JOURNAL OF SYNCHROTRON RADIATION

ISSN 1600-5775

*AI-BL*1.0: a program for automatic on-line beamline optimization using the evolutionary algorithm

Shibo Xi,[a] Lucas Santiago Borgna,[b] Lirong Zheng,[c] Yonghua Du[a]* and Tiandou Hu[c]*

[a]Heterogeneous Catalysis, Institute of Chemical and Engineering Sciences, Agency for Science, Technology and Research, 1 Pesek Road, Jurong Island 627833, Singapore, [b]Department of Physics, Loughborough University, Leicestershire LE11 3TU, UK, and [c]Beijing Synchrotron Radiation Facility, Institute of High Energy Physics, Chinese Academy of Sciences, 19B Yuquan Road, Shijingshan District, Beijing 100049, People's Republic of China. *Correspondence e-mail: du_yonghua@ices.a-star.edu.sg, hutd@ihep.ac.cn

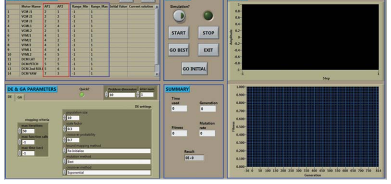
Received 27 July 2016

Accepted 11 November 2016

Edited by S. Svensson, Uppsala University, Sweden

Keywords: automatic beamline optimization; evolutionary algorithm; Labview.

In this report, *AI-BL*1.0, an open-source Labview-based program for automatic on-line beamline optimization, is presented. The optimization algorithms used in the program are Genetic Algorithm and Differential Evolution. Efficiency was improved by use of a strategy known as Observer Mode for Evolutionary Algorithm. The program was constructed and validated at the XAFCA beamline of the Singapore Synchrotron Light Source and 1W1B beamline of the Beijing Synchrotron Radiation Facility.



1. Introduction

In synchrotron radiation facilities, it is of fundamental importance to maintain beamlines at their optimal conditions. Due to the number of degrees of freedom, obtaining and maintaining beamlines under optimal conditions is not a straightforward task. Conventionally, optimization of the beamline components is done *via* manual intervention. Not only is this method severely time-consuming, but it does not always assure us that the global optimum is reached. The complications with manual adjustments arise simply from the large degrees of freedom and non-linear drifts in equipment. In a recent research paper (Xi *et al.*, 2015), a general method based on the Genetic Algorithm (GA) (Holland, 1975) was proposed and shown to successfully perform the desired beamline optimization automatically. The GA-based optimization strategy was applied on the XAFCA beamline (Du *et al.*, 2015) of Singapore Synchrotron Light Source (SSLS) and it was demonstrated that this method can optimize all the optical components of the XAFCA beamline simultaneously and efficiently.

From a higher level of abstraction, the GA method is a subclass of the Evolutionary Algorithm (EA) implementations (Hertz & Kobler, 2000). This subclass also includes Differential Evolution (DE) (Storn & Price, 1997; Amaro *et al.*, 2011), Neuroevolution (Whitley *et al.*, 1993), Learning classifier system (Holland, 1976), *etc.* EAs are a type of numerical technique used for solving problems of high complexity, particularly those involving global optimization. Their fundamental basis stems from mimicking the process of natural selection and survival of the fittest based on Darwin's theory of evolution. In an EA, a population of artificial creatures are generated over the search space of the problem, which is analogous to Monte Carlo techniques. They compete continually with each other to discover optimal areas of the search space. EA has been successfully used for a wide range of applications (Bäck *et al.*, 1993; Alander, 1995). Typically, in EAs, the gradient information and mathematical model of the problem to be solved are not required. The only feedback necessary is the evaluation of each trial relative to the output function. This high degree of generality enables EAs to be well suited for various complex search spaces, which is the case for beamline optimization. The literature on applications of EAs provides an empirical perspective on their robustness (Rogenmoser *et al.*, 1996). Furthermore, its population-based characteristics make for an embarrassingly parallel problem.

In this report, details, including the modifications required for other beamlines, of the *AI-BL*1.0 program are described. The program will implement the automatic beamline optimization methods based on DE and GA, with the OMEA (Xi *et al.*, 2015), enhancement. *AI-BL*1.0 was developed using the Labview platform (National Instruments, USA). Please contact the authors for the program.

2. Architecture

In this report, we discuss the use of EA to optimize a beamline, for which we can adjust its conditions through the use of stepper motors (SMs). The program's implementation of EA is performed *via* both the GA and DE approach. The typical pseudo-codes for both GA and DE are presented in Appendix *A*. For *AI-BL*1.0, some of the GA modules come from the open-source code *Waptia* (Golebiowski, 2009), and some of the DE modules are reorganized from Labview's *Global Optimization Toolbox*.

For this particular optimization problem, the optimization parameters are defined as the positions of the SMs. These parameters directly affect the conditions of the optical components. Further details about how the search space is constructed for the use of GA were introduced in our previous research paper (Xi *et al.*, 2015). The fitness of EA modules depends on the optimization objective, such as flux, resolution power, spot shape and position. The aim of the optimization is to reach the best alignment of the optical arrangement through the evolution of the SM position. Therefore, to make the connection with EA, the SM positions are considered to be the genes of EA modules, while the combination of these genes forms an individual. As shown in Fig. 1, the EA modules, which here include both GA and DE, are responsible for the evolution of genes. The optimization object is used to adjust the conditions according to the instructions from the EA modules and provide feedback to the EA modules. The EA modules apply the necessary EA operations, such as mutation and crossover of individuals, generating their offspring. The evaluation of each offspring individual is then based upon the feedback received from the optimization object.

Using a classical EA approach for directly optimizing the beamline yields low efficiency. To accelerate the optimization, we applied OMEA to the evaluation. In classical EA, the evaluation of individuals is performed sequentially. More specifically, as shown in Fig. 2(*a*), the two adjacent points

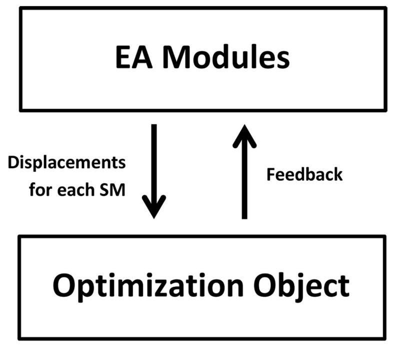


Figure 1
Flow chart of the framework for optimization using EA. For beamline optimization, the information that the EA modules provide to the beamline is commands, which make the SMs move by a certain displacement. The information that the beamline communicates back to the EA modules is the position for each SM and the beamline parameters such as flux, resolution power, spot shape and position, *etc*.

A and *B* in the search space are evaluated to obtain their respective fitness f_A and f_B. Meanwhile, OMEA will also evaluate individuals on the path from points *A* and *B*, gaining an additional degree of information, as shown in Fig. 2(*b*). A comparison of these individuals will then yield the fittest to be point *C* with its respective fitness f_C. The individual *B* and its fitness f_B will then be replaced by *C* and f_C, respectively. For systems where the conditions are adjusted by SMs, OMEA is more suitable. Removing the layer of abstraction made in Fig. 2(*b*), the evaluation of individuals on the path from point *A* to *B* refers to the movement of the SM from *A* to *B*. The EA modules will then record the feedback from the optimization object, in real time, determining the fitness of individuals that

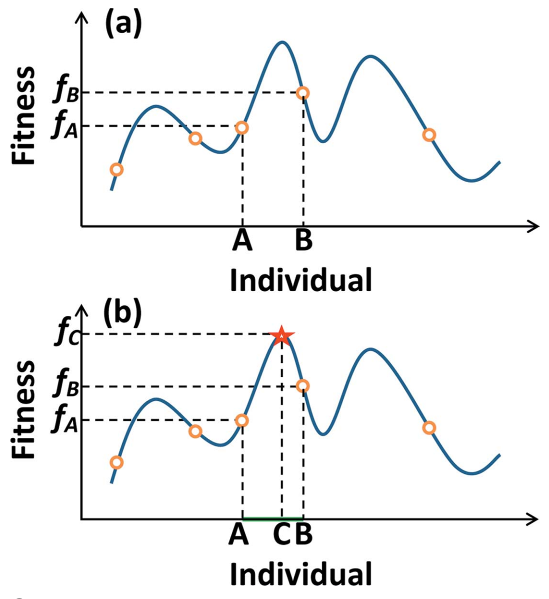


Figure 2
Diagram for classical EA (*a*) and OMEA (*b*). Points *A* and *B* are two adjacent trail solutions (individuals) within the search space. At this stage of evaluation, a classical EA just evaluates these two points in succession, while, in OMEA, the whole path from points *A* and *B*, highlighted in green, will be evaluated. The best point on the path, point *C*, labelled with a red star, with maximum fitness, will replace the individual *B*.

belong to the path *A* to *B*. This means that OMEA can obtain more information without adding delays to the classical EA scheme. Therefore, OMEA is more efficient than the classical EA scheme.

3. Features of *AI-BL*1.0

*AI-BL*1.0 has been designed and developed with the aim of being easily configurable, expandable and maintainable. A Labview-based graphical user interface (GUI), which is composed of five panels, is available for launching the program. The normal view of the GUI is shown in Fig. 3.

The first panel, MOTOR PARAMETERS, defines the required inputs for each SM. The required inputs are the name of the SM, address of the SM (framed by a red rectangle) and the search range of the SM (framed by a purple rectangle). The address for each SM comprises two strings, through which the program can find the SM to be optimized. When *AI-BL*1.0 is running, the original value ('Initial value') and current best value ('Current solution') corresponding to the position of each SM will be displayed in this panel. The data type of all inputs here is a string. The parameters can be exported to an ACSII file by clicking the 'Save' button, and can also be imported by clicking the 'Load' button.

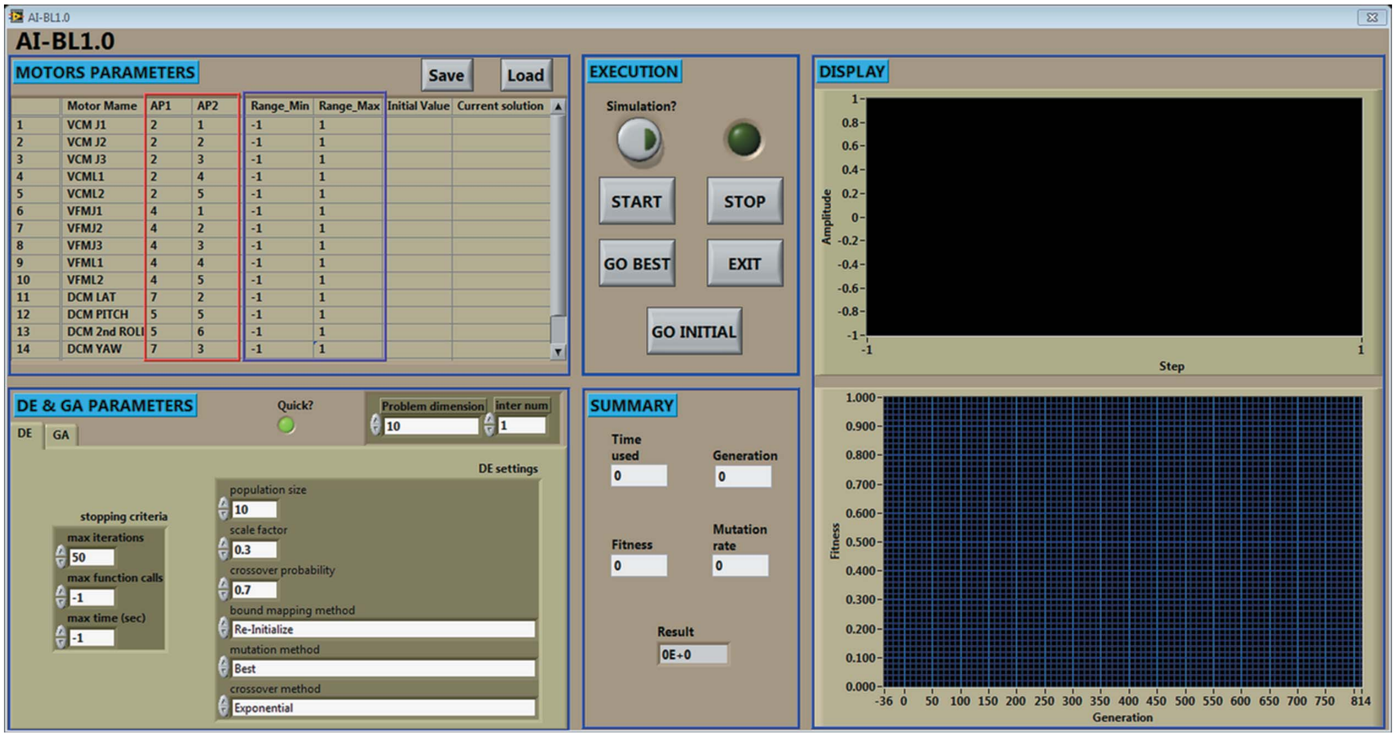
The second panel defines the DE & GA PARAMETERS. Detailed information about each parameter is presented in Appendix *B*.

The EXECUTION panel contains the interface for controlling the program. The button 'Simulation' is used for switching between simulation and real optimization. If the simulation button is active, the program will optimize the 'benchmark' function, which is located in the '..\benchmarks' file folder. To use the 'Simulation' function, benchmark.vi can be replaced with a user-defined method. The START and STOP buttons are for initiating and stopping the optimization procedure, respectively. 'Round LED' is used to indicate when the algorithm is executing the optimization process. Once the procedure is completed or halted, the GO BEST feature can be used to move all the SMs to their corresponding optimal solution. The GO INITIAL button can recover the SMs to their original positions. The EXIT button is used for safely quitting the program.

The SUMMARY panel will present in real time the parameters corresponding to the status of the program under operation. 'Time used' indicates the time elapsed since the START button was pressed. 'Generation' displays the current generation. 'Fitness' indicates the best fitness achieved from initiating the optimization procedure. 'Mutation rate' is used to display the current mutation rate for when a GA-based optimization is running, with an unspecified mutation target rate. The numeric indicator 'Result' displays the feedback from the beamline when the optimization is completed.

The DISPLAY panel will display in real time the feedback of the beamline (upper sub-panel) and the plot of fitness against generation number (lower sub-panel), when the program is running.

During the optimization, two log files with a timestamp (using system time) will be created and saved under the '..\ log' directory. One will record the real-time SM positions and the corresponding feedback; the other will record the optimization results of each generation. All the data plotted in the DISPLAY panel will be recorded in these two files.



	Motor Mame	AP1	AP2	Range_Min	Range_Max	Initial Value	Current solution
1	VCM J1	2	1	-1	1		
2	VCM J2	2	2	-1	1		
3	VCM J3	2	3	-1	1		
4	VCML1	2	4	-1	1		
5	VCML2	2	5	-1	1		
6	VFMJ1	4	1	-1	1		
7	VFMJ2	4	2	-1	1		
8	VFMJ3	4	3	-1	1		
9	VFML1	4	4	-1	1		
10	VFML2	4	5	-1	1		
11	DCM LAT	7	2	-1	1		
12	DCM PITCH	5	5	-1	1		
13	DCM 2nd ROLl	5	6	-1	1		
14	DCM YAW	7	3	-1	1		

Problem dimension: 10
inter num: 1

DE settings

stopping criteria
max iterations: 50
max function calls: -1
max time (sec): -1

population size: 10
scale factor: 0.3
crossover probability: 0.7
bound mapping method: Re-Initialize
mutation method: Best
crossover method: Exponential

Time used: 0
Generation: 0
Fitness: 0
Mutation rate: 0
Result: 0E+0

Figure 3
GUI of *AI-BL*1.0.

4. Usage instructions of *AI-BL*1.0

To apply *AI-BL*1.0 to other beamlines, seven sub-VIs need modifications. The core functionalities of all sub-VIs are presented in Table 1.

As shown in Table 1, the sub-VIs, required for users, can be classified into controlling modules of SMs, feedback from beamline and benchmark function for simulation mode. There are a total of six sub-VIs required for controlling the SM: Connect SM, Inquire status of SM, Inquire positions of SM, Set speed, SM move and Kill SM. All of these sub-VIs require the input parameter 'Device table' that passes the address of the SM. Each row of the table contains two elements, such as axis number and motor number, which is the case of XAFCA beamline. The number of rows is equal to the number of SMs, which is also the dimension of the problem. For the VI SM Move, the input parameter 'Stepper array' contains the counts of each SM to be run, and the input parameter 'Maxspeed' gives the upper limit of the running speed for the SM. For the sub-VI Feedback, there are two input parameters: the address for the Ionization Chamber (IC) and the integration time of the IC. The output for this sub-VI is just one numeric data, the reading of the IC for instance. Users at other beamline can also check the templates for these input sub-VIs, which are placed in the '..\Input files' folder. These templates contain all of the necessary information about the input and output (I/O) data types for the sub-VIs.

Table 1
Sub-VIs required for *AI-BL*1.0.

All of the SMs controlling the VIs and feedback from the beamline VI are located in the '..\input files' folder. The benchmark VI is located in the 'benchmarks' folder.

Name of sub-VI	Input data type	Output data type	Function
SM controlling			
Connect SM	Char Array	Boolean	Establish the connection between GA/DE modules and SMs driver
Inquire status of SM	Char Array	Boolean	Inquire whether at least one SM is running
Inquire positions of SM	Char Array	Float Array	Inquire the value of encoder corresponding to each SM
Set speed	Float Array Char Array	None	Set speed for each SM
SM Move	Float Array Char Array Float	None	Drive SMs to run by steps reading from steps array
Kill SM	Char Array	Boolean	Stop and disconnect SMs
Feedback from beamline			
Feedback	Char Float	Float	Obtain the feedback from beamline, such as flux, resolution power, spot shape and position, *etc*.
Benchmark function			
Benchmark	Float Array	Float	Evaluate the benchmark function

5. Testing

Tests were carried out at the XAFCA beamline of SSLS and 1W1B beamline (Xie *et al.*, 2007) of the Beijing Synchrotron Radiation Facility (BSRF). At XAFCA, the conditions of the vertical collimating mirror (VCM), double-crystal monochromator (DCM) and vertical focusing mirror (VFM) were optimized with the aim to obtain maximum photon flux at the sample position. The conditions of these optical components are controlled by a total of 15 SMs. The flux at the sample position, monitored by the IC, is used to provide the feedback from the beamline. The parameters used in this testing are listed in Table 2. These parameters were chosen by a combination of reference values from previous research (Boyabatli & Sabuncuoglu, 2004; Pedersen, 2010) and empirical values obtained from our hot commissioning. Further details on the experimental setup are given by Xi *et al.* (2015).

Before optimization, each SM to be optimized was moved by a uniform random displacement, between 0 and 0.5 mm. This led to a complete misalignment of the beamline, resulting in negligible flux detection at the sample position. The positions of these SMs are regarded as the initial conditions of optimization. Further testing was carried out using the same initial conditions.

Table 2
Parameters used to optimize the optical components of the XAFCA beamline.

Search ranges for all SMs are ±0.5 mm.

DE		GA	
Parameter name	Parameter value	Parameter name	Parameter value
Settings			
Population size	10	Population size	10
Scale factor	0.3	Significant digits	5
Crossover rate	0.7	Crossover rate	0.85
Bound mapping method	Re-initialize	Reproduction plan	Full generation replacement
Mutation method	Best	Elitism	ON
Crossover method	Exponential	Mutation method	One-point, fixed rate
		Creep mutation	Off
		Mutation rate	0.1
		Relative fitness differential	1
Stopping criteria			
Max iterations	50	Satisfying fitness	Inf
Max function calls	−1	Maximum iterations	50
Max time (s)	−1		

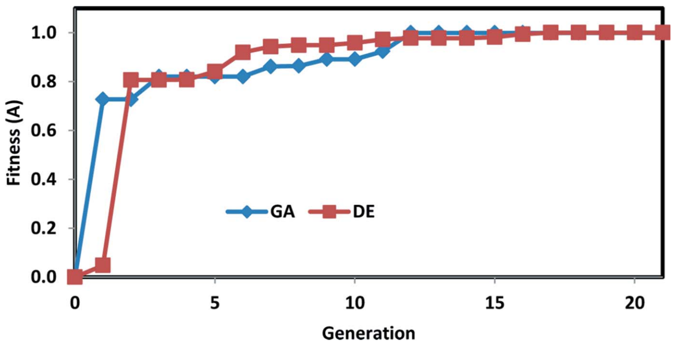


Figure 4
Fitness *versus* generation using GA and DE. For better comparison, the optimized fitness is normalized to unity.

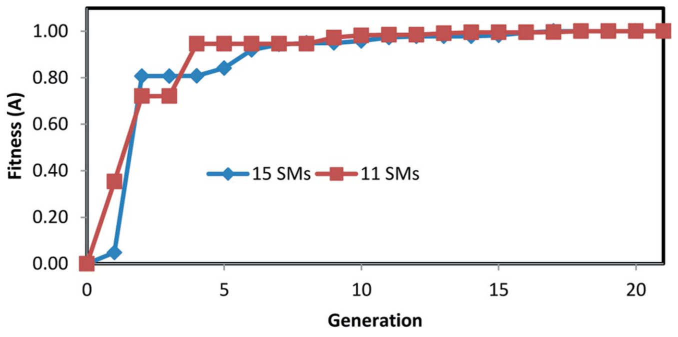


Figure 5
Fitness *versus* generation for optimizing 11 and 15 SMs using DE. For better comparison, the optimized fitness is normalized to unity.

Fig. 4 shows the fitness *versus* generation for GA and DE. In this testing, 11 SMs [VCM_J1, VCM_J2, VCM_J3, VFM_J1, VFM_J2, VFM_J3, DCM_LAT, DCM_PITCH, DCM_2nd_ROLL, DCM_YAW, DCM_PARA; details of these SMs are given by Xi *et al.* (2015)] were optimized. After 16 and 21 generations of evolution for GA and DE, respectively, the solution *versus* generation curves reached a plateau, where successive iterations no longer produced better results, which suggested that the optimized condition was reached. Both the GA module and DE module successfully performed the optimization procedure. Fig. 5 shows the fitness *versus* generation curve for the optimization of 11 SMs and 15 SMs (besides the above-mentioned 11 SMs, VCM_L1, VCM_L2, VFM_L1 and VFM_L2 were added) using DE. This demonstrates that the convergence time does not increase significantly when four SMs were added to the optimization parameters.

For the test carried out at the 1W1B beamline of BSRF, the focusing mirror condition was optimized to maximize the flux at the sample position monitored by an IC. When only four SMs are included in the optimization process, the program is typically able to find the optimal solution within seven generations.

6. Discussions and conclusions

AI-BL is an automatic beamline optimization program based on Labview. The core algorithm is implemented by GA and DE. The algorithm is enhanced by OMEA, which is inherently more efficient than the traditional EA. The user interface is an intuitive and convenient GUI. All the parameters for optimization can be input through the GUI. The progress of the program, such as solution and fitness, are displayed on the GUI in real time when the program is running. Testing on the XAFCA beamline indicated that, by using both GA and DE, this program can simultaneously optimize the condition of the VCM, VFM and DCM, which are controlled through 15 SMs, to obtain maximum flux at the sample position. The abstract and object-oriented nature of the program enables it to be versatile and easily modified to suit other beamlines. As mentioned previously, by replacing seven sub-VIs, the business logic of the program can be applied to other beamlines. The aim of optimization can be virtually anything concerning beamlines, such as flux, resolution power, spot shape and position, *etc*.

The main advantage of the EA is that the mathematical model of the problem to be solved is not required. It is sufficient to have an input (stepper motors to adjust) and output (the target to optimized, for example, the flux at the sample position, beam shape or position, *etc*). Confinement of the search space for each parameter ensures that all stepper motors are under control and all optical components remain safe during the optimization. Tests have been carried out (Xi *et al.*, 2015) to demonstrate that our program can optimize the beamline robustly and globally. Fig. 4 shows that the number of generations to complete the optimization of 11 SMs is 16 and 21 for GA and DE, respectively. However, the time spent for GA is longer than that for DE. The recent tests at XAFCA showed that it took around 10 min for DE and 15 min for GA to reach their target goal, indicating that DE is more efficient than GA in this program.

APPENDIX *A*
Typical GA and DE in pseudo-codes

```
Pseudo-Code for GA:
    Begin
      Generate, randomly, an initial population of solutions.
      Evaluate the fitness of the initial population.
      Repeat
        1. Select a pair of parents according to the 'survival
           of the fittest' principle.
        2. The selected parents mate to create two offspring.
        3. Offspring mutate.
        4. Evaluate the mutated offspring.
        5. All the offspring replace their parents and
           compose the new generation.
      Until termination criteria are satisfied.
    End.
```

```
Pseudo-Code for DE:
  Begin
    Generate, randomly, an initial population of solutions.
    Determine the fitness of the initial population.
    Repeat
      1. Repeat
        For each parent, select 3 individuals randomly.
        Apply DE operators to the parent and the three
        selected individuals to generate an offspring.
        Until the number of offspring is equal to the
        population size.
      2. Evaluate each offspring.
      3. For each offspring
      If offspring is more fit than its parent
        Its parent is replaced.
    Until termination criteria are satisfied.
  End.
```

In a typical DE, the DE operator is a prime operator, and the implementation of this operation makes DE different from other evolutionary algorithms. The main function of the DE operator is combining the randomly selected three individuals (vectors) to generate a new vector, and then mixing the new vector with the current parent. Here, the combination of vectors is also called 'mutation'; the mixing of combined vector and parent vector is also called 'crossover'.

APPENDIX *B*
Parameters for the GA/DE module

The second panel defines the parameters for the GA/DE module. Detailed information about each parameter is presented as follows.

Program Dimension: controls the dimension for the problem to be solved.

Inter Num: dedicated for **simulation**. When a mathematic function is optimized, a local-learning method, as mentioned above, can be implemented though interpolation between two individuals, in which the number of interpolations is equal to **Inter Num**.

Quick: switch between quick and slow mode. When it is lit, the beamline will keep sending feedback to EA Modules when the SMs are running, otherwise the feedback will be sent merely when the SMs are suspended.

DE

Termination criteria

Max iterations: when the number of generation equals this value, the program will stop and return the solution of the last generation as the solution of the problem. When it is set as −1, this criterion will not work.

Max function calls: when the number of calls of fitness function equals this value, the program will stop and return the solution of the last generation as the solution of the problem. When it is set as −1, this criterion will not work.

Max time (s): When the time elapsed reaches this value, the program will stop and return the solution of the last generation as the solution of the problem. When it is set as −1, this criterion will not work.

DE Setting

Population size: the number of individuals.

Scale factor: The factor used to multiply the differ ential vector of two randomly selected individuals to generate the trial vector.

Bound mapping method: method to map the vector, generated by the DE operator, into the search space.

Mutation method: strategy to combine the randomly selected individuals to the new vector.

Crossover method: strategy to mix the new vector generated by mutation operator with the predetermined vector.

GA

Termination criteria: when the best fitness equals this value, the program will stop and return the solution of the last generation as the solution of the problem. When it is set as Inf, this criterion will not work.

GA Settings

Population size: the number of individuals.

Significant digits: number of significant digits for each variables being optimized.

Crossover rate: the possibility that two selected individuals mate.

Reproduction plan: the strategy used for replacing parent generation with offspring generation.

Elitism: the best individual of parent generation will replace the worst individual of the offspring generation if Elitism is selected.

Mutation method: method used to mutate.

Creep mutation: if mutations are performed with 'creep' operator, it means that change caused by mutation is rather smooth than rapid.

Mutation rate: fixed rate of mutation.

Min mutation rate: lower bound of mutation rate, applicable when the mutation rate is not fixed.

Max mutation rate: upper bound of mutation rate, applicable when the mutation rate is not fixed.

Relative fitness differential: higher number causes more often better individuals to become parents.

Acknowledgements

This work is supported by the National Natural Science Foundation of China (Y61161005Z).

References

Alander, J. T. (1995). *Indexed bibliographies of genetic algorithms*, Technical Report 94–1-*. University of Vaasa, Finland.

Amaro, E., López, J. M., Zamudio, V., Baltazar, R. & Casillas, M. A. & Callaghan, V. (2011). *Proceedings of the 9th International Conference on Intelligent Environments*, Vol. 17, p. 222.

Bäck, T., Hoffmeister, F. & Schwefel, H.-P. (1993). *Applications of evolutionary algorithms*, Technical Report SYS-2/92. System Analysis Research Group, University of Dortmund, Germany.

Boyabatli, O. & Sabuncuoglu, I. (2004). *J. Syst. Cybern. Inf.* **4**, 78.

Du, Y., Zhu, Y., Xi, S., Yang, P., Moser, H. O., Breese, M. B. H. & Borgna, A. (2015). *J. Synchrotron Rad.* **22**, 839–843.

Golebiowski, W. (2009). *Waptia – genetic optimization algorithm*, https://lavag.org/topic/10984-cr-waptia-geneticoptimization-algorithm/.

Hertz, A. & Kobler, D. (2000). *Eur. J. Oper. Res.* **126**, 1–12.

Holland, J. H. (1975). In *Natural and Artificial Systems.* The University of Michigan Press.

Holland, J. H. (1976). In *Progress in Theoretical Biology*, Vol. 4, edited by R. Rosen and F. M. Snell. New York: Plenum.

Pedersen, M. E. H. (2010). *Good parameters for differential evolution.* Technical Report HL1002. Hvass Laboratories.

Rogenmoser, R., Kaeslin, H. & Blickle, T. (1996). *Lect. Notes Comput. Sci.* **1141**, 849–858.

Storn, R. & Price, K. J. (1997). *Glob. Optim.* **11**, 341–359.

Whitley, D., Dominic, S., Das, R. & Anderson, C. W. (1993). *Mach. Learn.* **13**, 259–284.

Xi, S., Borgna, L. S. & Du, Y. (2015). *J. Synchrotron Rad.* **22**, 661–665.

Xie, Y., Hu, T., Liu, T. & Zhang, J. (2007). *AIP Conf. Proc.* **879**, 856–859.