AN ONLINE MULTI-OBJECTIVE OPTIMISATION PACKAGE

I.P.S. Martin, M. Apollonio, R. Bartolini[1], M. Furseman, Diamond Light Source, Oxfordshire, U.K.
D. Obee, Durham University, U.K.
G. Bird, University of Oxford, U.K.
[1]also at John Adams Institute, University of Oxford, U.K.

Abstract

The overall performance of an electron storage ring is critically dependant on a large number of variables. It can be characterised in many ways, such as by lifetime, injection efficiency, beam stability and so on. It is frequently the case however that improving one parameter comes at the cost of harming another. Equally, given the large number of variables involved in optimising the ring performance, the true, global optimum solution may be difficult to identify using simple parameter scans. In order to address this problem, a flexible optimisation tool has been developed. This tool is capable of optimising several parameters at once and can cope with an arbitrary number of input variables (individuals or families). The tool is designed to be robust to measurement noise, and has been applied to a number of different optimisation problems. This paper presents an overview of the package, as well as the results of the first tests.

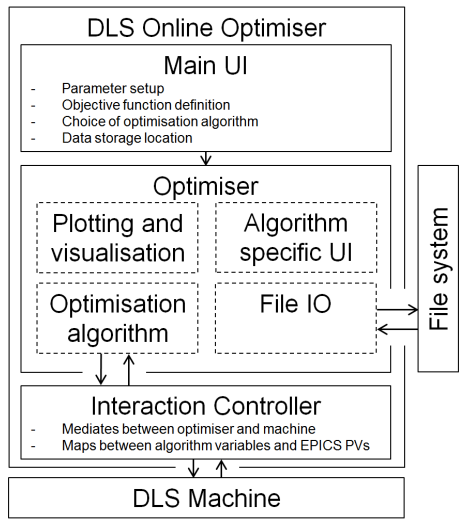

Figure 1: Structure of online optimisation package.

INTRODUCTION

Optimising an electron storage ring performance can face a number of potential difficulties. The impact of varying parameters can be complex to predict due to the non-linear nature of the system and the large number of variables involved. The optimisation function may suffer from dis-continuities or local minima that can prevent traditional gradient-based searches from succeeding. Quantifying the performance can also be imprecise due to the presence of noise or drift in the objective that is to be optimised. It can also be subjective, as frequently the improvement of one parameter can lead to the degradation of another.

At Diamond, the primary optimisation method is to make use of 1D or 2D parameter scans. This can be very effective, but relies on starting close to the optimal position, is limited by the number of variables that can be scanned at one time and can be inefficient. For more complex problems, the Robust Conjugate Direction Search algorithm is used [1]. This has been demonstrated to converge quickly, copes well with noisy data and can handle an arbitrary number of input variables, but is limited to optimising a single objective.

In order to address these issues, Diamond has implemented an online tool that uses modern multi-objective, population-based optimisation techniques. The tool has been structured with future extensions in mind, so that additional optimisation algorithms can be included as modules. It can be applied to single or multi-objective problems, is flexible in the definition of input variables, and has been demonstrated to be robust against measurement noise.

MULTI-OBJECTIVE OPTIMISATION

Pareto Front and Solution Ranking

When considering two or more objectives, the optimum solution can vary depending upon the circumstances. For instance, it may be that for one situation the lifetime may be the biggest concern, but for another the injection efficiency must be maximised, even though it results in a lower lifetime.

Population-based searches address this by maintaining a distribution of solutions. These are sorted into 'non-dominated fronts', for which no member is worse than another in more than one objective simultaneously. The fronts can be further sorted according to how diverse the solutions are, as this helps to identify the true global optimum for each objective. The goal of the optimisation algorithm is then to move the population towards the so-called Pareto-optimal front, after which no further improvement is possible.

Genetic Algorithm

Genetic algorithms (GA) lend themselves well to the topic of multi-objective optimisation. For the package described here, the NSGA-II genetic algorithm has been used [2]. This is a computationally-efficient algorithm, in which an initial random population evolves towards the optimum through the process of selection, cross-over and mutation. The algorithm is elitist, ensuring good solutions are prevented from being lost from the population and helping to speed up convergence. In this implementation, it is also possible to include the initial machine state in the starting population.

ISBN 978-3-95450-182-3

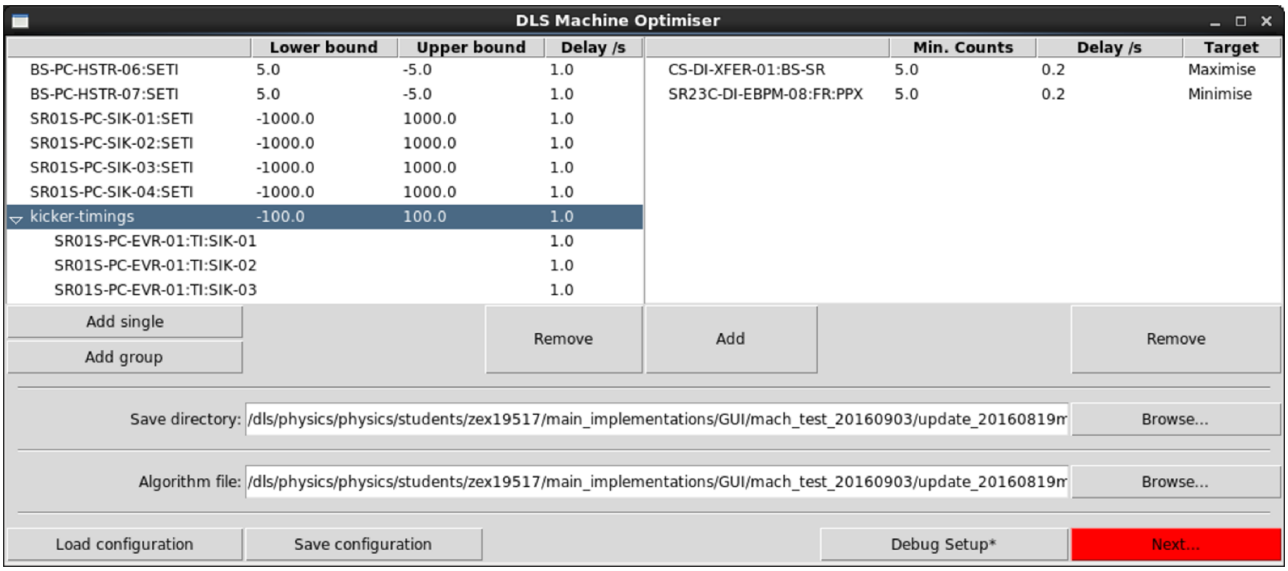

Figure 2: Screenshot of the Main UI for the optimiser. The top-left panel controls the parameters to be varied, the top-right panel controls the objective functions, and the lower panels are used to define the paths.

Simulated Annealing

Simulated annealing (SA) algorithms are well-suited to cases that consist of many disconnected local minima or that suffer from noisy data, as no gradient information is required. They can also be applied to multi-objective problems by creating an archive of non-dominated solutions [3].

In the implementation used here, an initial input vector x_n is assessed against each of the objective functions $f_i(x_n)$, before being randomly adjusted and re-evaluated. The probability P that new solution will be accepted and stored in the archive is given by

$$P = \min\left(1, \exp\left(\sum_i \frac{f_i(x_{n+1}) - f_i(x_n)}{T_i}\right)\right) \qquad (1)$$

In common with standard simulated annealing algorithms, the temperature for each objective T_i is reduced according to a schedule, thereby lowering the probability over time that an objectively worse solution will be accepted as the new solution. The definition of an effective temperature schedule is one of the biggest challenge of such optimisers [4]. Periodically, the optimisation is 'reset' by re-starting the procedure using a member of the non-dominated archive, and the temperatures increased.

OPTIMISATION PACKAGE

The optimisation package is predominantly written in Python, with the except of the simulated annealing algorithm. This section has been written in Matlab, with Python wrappers to link it to the main Python control scripts. The structure of the code is illustrated in Fig. 1, and consists of:

- the main user interface (UI)
- the optimiser
- the interaction controller

Main UI

The main UI can be seen in Fig. 2. This is where the input variables and objective function(s) are configured and the choice of optimisation algorithm is made. The input variables can be selected either singly or in groups, and absolute or relative bounds can be defined for each parameter. When adding in groups, the user can choose whether to maintain any initial differences, or if all members should be set to the same absolute values. Minimum wait times after setting the process variables (PVs) can also be defined.

At present it is only possible to select either one or two objective functions, but extension to higher numbers is straightforward. The UI can be used to define the number of times each objective should be measured, and the delay between acquisitions can be set to ensure the data is fresh. It is also possible to choose whether to maximise or minimise each objective. The main UI is also used to select which algorithm is used to carry out the optimisation, and pre-defined configurations can be loaded or saved.

Optimiser

Once the problem has been defined, the optimisation script is called. This section allows the algorithm-specific parameters to be set (such as population size, mutation probability, etc.), and controls any plotting/visualisation relevant to the chosen algorithm. Interaction with the machine is carried out using the interaction controller.

Interaction Controller

The interaction controller is responsible for applying the input variables to the machine via EPICS and for returning each of the objective functions. The controller maps between the algorithm-specific parameters (such as the relative change in strength for each sextupole family) and the machine parameters (i.e. the absolute individual magnet

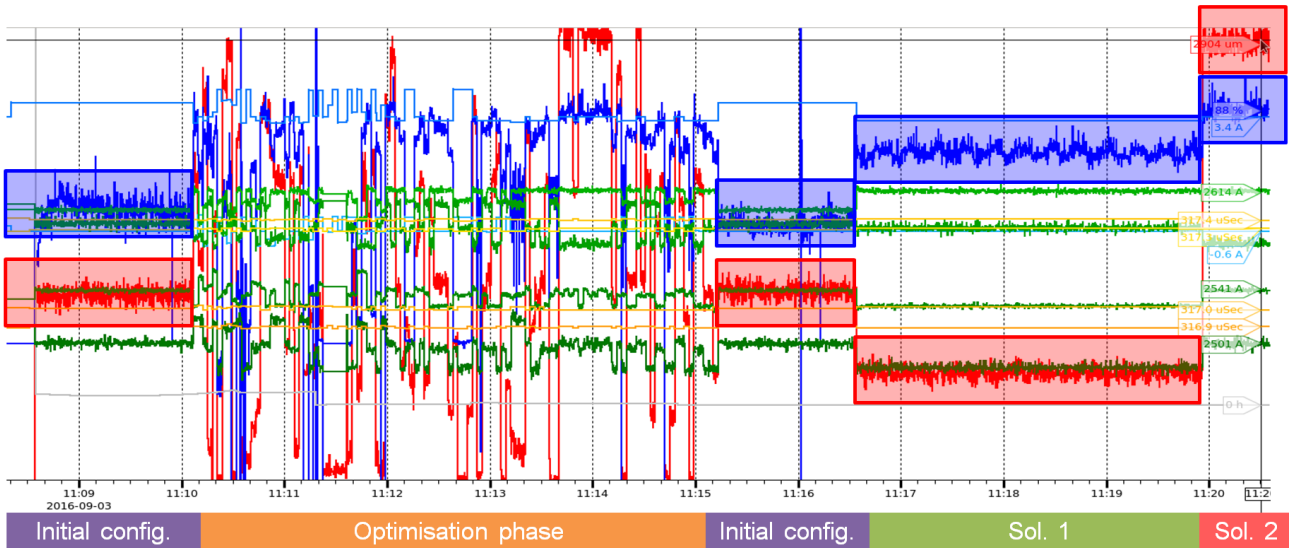

Figure 3: Evolution of MOGA variables during the injection optimisation test. The injection efficiency and stored beam oscillation amplitude are highlighted in blue and red respectively. Green lines indicate kicker amplitudes, orange lines show kicker timing, and light blue lines show the strength of two steerers in the transfer line.

strengths). Depending upon the configuration, these two sets can have different sizes and different values. For each objective function, the mean and standard error are returned over the specified number of measurement samples. Each objective is measured simultaneously in order to reduce the total acquisition time.

Running the Optimiser

Once the optimisation has started, a figure is displayed giving information about progress. In the case of the GA, this consists of a progress bar, a strip-tool of the parameter values, and non-dominated fronts after each generation. At the end of the optimisation the machine is returned to its initial configuration and a second, interactive figure is displayed. Using this figure it is possible to select any given point on the non-dominated front by clicking on it. This in turn pops up another UI which displays information about the selected point (such as input parameter and objective function values), and gives the option to apply the settings to the machine. The details of each optimisation run are archived for later analysis.

MACHINE TESTS

To illustrate the effectiveness of the optimiser, the results of a GA run are shown in Figs. 3 and 4. In this example, the objective was to recover injection efficiency from an initially poor state, whilst minimising disturbance to the stored beam (blue and red lines respectively in Fig. 3). To achieve this goal, the individual strengths and timings of the four injection kicker magnets plus the strengths of the last two steerers in the transfer line were varied.

There are three points highlighted in Fig. 4. In purple, the initial machine state is shown, then two candidate solutions are shown in green and red. The first manages to improve

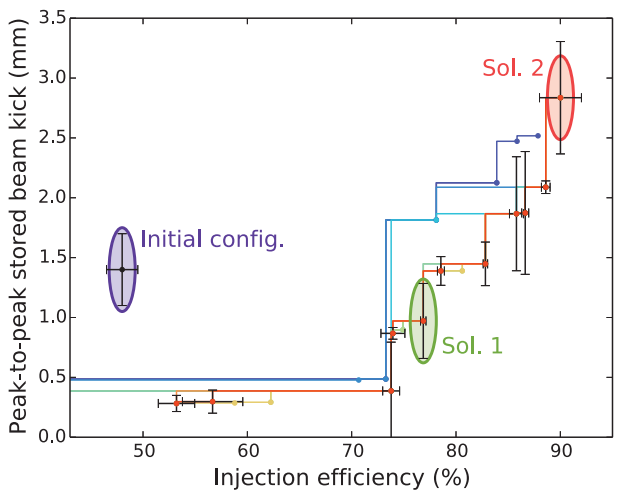

Figure 4: Non-dominated fronts after each generation for the optimisation shown in Fig. 3. See text for more details.

both objectives simultaneously. The second achieves an even higher injection efficiency at the expense of a considerable increase in stored beam disturbance, as confirmed by the data shown in Fig. 3.

CONCLUSIONS

Despite the demonstrated effectiveness of the tool, the optimisation package remains a work in progress. Future developments include the addition of other optimisation algorithms (such as Particle Swarm [5]), the ability to use more complex objective functions beyond simple PVs, and outlier-rejection for the measured data. The ability to view/apply the results of a previous optimisations will also be added.

REFERENCES

[1] X. Huang, J. Corbett, J. Safranek, J. Wu, "An algorithm for online optimization of accelerators", *Nucl. Instr. Meth. A*, vol. 726, pp. 77–83, 2013.

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

[3] A. Suppapitnarm, K.A. Seffen, G.T. Parks, P.J. Clarkson, "A simulated annealing algorithm for multiobjective optimization", *Engineering Optimisation*, vol. 33, No. 1, pp. 59–85, 2000.

[4] W. Ben Ameur, "Computing the initial temperature of simulated annealing", *Computational Optimisation and Applications*, vol. 29, No. 3, pp. 369–385, 2004.

[5] X. Pang, L.J. Rybarcyk, "Multi-objective particle swarm and genetic algorithm for the optimisation of the LANSCE linac operation", *Nucl. Instr. Meth. A*, vol. 741, pp. 124–129, 2014.

ISBN 978-3-95450-182-3