be expanded, some trigger must be included like 'measurement failed and has to be repeated' and 'Set value of devices have been reached'. Crucial for the transmission through the RFQ is an optimization of beam size as well as position and must therefore be included. For the GSI beam time in 2018 it is planned to test PEP at the transfer channel to SIS18 for optimizing the injection. A python bridge to LSA would simplify the development of new operation techniques. Therefore first Java-python bridge to LSA with Py4J as been implement recently.

REFERENCES

[1] S. Appel, O. Boine-Frankenheim, F. Petrov, Injection optimization in a heavy-ion synchrotron using genetic algorithms, Nucl. Instrum. Methods A, 852 (2017) pp. 73–79.

[2] S. Ratschow, F. Hagenbuck, P. J. Spiller, The High Energy Beam Transport System for FAIR in Proceedings of EPAC08 (2008) pp. 3608–3610.

[3] S. Reimann, M. Sapinski, P. Schütt, M. Vossberg, Building an Operation Team for FAIR Nearly from Scratch, presentation at WAO10 (2016).

[4] K. Tian, J. Safranek, and Y. Yan, Machine based optimization using genetic algorithms in a storage ring, Phys. Rev. ST Accel. Beams 17, 2 (2014).

[5] W. Geithner, Z. Andelkovic, D. Beck, H. Bräuning, A. Bräuning-Demian, H. Danared, C. Dimopoulou, M. Engström, S. Fedotova, O. Gorda, F. Herfurth, R. Hess, A. Källberg, C. Kleffner, N. Kotovskiy, I. Kraus, M. Lestinsky, S. Litvinov, F. Nolden, A. Reiter, T. Sieber, M. Steck, and G. Vorobyev, Status and outlook of the CRYRING@ESR project, Hyperfine Interact. 238, (2017).

[6] O. Gorda, A. Braeuning-Demian, H. Danared, A. Dolinskii, M. Engstroem, F. Herfurth, A. Kaellberg, C.-M. Kleffner, M. Lestinsky, A. Simonsson, J. Sjoeholm, and G. Vorobjev, Ion-optical design of CRYRING@ESR, Phys. Scr. T 166, 14043 (2015).

[7] A. Källberg, A. Simonsson, "Beam Steering with Image Processing in the CRYRING Injection Beamline", in *Proceedings of DIPAC99*, Chester, UK, 1999, pp. 118–119.

[8] R. Huhmann, C. B. Ralph, D. H. Beck, J. Fitzek, L. Hechler, U. Krause, and M. Thieme, "The Fair Control System – System Architecture and First Implementations", in *Proc. ICALEPCS2013*, San Francisco, USA, 2013, pp. 328–331.

[9] J. Fitzek, H. Hüther, R. Müller, and A. Schaller, "First Production Use of the New Settings Management System for FAIR", in *Proc. ICALEPCS2017*, Barcelona, Spain, 2017, this conference, THPHA062

[10] F. Fortin, D. Rainville, DEAP: Evolutionary algorithms made easy, J. Mach. Learn. Res. 13 (2012) 2171–2175, https://github.com/DEAP/deap

[11] Py4J, https://www.py4j.org/

[12] A. Konak, D.W. Coit, A.E. Smith, Multi-objective optimization using genetic algorithms: A tutorial, Reliab. Eng. Syst. Saf. 91 (9) (2006) 992–1007. doi:10.1016/j.ress.2005.11.018.

[13] A. Hofler, B.Terzić, M. Kramer, A. Zvezdin, V. Morozov, Y. Roblin, F. Lin, C. Jarvis, Innovative applications of genetic algorithms to problems in accelerator physics, Phys. Rev. ST Accel. Beams 16 (1) (2013) 010101. doi:10.1103/PhysRevSTAB.16.010101.

[14] X. Pang, L. Rybarcyk, Multi-objective particle swarm and genetic algorithm for the optimization of the LANSCE linac operation, Nucl. Instrum. Methods A, 741 (2013), pp. 124–129

[15] A. Shishlo, S. Cousineau, J. Holmes, T. Gorlov, The Particle Accelerator Simulation Code PyORBIT, Procedia Comput. Sci. 51 (2015) 1272–1281. doi:10.1016/j.procs.2015.05.312, https://github.com/PyORBIT-Collaboration