Automated Materials Spectroscopy Analysis using Genetic Algorithms

Miu Lun Lau[1], Min Long[1], and Jeff Terry[2]

[1] Boise State University, Boise ID USA,
andylau@u.boisestate.edu, minlong@boisestate.edu
[2] Illinois Institute of Technology, Chicago, Illinois,
terryj@iit.edu

Abstract. We introduce a Genetic Algorithm (GA) based, open-source software to solve multi-objective optimization problems of materials characterization data analysis including EXAFS, XPS and nanoindentation. The modular design and multiple crossover and mutation options make the software extensible for other applications too. This automation of the analysis is crucial in the era when instrumentation acquires data orders of magnitude more rapidly than it can be analyzed by hand. Our results demonstrated good fitness scores with minimal human intervention.

Keywords: Genetic Algorithm, EXAFS, X-Ray

1 Introduction

Data analysis of materials characterization is a useful tool with growing interests in the material science community due to its capability in detecting the structure of materials by monitoring the interactions between electrons and X-ray radiation. The performance of this process is limited by two main factors: the significant inputs from users needed to retrieve important structural parameters and the high quality data can be retrieved from advances in instrumentation. However, the experiential data collected from such modern instruments are in the orders of magnitude larger than it can be analyzed by trained personnel. For example, the Fourth Generation Synchrotron Light Source is expected to produce data at a rate of 2-3 orders of magnitude greater than current rates as high as 6 GB/s [1], allowing for in-situ real-time characterization of materials feasible. Such issues can also cause reproducibility problems [2] that are slowing research productivity, discouraging the quest for research excellence, and inhibiting effective technology transfer and manufacturing innovation.

In order to address the need to analyze massive datasets both quickly and accurately through solving such a multi-objective optimization problem, we have been developing a Genetic Algorithm (GA) based, open-source code that can analyze a variety of materials characterization data types with minimal human input to retrieve important parameters [3]. Although it is still under active development, it has already demonstrated power in automatically analyzing extended

X-ray Absorption Fine Structure (EXAFS) data, giving a reproducible description of the local atomic structure of materials.

The GA-based EXAFS analysis package called `EXAFS Neo` [4] is written primarily in Python. It requires the installation of `Larch` software [5] and utilizes a version of `FEFF8.51` [6] code contained within `Larch` for calculating the initial scattering paths. The code was not parallelized and we found that parallelization was really not needed for effective use as an analysis tool. However, we strongly recommended that users execute multiple calculations of the same date set simultaneously for further error analysis. The included graphical user interface (GUI) allows for populating multiple parameter sets for simultaneous exploration of parameter space. Several tutorial video demonstrations of the `EXAFS Neo` package are available for viewing at the package download website [4]. Due to the modular design, this open-source software can be extended to various other analytical spectra, and recently been extended to X-ray Photoelectron Spectroscopy (XPS), and nanoindentation.

We will briefly introduce the materials characterization analysis in Section II, with a focus on distinct parameters not commonly seen in other GA codes. The design and implementation of GA for this problem is given in Section III. We then evaluate the experimental results and give performance analysis of the code in Section IV. The summary of the work is given in Section V.

2 Analysis of Materials Characterization Data

We use EXAFS as an example to demonstrate the process of materials characterization analysis. The principles discussed here can be applied to other spectrum analysis such as XPS and nanoindentation. EXAFS is a tool to study local structure around selected element within atomic and molecular scale, and can be applied to a wide range of materials. The absorption spectrum $\mu(E)$ with energy E of X-ray photons can be represented as a combination of the background μ_0 and the oscillation from the scattering events $\chi(E)$, i.e., $\mu(E) = \mu_0(1+\chi(E))$. It can be rewritten using photoelectron wavenumber $k = [\frac{2m}{\hbar^2}(E - E_0)]^{1/2}$, where m is the electron mass, $\hbar$ is the Plank's constant, and E_0 is called the energy of the absorption edge, as

$$\chi(k) = \frac{\mu(k) - \mu_0(k)}{\mu_0(k)}, \tag{1}$$

The oscillations $\chi(k)$ are caused by photoelectron scattering which can be consist of either *single* and/or *multiple* scattering events. In *single* scattering events, the electron wave scatters from only one neighboring atom before returning to the source atom. In *multiple* scattering events, the wave may travel to multiple nearby atoms through a variety of paths before returning to the source atom. The trajectories of each photoelectron scattering event are described as *paths*. These paths will be used in the following sections.

Fig. 1 shows the scattering paths. The core absorbing atom (yellow) emits a scattered electronic wave toward the surrounding atoms from the excitation (blue), when the wave upon reaching the wave will interact with the potential of a neighboring atom and will be scattered to nearby atoms and also back toward the absorbing atom. While single scattering paths generally dominate most EXAFS spectra, multiple scattering paths can also contribute important contributions, such as those in crystalline materials. It is worth noting that the existence of "path" complicates the processing of gene components in chromosome in the GA code.

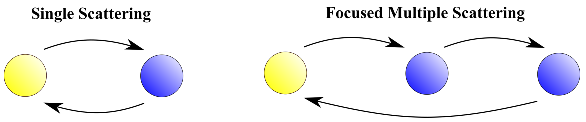

Fig. 1. Example of the single and multiple photoelectron scattering paths, from which the photoelectron will scatter from an absorbing atom (yellow) to nearby atoms (blue).

The measured X-ray absorption spectrum (XAS) [7] consists of two distinct regions: X-ray Absorption Near Edge Structure (XANES) and Extended X-ray Absorption Fine Structure (EXAFS), which begins at the end of the XANES region and extends beyond effectively until the oscillations are damped out. The endpoint is very dependent upon the nature of the scattering atoms. It is used to study interatomic distances, coordination numbers, and lattice dynamics, from which one can often infer the surface chemistry of complex systems.

The interference of electron scattering that leads to the measure of EXAFS oscillations is due to the interactions of photoelectrons, ejected from inner core shells by resonant radiation, with neighboring atoms. These scattered electron waves modulates the wavefunction of the original photoelection. If the interference is destructive, a photon cannot be absorbed if the the electron wave cannot be created. Conversely, if the interference is constructive, more photons can be absorbed. Sayers et al. [8] was first to invert measured experimental EXAFS data into radial distribution functions using a simple point scattering theory. Extensions using this simple methodology will result in full EXAFS equation:

$$\chi(k) = \sum_{i(paths)} \frac{(S_0^2 N_i) F_i(k)}{k R_i^2} e^{-2\sigma_i^2 k^2} e^{-2R_i/\lambda(k)} \sin[2kR_i + \phi_i(k) + \delta_c(k)], \quad (2)$$

where i represents an individual scattering path, R_i is half of the scattering distance, $\phi_i(k)$ is phase shift due to scattering, and $\delta_c(k)$ is the final phase shift of the absorbing atom. N_i is the degeneracy, and S_0^2 is the amplitude reduction factor due to excitation. S_0^2 and N_i are coupled together to describe the amplitude of each scattering path. $F_i(k)$ is the effective scattering amplitude for the waves in path and σ^2 is the Debye-Waller factor, which accounts for the thermal and static disorder.

We can observe from Eqn. (2) that the EXAFS signal is a summation of the sinusoidal waves of varying amplitudes from scattering events from neighboring atoms. The waves are inherently spherical in nature and are affected by the type of atoms, temperature, neighboring atoms, as well as inelastic loss of the central atom. The scattering events $\chi(k)$ typically decay rapidly with increased wave number k, which leads to interpretation difficulties. Thus, greater values of $k^2\chi(k)$ or $k^3\chi(k)$ other than $\chi(k)$ are usually analyzed to highlight the contribution from the lower signal regions. Analysis of the spectrum of $k^2\chi(k)$ in terms of k is known as K-space analysis.

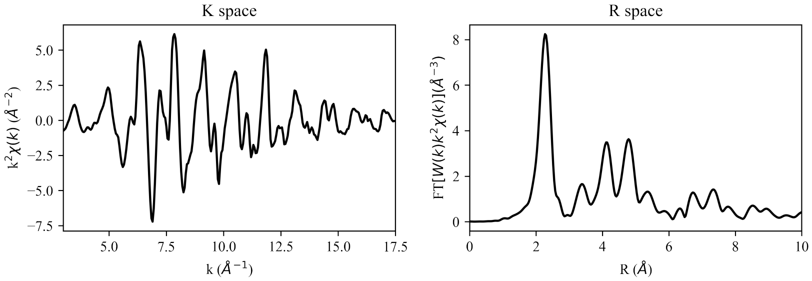

Fig. 2. EXAFS spectra of copper metal of K edge in K and R-space. The data was taken from XASLAB `https://xaslib.xrayabsorption.org/spectrum/91/`.

Fourier Transform can be applied on to K-space spectrum which becomes what is known as R-space EXAFS spectra. It is used to filter out the noise or analyze the contribution from each individual scattering wave. This is useful in highlighting the contributions of each scattering paths, and demonstrate thermal broadening and destructive interference. An example of K and R-space EXAFS spectra is shown below in Figure 2. The transformed spectra is:

$$\chi(r) = \frac{i\delta k}{\sqrt{\pi N_{max}}} \sum_{n=1}^{N_{max}} \chi(k)\Omega(k)k_n^w \exp(2i\pi n/N_{max}) \tag{3}$$

where δk is the spacing in the k-space spectrum, N_{max} is the array size, $\Omega(k)$ is the input window function, used to reduced oscillation from truncation of the input spectrum, and k_n^w is the k spectrum weights applied to the value.

3 Design and Implementation of GA Analysis Code

Conventional analysis of EXAFS can be very difficult depending on the number of scattering paths to consider. Each path is characterized with a set of four parameters: $[(S_0^2N_i)$, ΔE_0, σ_i^2, $R_i]$. We use ΔE_0 to correct energy mismatch of the absorption edge between experiment and theory. It should be identical for all scattering paths from the same calculation. Many software and methodologies have been developed for EXAFS analysis, such as `Larch` [5], `Demeter` [9], `WinXAS` [10]. However these tools require significant knowledge in condensed matter physics, and therefore are limited by the availability of experts.

For this reason, we have been developing an automated Materials Characterization Software package for EXAFS, XPS, and nanoindentation analysis. The code is GA-based for such a multi-objective optimization problem. We should point out that GA is not the first time applied to EXAFS spectrum [11]. However a comprehensive study (e.g., crossover and mutation options) of GA algorithms and their effects on uncovering the parameters for materials characterization analysis have not been studied. This is the goal of this paper.

GA is a heuristic optimization method inspired by Darwinian theory of evolution [12][13]. At the start of the algorithms, a population consisting of n temporary solutions (individuals) is generated randomly throughout the solution domains. Each solution is considered as a chromosome consisting of a number of parameters of interests and each parameter represents a gene of the chromosome. The GA evaluates the fitness of each solution in a population using a fitness function to determine the evolution of the next generation of solutions. To improve the accuracy of the final solution and the convergence rate of the involved iterations, a number of evolutionary inspired operators (e.g., crossover, mutation) are applied on each solution throughout subsequent generations. Since GA is also a stochastic process, it requires **multiple** runs of identical parameters to evaluate if the resulting solution reaches a global optimum. Compared to other search methods, GA operates by performing a multi-dimensional search, and encourages information exchanges among the different solutions.

In the following we will briefly describe how various operators effect the optimization process and the implementation applied for EXAFS analysis. The literature of [14] is listed for reference on general principles of GA.

3.1 Chromosome Representations of $[S_0^2, \Delta E_0, \sigma^2, \Delta R_i]$

The conventional GA cannot be applied directly but has to be customized for automated materials characterization analysis due to the reasons discussed in Section II. Unlike the conventional GA, where binary strings are used as individual chromosome in each individual, our software uses a floating point representation of a set of parameters as individual components: $[S_0^2, \Delta E_0, \sigma^2, \Delta R_i]$. S_0^2 other than $(S_0^2 N_i)$ is adopted because the degeneracy parameter N_i is usually assumed to be static and taken from the ideal structure under evaluation. In addition, all our parameters are constrained to valid physical ranges to prevent unreasonable values from being introduced, such as non-positive amplitude or Debye-Waller factors. When the number of "paths" (see Fig. 1) of interest are determined, e.g., n_{path}, each individual in evolutionary population will consists of **multiple** (e.g.,n_{path}) sets of parameters.

3.2 Crossover and Mutation

To allow convergence toward global optimal solution, best parental solutions are selected to be parents in the future populations, mirroring observation seen in nature. The selection process is based on the fitness values in the population. We employed a selection approach similar to rank selection, where each solution will be sorted based on their fitness values, from best performing to the least. A certain percentage of the best performing solutions will be selected to retain in the populations. To ensure biodiversity and reduce the risk of getting trapped in local extrema, a percentage of random solutions is generated as well. The main challenges regarding generating the optimial solution is the maintain balance between population diversity and selective pressure. A large population diversity leads to slow convergence on the optimum solution, while a low population diversity leads to premature convergence.

Crossover or *Recombination* is described as the operation of combining parental materials of two or more solutions during which the information is inherited and exchanged to produce a new solution. In nature most species have two parents but in GA the crossover operations can extend into more than two parents. Following the crossover operator, two individuals will be selected to generate subsequent individuals throughout the GA. There are numerous techniques for the crossover operator in the literature. For example, in the single-point cross over, the chromosomes of two parent solutions are swapped before and after a single point. In the double-point crossover, however, there are two cross over points and the chromosomes between the points are swapped only. We developed three crossover methods in our GA code and tested them for EXAFS analysis. The details of results will be discussed in Section IV.

Mutation operators modify existing solution by disturbing them by random chance. The mutation usually occurs in the “gene” level. For example, an offspring in binary representation of “0101” can become “0111” with a chance of mutation or the mutation rate. The exploration of mutation performed by the mutation operator can help to find the global optimal solution other than the local optimal solution. The mutation rate is usually set to be very low, but allows for perturbations in the ranges of values. Conventionally, the mutation operator remains fixed throughout the entire optimization process. This however can lead to ill-condition where the mutation operator is not sufficient to steer the solution out of a local extreme, which leads to premature convergence of the solution set. A self-adaptive mutation operator can solve premature convergence by increasing or decreasing the chance of mutation at each generation, which allows the algorithm to continuously refine the search area.

In our GA code, we implemented an algorithm based on the Rechenberg 1/5 success rule [15], which increases the mutation probability (σ) based on the “success ratio” S_i at the current generation. It is defined as the probability of generating an improved fitness value compared to the previous population. This probability can be further extended to the *Crossover* operator as well but was limited to the *Mutation* operator in this work. The mutation rate will increase in subsequent generation if S_i is greater than 1/5, and decrease if it less.

3.3 Fitness Calculation and Exit Conditions

The fitness calculation is used to compare the quality of individuals. Typically the fitness is computed every generation to ensure the solution is converging toward the optimal. However the calculation can be very expensive and therefore most approaches aim to minimize the number of fitness function calls. The fitness function employed for EXAFS is determined by computing the differences between the GA model and experimental data at each data point using χ^2 model:

$$\chi^2 = \frac{N_{indep}}{N} \sum_{i=1}^{N} \frac{\left(y_i^{\text{model}} - y_i^{\text{data}}\right)^2}{\epsilon_i^2} \tag{4}$$

y_i^{model} represents the model data of interests (e.g., $\chi(r)$) at i, y_i^{data} represents the experimental data, N_{indep} is the number of independent points, N is the total number of data points, and ϵ_i is the measure of uncertainty at each data point. Depending upon the collection methodology of the EXAFS data, the ratio N_{indep}/N can range from 1/10 when the EXAFS signal is over-sampled to speed data collection to 1 when the data is collected stepwise. In fact, this ratio can vary across the energy range of the collected EXAFS data. For other materials characterization analysis like XPS and nanoindentation, we can utilize similar fitness functions (e.g., Eqn. 4) but with the appropriate theoretical functions used to replace Eqn. 2 to construct the GA model. This modularized approach makes our GA-based code a good platform for expansion to other functions.

The principles of GA present the reasoning that an optimal solution can be reached after enough iterations. However, practical applications may deviate from this ideal case because it is possible that the optimal solution has not been found while iteration stops. So it is important to implement proper exit conditions to terminate the running of the code. Our code presents two method for determining when the algorithm reaches exiting conditions. The first one is to set a maximum number of iteration by users. The second method is more adaptive. When the solution doesn't improves for a number of generations, we believe the optimal is reached. In our software, we provided both methods to control the exit condition of the code.

3.4 Error Analysis

GA is a probabilistic optimization method where the solution can't be expected to be identical over repeated runs. It is **critical** to perform a set of individual, independent runs (e.g., 50) with varying conditions (e.g., population size, generation number, mutation rate) to evaluate the accuracy of the solution and determine the range of errors. In our GA code, we use the Global Random Analysis method [16] to estimate errors of those measured parameters. The error of each individual parameter (e.g., $S_0^2, \Delta E_0, \sigma^2, \Delta R_i$) in the simplest of X-ray absorption cases can be quantified using a Poisson distribution and the standard deviation is roughly equal to the square root of the absorption duration time [17]. It is worth noting that this quantification can only be used when the sample is uniform in composition and thickness, and cannot be applied to heterogeneous sample or sample which has been irradiation damaged. However, this limitation doesn't apply to our error analysis because our data sets are uniform.

During each generation, the individuals of the best fitness value will be stored and used to construct the covariance matrix which contains the error for each parameter in the solution. Random perturbations of three parameters are selected in a bounded range: population size (100-5000), total number of generations (10-50), and mutation rate (0%-100%). For example, one individual run of 50 runs could randomly select a condition in the bounded ranges like population size 200, generation number 30, mutation rate 20% and other runs may select other conditions. The combined results will be utilized for global random analysis.

4 Experiment Results

The majority of the code is implemented and written in Python. All our experiments were conducted on Idaho National Laboratory High Performance Computing (HPC) cluster 'Sawtooth' which is comprised of Xeon Platinum 8268 Processors. Each experiment were repeated 100 times to compute the standard deviations and error matrix, and for each experiment we requested four CPU cores and 12.0 GB of memory. To validate our GA code before applying it to analysis of real experimental EXAFS data, we first applied the code to a set of synthetic data, which is generated with five scattering paths. Gaussian noise with a signal to noise ratio of 20 was added to the synthetic spectra.

Table 1. GA fitted parameters and errors from the synthetic data set.

Path	N	S_0^2		ΔE_0 (eV)		σ^2 (Å^2)		ΔR (Å)	
#		Model	True	Model	True	Model	True	Model	True
1	12	0.62±0.03	0.62	-0.3±0.7	-0.91	0.0041±0.0004	0.004	0.0512±0.003	0.05
2	6	0.72±0.08	0.66	-0.3±0.7	-0.91	0.0015±0.0007	0.001	0.0118±0.005	0.01
3	48	0.5±0.2	0.74	-0.3±0.7	-0.91	0.0102±0.0044	0.014	0.0615±0.032	0.08
4	48	0.3±0.2	0.45	-0.3±0.7	-0.91	0.0068±0.0043	0.009	0.0467±0.043	0.00
5	24	0.22±0.08	0.14	-0.3±0.7	-0.91	0.0081±0.0024	0.005	0.0538±0.012	0.05

Tab. 1 shows a set of GA fitted parameters with the corresponding errors generated from the synthetic data. The true values are also listed for comparison. Since the scattering with single path has the shortest traveling distance, it has the largest contribution to the absorption spectrum and is of particular interest. It can be seen from the first row (Path 1), all of the fitted parameters: S_0^2, E_0, σ^2, and δ_R match extremely well to the true values, indicating a high confidence of accuracy using our GA model. For other scattering paths which travels further, the errors are presented larger but are in a satisfactory level to our fitting because their contribution to the fitting spectra is lower.

Fig. 3 shows the GA fitted spectra in both K-space and R-space in comparison with the true synthetic data. The included Gaussian noise is selected much higher than that usually could be seen from real experimental EXAFS data to demonstrate the performance of our GA code under extreme conditions. It

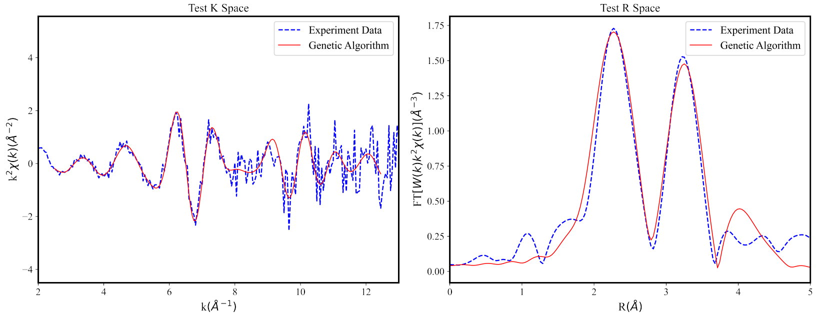

Fig. 3. EXAFS spectra fits of the synthetic experimental spectra with added Gaussian noise. Top: Spectra were fitted in K-space over the region from 2.5 to 12.5 $\AA^{-1}$. Bottom: Spectra in R space in a range from 0 to 5 $\AA$.

can be seen that the GA code was able to obtain good matches to true values, especially in lower k ranger, which contribute to the fitted parameters the most.

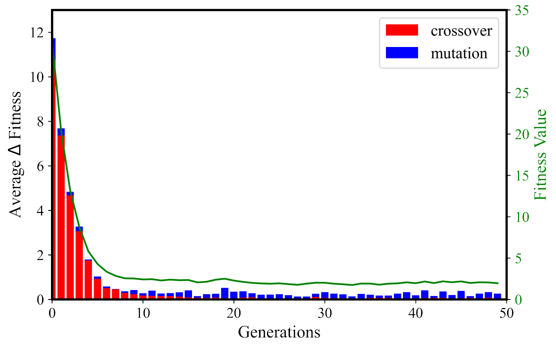

Fig. 4. The effect of the crossover (red) and mutation (blue) operators on the overall calculation of fitness value.

Fig. 4 shows the effects of varying both the mutation and crossover operators, to explore their contributions to the change of the fitness value. We can observe the average fitness value significantly decreases (i.e., better) along the evolution of generations. The crossover operator contributes most to the improvement of the fitness value in early stages of the evolved generation, which can be seen from the significant heights of red bars in Fig. 4. The mutation operator contributes less in general compared to the crossover operator in early stages and even zero in a few cases. However, the mutation operator contribution dominates at the late stages of the evolution. This is because the mutation probability increases as the generations evolve under the Rechenberg algorithm, which leads to an increase in exploration area where mutation can occur. It is also used to validate if the solution is in global optimal.

Next, we will employ three metrics to evaluate the quality of our fits in both K and R spaces: Coefficient of Variation (R2), Mean Absolute Error (MAE), and Root Mean Square Error (RMSE) [18]. R2 measures the residuals of spectra, MAE measures errors within the same unit spectrum but gives similar weights to outliers, and RSME is used to amplitude errors from outliers.

4.1 Crossover Analysis and Mutation Analysis

We provided three crossover methods in our GA code. Users can examine their effects and select the best option that fits their specific problems. The first method "uniform random crossover" utilizes uniform crossover where each gene is selected randomly from either parent with equal probability. The second method "AND crossover" utilizes a mixing rule where each gene in child is a result of an AND logic operation on to each gene from its parents. The third method"OR crossover" utilizes the similar mixing rule using OR logic operator.

Table 2. Error analysis for three crossover and three mutations methods.

	Crossover			Mutation		
Method	1(uniform random)	2(AND)	3(OR)	1(maximum)	2(nested)	3(matropolis)
R2-K	0.99856	0.95530	0.94321	0.99900	0.98625	0.99920
R2-R	0.99661	0.98541	0.97431	0.99632	0.99163	0.99656
MAE-K	0.03912	0.17298	0.03345	0.03389	0.11349	0.03047
MAE-R	0.03455	0.04561	0.04733	0.03700	0.04119	0.03611
RMSE-K	0.00231	0.05333	0.00255	0.00175	0.05250	0.00142
RMSE-R	0.00211	0.01044	0.00355	0.00264	0.00604	0.00247

Tab. 2 shows that the first method returns the highest contribution to the true value in both K and R spaces. The second and third methods may produce lower accuracy than the uniform random crossover in our application. This is because our solution set only contains a limited number of four parameters, therefore it is ideal to use uniform random crossover to promote information exchange and diversity as much as possible between individuals. For this reason, we adopted the uniform random crossover in the following experiments.

We also provided three mutation methods in our GA code for users. They all utilize the same starting mutation probability at each generation σ_i and compare it with some random number to determine if a new mutation is necessary. The first method "maximum mutation" generates a complete new individual in units of parameter if a random generated number reaches the mutation probability. It is a simple method and maximizes the number of possible mutations in the population, and ensure sufficient genetic diversity to be introduced to the population. The second method "nested mutation" introduces a secondary random number to control the actual mutation. The new individual can only be generated in units of gene when the both random numbers are less than the mutation rate at each generation. This method fits a more traditional GA where the actual mutation rate for each gene is typical set very low (e.g., 1 to 5%).

Algorithm 1 Mutation method 2: Nested Mutation. Note: Removing lines 4,5,6,8,9 can lead to Mutation method 1: Maximum Mutation.

```
Input: Mutation Rate $\sigma$, Individual $P_i$
Output: Individual $P_i$
 1: for $P_i$ in Populations ($P$) do
 2:     Generate a number $x$ in [0..100]
 3:     if $x < \sigma$ then
 4:         for number of variables in each scattering paths do
 5:             Generate a number $y$ in [0..100]
 6:             if $y < \sigma$ then
 7:                 Generate new individual
 8:             end if
 9:         end for
10:     end if
11: end for
```

Algorithm 2 Mutation method 3: Metropolis Mutation

```
Input: Mutation Rate $\sigma$, Individual $P_i$, Fitness $f_i^{orig}$
Output: Individual $P_i$
for $P_i$ in Populations ($P$) do
    Generate a random $x$ in [0..100]
    if $x < \sigma$ then
        Mutate the Scattering Paths
        Calculate New Fitness $f_i^{mut}$
        if $f_i^{mut} < f_i^{orig}$ then
            Accept the Mutation
        else if $\exp(-(f_{\mathrm{mut}} - f_{\mathrm{orig}})/K(i)) < t$ then
            Accept the Mutation
        else
            Reject the Mutation
        end if
    end if
end for
```

The third method "metropolis mutation" is modeled after the Metropolis-Hasting algorithm [19], which rejects the mutation if the fitness value is less than current fitness value, or accepts it if the following criterion is satisfied: $\exp(-(f_i^{\mathrm{mut}} - f_i^{\mathrm{orig}})/K(i)) < t$. The f_i^{mut} and f_i^{orig} are fitness values after and before the mutation, t is a random uniform value $\in [0,1]$, and where $K_iK(i) = -\delta_f / \ln(1 - i/i_{max})$ is a parameter called the cooling rate. The δ_f is the absolute difference in fitness value between subsequent generations, i is the current generation number, and i_{max} is the maximum generation. This cooling rate allows the probability to accept parameters outside of the predefined range, while assuming that the mutation rate will decrease at a linear rate.

Table 2 compares the different mutation methods. We can see that the value of R2 for mutation method 1 and method 3 is very similar, but the fitness value for the method 2 performed the worst. This indicates that analyzing a relatively small parameter set in EXAFS application should take a large mutation rate to enhance the mutation possibilities and ensure the population diversity.

4.2 Cut off Analysis for Selecting Scattering Paths

One of the main difficulties in obtaining an accurate fit using GA for EXAFS analysis lies in determining the number of scattering paths that may potentially be observed in an EXAFS measurement and selecting from that set the actual paths required to replicate the experimental results. The list of most significant paths can be difficult to obtain due to the effectively infinite number of potential combinations of paths. On the other hand, there's no universally optimized fixed set of paths for all applications and it is important to deselect insignificant scattering paths with low contribution to the spectra. The analysis tools must be able to handle these cases.

We have developed a process to analyze the contribution from potential scattering paths and uncover the scattering paths with the most contribution to the spectra. We first calculated the integrated area below the spectrum curve and the contribution from each individual path. We selected the paths that contribute more than the pre-defined cut off percentage of these areas (e.g., 1%), and obtained a list of significant paths for further analysis.

Fig. 5 shows χ^2 as a function of the cut off percentage. We analyzed four data sets of Copper metal EXAFS spectra at various temperatures of 10K, 50K, 150K and 298K to test the selecting method of cut off area percentage. These data were obtained from XASLAB. The initial number of scattering paths that we employed was 42 which represented a full set of scattering paths with distances from 2.5527 Å to 7.6580 Å. These paths were used to fit the experimental data over the K-space range from 3 $\AA^{-1}$ to 17.0 $\AA^{-1}$. After performing the cut off calculation, a subsequent optimized set of paths was obtained and applied to a second round of calculation that uses only the optimized scattering path list and excludes all paths with insignificant contributions to the measured spectrum.

The results of utilizing the cut off can be evaluated by using the final fitness value score (i.e., χ^2). However, we must strike a balance between the number of scattering paths and accuracy of the final fitness value. We tested our algorithms by placing various cut off percentages to observe the effects on their final average fitness value. We selected seven different percentage from 10%, 5%, 2%, 1%, 0.67%, 0.5% and 0.3%.

We can see that when the cut off percentage decreases from 10% to 1%, the average χ^2 score decreases (means better) for all of temperatures, which is good as we expected. When the cut off percentage goes below 1%, χ^2 score tends to increase due to the more paths that were included in the calculation, which makes insignificant changes to the overall fitness score. In our experiments, the best cut-off ratio in practice was found to be 1%, although users can select other cut off values for their applications.

4.3 Computational Performance

It is worth noting that there's no emerging need to parallelize our GA code since the same data set must be run/analyzed multiple times (e.g., 100) to gauge the errors compared to the experiments. However, parallelization is applicable if the number of IO operations is reduced significantly. In our current GA code, evolutionary operators perform very frequent IO operations.

We measured the scalability of our GA code in terms of the number of scattering paths. There are two major factors which can affect the complexity of the computation. The first is the number of sample points in the spectra, which is usually determined by the experimental conditions of the measurement, or instrumentation setting. The second is the number of scattering paths selected by users to represent the exploration range of atomic structure of interests. Fig. 6 shows the average time spent per generation as a function of the number of selected paths. We can see that the algorithms scales very well, with a time complexity $\mathcal{O}(n)$.

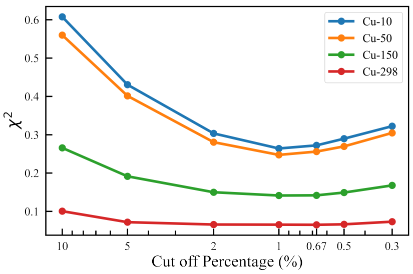

Fig. 5. χ^2 as a function of the cut off percentage for Cu foil EXAFS Spectra in K-edge at various temperatures.

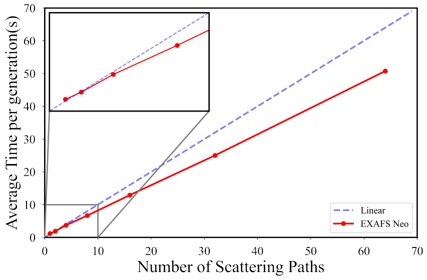

Fig. 6. The complexity increases as the number of scattering paths increases.

5 Conclusion

We have developed a GA based software with the capabilities for efficiently performing automated materials characterization analysis of both complex X-ray spectra and nanoindentation data. We provided multiple crossover and mutation options in the code from which users can choose to optimize the analysis for their specific materials systems. We have extensively tested our software under various synthetic and experimental conditions. Our results demonstrated good scores of fitting without human inputs. We also tested the code with various cut off percentage to obtain the best scattering paths set and remove possible noises. We note that caution must be exercised in selecting the data set over which to perform the cut off analysis. It must be representative of the entire collection of data to be analyzed for the results to be meaningful. The extensibility of our code base is a major advantage. Adding new analysis techniques does not require any new debugging of the core module of the GA code.

6 Acknowledgments

This work was supported by author ML's startup fund. It is also supported through Idaho National Laboratory (INL) Laboratory Directed Research and Development (LDRD) Program under DOE Idaho Operations Office Contract DE-AC07-05ID14517. This research made use of the resources of the High Performance Computing Center at INL, which is supported by the Office of Nuclear Energy of the U.S. Department of Energy and the Nuclear Science User Facilities under Contract No. DE-AC07-05ID14517.

References

[1] B. Blaiszik, K. Chard, R. Chard, I. Foster, and L. Ward, "Data automation at light sources," vol. 2054, Jan. 2019, p. 020 003.

[2] G. H. Major, T. G. Avval, B. Moeini, and et al., "Assessment of the frequency and nature of erroneous x-ray photoelectron spectroscopy analyses in the scientific literature," *Journal of Vacuum Science & Technology A*, vol. 38, no. 6, p. 061 204, 2020.

[3] J. Terry, M. L. Lau, J. Sun, and et al., "Analysis of extended x-ray absorption fine structure (exafs) data using artificial intelligence techniques," *Applied Surface Science*, vol. 547, p. 149 059, 2021, ISSN: 0169-4332.

[4] *EXAFS Neo*, https://github.com/laumiulun/EXAFS-Neo-Public.

[5] M. Newville, "Larch: An analysis package for xafs and related spectroscopies," in *J Phys Conf Ser*, vol. 430, 2013, p. 012 007.

[6] J. J. Rehr, J. J. Kas, F. D. Vila, M. P. Prange, and K. Jorissen, "Parameter-free calculations of x-ray spectra with FEFF9," *Physical Chemistry Chemical Physics*, vol. 12, no. 21, p. 5503, 2010.

[7] G. Bunker, *Introduction to XAFS*. Cambridge University Press, 2009.

[8] D. E. Sayers, E. A. Stern, and F. W. Lytle, "New technique for investigating noncrystalline structures: Fourier analysis of the extended x-ray absorption fine structure," *Phys. Rev. Lett.*, vol. 27, no. 18, p. 1204, 1971.

[9] B. Ravel and M. Newville, "ATHENA, ARTEMIS, HEPHAESTUS: Data analysis for x-ray absorption spectroscopy using IFEFFIT," *Journal of Synchrotron Radiation*, vol. 12, no. 4, pp. 537–541, Jun. 2005.

[10] *WinXAS v3.2*, http://www.winxas.de, Retrieved: 2020-10-30.

[11] G. Bunker, N. Dimakis, and G. Khelashvili, "New methods for exafs analysis in structural genomics," *Journal of Synchrotron Radiation*, vol. 12, no. 1, pp. 53–56, 2005.

[12] J. H. Holland, "Genetic algorithms," *Scientific American*, vol. 267, no. 1, pp. 66–73, 1992.

[13] D. E. Goldberg and J. H. Holland, "Genetic algorithms and machine learning," *Machine Learning*, vol. 3, no. 2, pp. 95–99, 1988.

[14] A. E. Eiben, J. E. Smith, *et al.*, *Introduction to Evolutionary Computing*. Springer, 2003, vol. 53.

[15] I. Rechenberg, "Evolutionsstrategie'94," in *Werkstatt Bionik und Evolutionstechnik*, 1994.

[16] D. R. Redhouse, "Uncertainty quantification of a genetic algorithm for neutron energy spectrum adjustment," Ph.D. dissertation, 2017.

[17] P. L. Meyer, *Introductory probability and statistical applications*. Oxford and IBH Publishing, 1965.

[18] P. R. Bevington, D. K. Robinson, J. M. Blair, A. J. Mallinckrodt, and S. McKay, "Data reduction and error analysis for the physical sciences," *Computers in Physics*, vol. 7, no. 4, pp. 415–416, 1993.

[19] S. Chib and E. Greenberg, "Understanding the metropolis-hastings algorithm," *The American Statistician*, vol. 49, no. 4, pp. 327–335, 1995.