Modeling Metal Protein Complexes from Experimental Extended X-ray Absorption Fine Structure using Computational Intelligence

Collin Price

Department of Computer Science

Submitted in partial fulfillment
of the requirements for the degree of

Master of Science

Faculty of Mathematics and Science, Brock University
St. Catharines, Ontario

Abstract

Experimental Extended X-ray Absorption Fine Structure (EXAFS) spectra carry information about the chemical structure of metal protein complexes. However, predicting the structure of such complexes from EXAFS spectra is not a simple task. Currently methods such as Monte Carlo Optimization or simulated annealing are used in structure refinement of EXAFS. These methods have proved somewhat successful in structure refinement but have not been successful in finding the global minima. Evolutionary algorithms have had success in overcoming local minima issues in other domains. We propose the use of three different approaches to better predict the structure of metal protein complexes: genetic algorithm (GA), particle swarm optimization (PSO), and differential evolution (DE).

Acknowledgements

Here I thank all the people!

C.P.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The aim of this thesis is to find a better method for determining the atomic structure of a molecule using Extended X-Ray Absorption Fine Structure (EXAFS). The thesis uses the oxygen-evolving complex (OEC) in state S_1 as an example for structure refinement but the developed process can be applied to any given chemical structure that has undergone x-ray absorption spectroscopy experimentation. In this chapter, we introduce the biological background and terms, followed by the problem definition, and finally elaborate on the computer science theories applied to the problem.

1.1 Biological Background

Photosystem II [1] is the protein complex responsible for the first stage of photosynthesis. Photosynthesis is a process used by plants and other organisms to convert light (photons) into energy. Photons, that are captured from the Sun or other light sources, and water are processed through a water-oxidizing enzyme known as the oxygen-evolving complex (OEC) [2]. The water molecule (H_2O) is split into two parts, O_2 and H^+. The O_2 is released from the system, and the H^+ will be stored and used as a source of energy.

The OEC complex performs oxidation on two water molecules through a series of intermediary states. The "S-State Cycle" [2] consists of 5 states: S_0, S_1, S_2, S_3, and S_4. During the transistion between each state a hydrogen electron is released.

After S_4 concludes O_2 is formed. For the purpose of this work the resting or so-called storage state S_1 will be analysed. The atomic structure of the OEC molecule is altered between each state.

The most significant feature of this compound is its inorganic core, which is $Mn_4Ca_1O_xCl_{1-2}(HCO_3)_y$ It is not found anywhere else in Biology and offers the only biological blueprint for water splitting. By studying OEC the hope is to understand how the oxidation of water can occur at such a low energy cost. Aquiring a better understanding of how the water splitting process occurs will assist in creating biomimetic catalysts or engineered PSII enzymes for real world applications.

1.2 X-ray Absorption Spectroscopy

The following overview is based on information contained in Matthew Newvilles Fundamentals of XAFS (2004) [3]. X-Ray absorption fine structure (XAFS) is a method used to measure the absorption coefficient of a material as a function of energy. X-rays are part of the electromagnetic spectrum with wavelengths ranging from 25Å to 0.25Å. All atoms resonate at a specific wavelength. The x-ray is tuned to have the same wavelength as the target atom. A photon from an x-ray is absorbed by an electron in a tightly bound quantum core level of an atom. Absorption only takes place if the binding energy of the core level is less than the energy of the x-ray photon. At the time of absorption a core electron moves to an empty outer shell and another electron moves in to take its place. Eventually the affected electrons decay to their original state. During this time fluorescence energies are emitted that characterize a specific atom.

The absorption coefficients measured after the initial absorption are referred to as the Extended X-ray Absorption Fine Structure(EXAFS). During the decay of the electrons to their original state, oscillations occur in the measure of the absorption coefficient. The different frequencies found within the oscillations correspond to different near-neighbour coordination shells, which can be described and modeled according to the EXAFS equation. From the oscillations, the number of neighbouring atoms, the distances to the neighbouring atoms, and the disorder in the neighbour distances can be determined. The energy spectra for OEC in S_1 is shown in Figure 1.1.

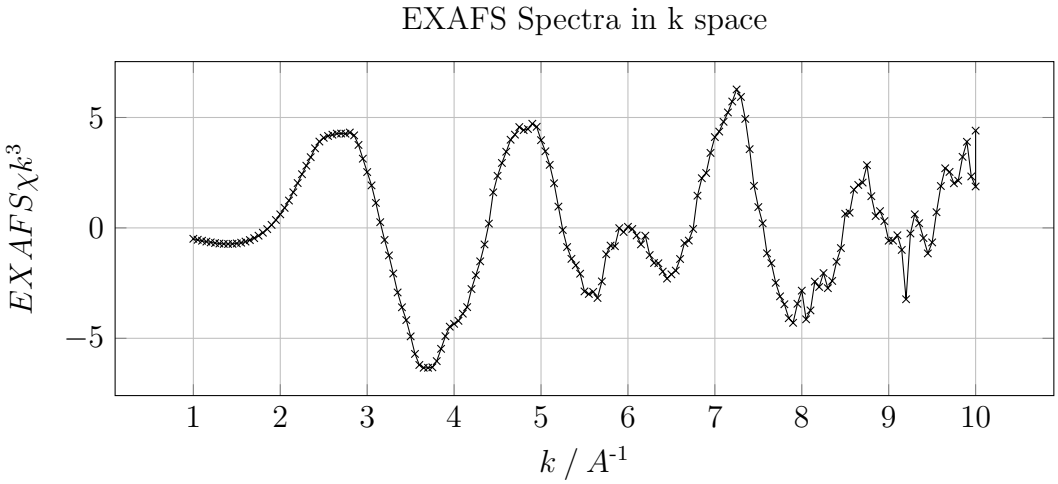

Figure 1.1: EXAFS Spectra of OEC in S_1

1.3 Force Fields

The atoms within a molecule are consistently interacting with each other. Atoms can directly and indirectly interact with neighbouring atoms. Atoms directly interact with neighbouring atoms with a bond or indirectly through van der Waals forces. Calculating the forces involved within the molecule would require a large amount of computing power to attain a high degree of accuracy. Instead simplier, classical formulas are used to calculate the energy within the system. There are several different formulas for calculating classical force fields. This work will utilize Assisted Model Building with Energy Refinement (AMBER) [4] force fields for the energy calculations. AMBER force fields are widely used with proteins and related systems [5]. Equation 1.1 shows the formula used when calculating the energy of a system using AMBER force fields.

$$\begin{aligned} V(r^N) = & \sum_{\text{bonds}} k_b(l-l_0)^2 \\ & + \sum_{\text{angles}} k_a(\theta-\theta_0)^2 \\ & + \sum_{\text{torsions}} \sum_{n} \frac{1}{2} V_n[1+\cos(n\omega-\gamma)] \\ & + \sum_{j=1}^{N-1} \sum_{i=j+1}^{N} f_{ij} \left\{ \epsilon_{ij} \left[\left(\frac{r_{0ij}}{r_{ij}} \right)^{12} - 2 \left(\frac{r_{0ij}}{r_{ij}} \right)^{6} \right] + \frac{q_i q_j}{4\pi\epsilon_0 r_{ij}} \right\} \end{aligned} \tag{1.1}$$

1.4 Problem Definition: Structure Refinement Problem

The goal of this thesis is to examine different search heuristics to determine the best method of finding the theoretical atomic structure of a molecule using the molecule's EXAFS spectrum for comparison. This problem contains two important but unrelated goals. Firstly, the algorithm must be able to find an atomic structure whose EXAFS spectrum matches the experimental EXAFS spectrum, and secondly, creating an atomic structure whose potential energy is a low as possible.

EXAFS can be used to identify properties of a molecule, but they do not provide enough detail to determine the atomic structure of a molecule in 3-dimensional space. An EXAFS spectrum allows you to identify how far apart atoms are from each other, but does not give enough information to identify their dihedral angles. Fortunately, EXAFS can be used to assist in determining the atomic structure of a molecule. The energy spectrum given off by the molecule is unique to its structure, meaning that you can create an atomic structure, obtain its EXAFS spectrum, and compare the results. The hope is that if you create an atomic structure whose EXAFS spectrum closely matches the EXAFS spectrum of an actual model, then there is a high likelihood that the created structure will closely match the actual structure.

Using EXAFS spectrum comparison the goal is to obtain a set of candidate atomic structures. Atomic structures that generate similar EXAFS spectra may have different geometries. An expert will have to analyse the candidate solutions to determine if any of these atomic structures are actually chemically infeasible. **Having a set of candidate solutions will improve the odds of finding the actual solution.**

The IFEFFIT XAFS data analysis suite [6] is used to simulate the EXAFS experiments. This suite includes two applications that will be used: FEFF6, and IFEFFIT. FEFF6 is used to simulate an XAFS experiment and IFEFFIT does post processing of the simulated EXAFS spectra. During the atomic structure refinement, the generated atomic structures will be run through these applications to obtain an EXAFS spectrum.

NAMD [7] will be used for the energy calculations. The NAMD Energy Plugin [8] will calculate the potential energy of the generated atomic structure.

1.5 Thesis Organization

The remainder of this thesis is organized as follows.

Chapter 2

Background

The purpose of this chapter is to assist the reader in understanding the search techniques used in this work. Section 2.1 will explain the framework used for the algorithms described in Sections 2.2, 2.3, 2.4, and 2.5.

2.1 Evolutionary Algorithms

An evolutionary algorithm (EA) is a population-based metaheuristic optimization algorithm. An evolutionary algorithm is a search heuristic that is based on Darwin's theory of natural evolution. Darwin theorized that over a period of time a population of individuals would naturally mate and create offspring that were better than themselves. He suggested that not all individuals are created equally and that eventually the weaker individuals would die off. This same principle can be applied to a search algorithm as a heuristic. An EA contains a population of individuals that are evolved to find improved candidate solutions.

Figure 2.1 demonstrates how the basic EA operates. Initially a *population* of candidate solutions is generated. The individuals are evaluated based on an *evaluation function* and are checked against the *stopping criterion*. If the *stopping criterion* has not been reached the population goes through an *evolutionary period* where a new population of candidate individuals are created from the last population. This iterative process, also called a *generation*, is repeated until the *stopping criterion* is

reached. The following subsections will explain each of these parts.

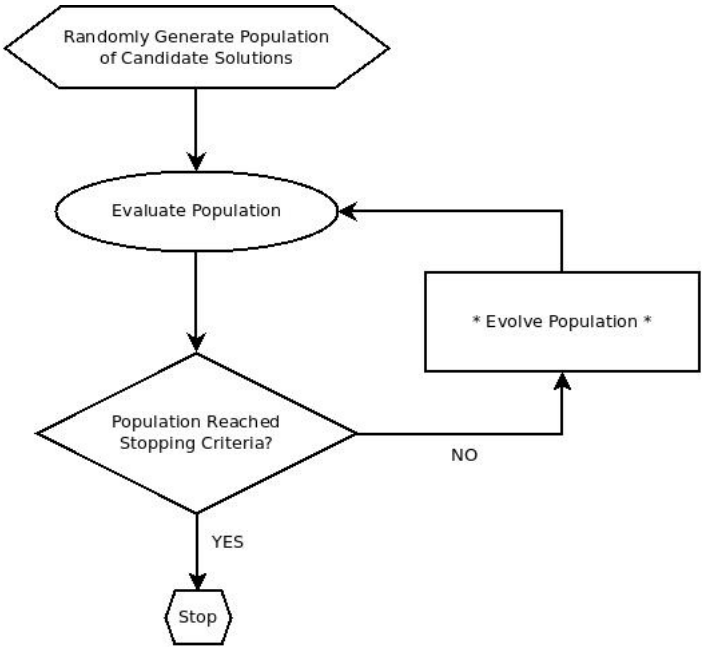

Figure 2.1: Population Individual Modification

2.1.1 Population

The population is a key piece to an EA. Each individual in the population represents a possible candidate solution to the problem that we are attemping to solve. The representation of the individual is usually unique to the problem. Generating the initial population can either be done randomly or by some procedural method. The goal of generating the initial population is to create a diverse enough population to find improved solutions.

2.1.2 Evaluation Function

This operator determines the fitness of an individual. Each individual is evaluated and given a fitness score to represent how well the individual performed on the problem. This operation is problem specific and it can be very difficult to determine how a problem should be evaluated. The evaluation function is important for differentiating

individuals. A poor evaluation function can make each of the individuals appear to be similar when they actually have small key differences.

2.1.3 Stopping Criterion

Stopping criteria are used to determine when the EA should stop evolving. There are generally three ways stopping criteria can be reached: a maximum number of iterations is reached, the population has converged on the same solution, or the solution has been found.

2.1.4 Evolving the Population

The evolutionary process of an EA is what differs in each implementation of an EA. Each algorithm has a different interpretation of how the population should be evolved. Evolving the population consists of using the individuals in the population to create a new population. Sections 2.2, 2.3, 2.4, and 2.5 provide detail on the different forms of evolution.

2.2 Genetic Algorithm

A genetic algorithm (GA) is a search heuristic that mimics natural selection. In a GA individuals go through a *selection* process and are bred(*crossed*) with other individuals to create new individuals. Individuals in a GA are commonly refered to as *chromosomes*. Figure 2.2 depicts how evolution occurs in a GA.

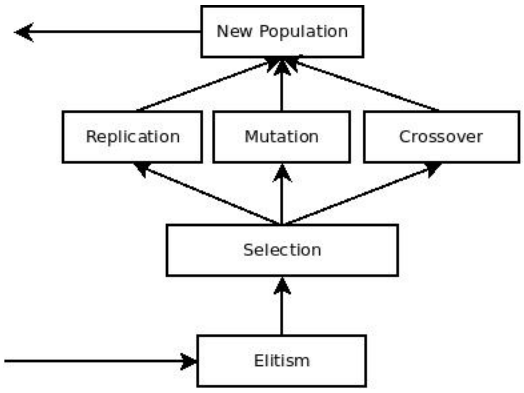

Figure 2.2: GA Evolution

During each generation a new population is created using the previous generation's population. Initially the best individuals from the previous population might be copied directly into the new population using an operator called *elitism*. To obtain the remaining individuals needed to fill the new population a *selection* process occurs. Two individuals are chosen using a *selection* method and then one of three options can occur: *crossover*, *mutation*, or *replication*. Crossover mixes two individuals together to create two new individuals, mutation randomly modifies each individual individually, and replication copies the individuals. The two individuals are then placed in the new population and the process is repeated until the new population is the same size as the previous population. Figure 2.2 depicts how evolution occurs in a GA. See Subsection 2.2.2 for more details on the operators discussed.

2.2.1 Chromosome

The individuals of a GA represent possible candidate solutions to the problem. Typically a chromosome is represented as an array where each index of the array represents a property of the candidate solution. There are no restrictions to the encoding of a chromosome but each property of the chromosome must be independant from the others. The simplest example of a representation is a binary array of 1's and 0's, as shown in Figure 2.2.1. In the sample chromosomes provided the 1's and 0's might represent whether a feature is enabled or disabled, 1 being enabled and 0 being disabled,

in the candidate solution.

0	1	1	0	1	0	1	0

Figure 2.3: Simple Chromosome Represention

2.2.2 Genetic Operators

Each of the following operators represent a piece of a genetic algorithm. They facilitate the evolutionary process in the effort to find better candidate solutions. Each of these operators has a unique purpose in the search algorthm but there are many different ways in which these goals can be carried out. Only a few of the different methods will be described in this work.

Selection Operator: This operator is very important to the *Crossover* and *Mutation* operators. The idea behind this operator is to put selection pressure on the population during the evolutionary process. Individuals with a better fitness score should be allowed a better chance of breeding to create the next population. During the selection process two individuals are chosen for breeding or reproduction. There are several varieties of selection methods but only *k-Tournament selection* will be explained as this is the method used in the algorithm in **(chapter goes here)**.

The k-Tournament selection method works by randomly selecting k individuals from the population, where k is less than the number of individuals in the population, and selecting the individual that has the best fitness score from the k individuals. The value of k should be relatively small compared to the size of the population. If the value of k is too large it would defeat the purpose of this selection method. For example, if there is a population size of 100 a suitable value of k is around 2-5.

Crossover Operator: This operator is essential to evolving the individuals of the population. Crossover is the mechanism by which two individuals breed to create two new individuals. With respect to the evolutionary process, crossover exploits the current information that is contained within the population in order to find improved individuals. The most widely used type of crossover is N-point crossover.

N-point crossover works by randomly selecting N cutting points and swapping the information between the two individuals along those N points. Figure 2.4 demonstrates

how the swapping of information occurs during 2-point crossover.

Parent 1	0	1	1	0	1	0	1	0
Parent 2	0	0	1	0	0	1	0	0

Child 1	0	0	1	0	1	1	0	0
Child 2	0	1	1	0	0	0	1	0

Figure 2.4: 2-Point Crossover

Mutation Operator: The mutation operator is used to introduce random changes to the individuals during evolution. Mutations to individuals are a way to explore the search space. Depending on how the initial population was created there may not be the necessary information in the population to find the optimal solution with crossover alone. Mutations allow for new information to possibly be introduced into the population. A common type of mutation is single-point mutation where a single index in your individual is modified. Figure 2.5 demonstrates single-point mutation.

Individual	0	1	1	0	1	0	1	0

Mutant	0	1	1	1	1	0	1	0

Figure 2.5: Single-point Mutation

Elitism Operator: During each generation of the genetic algorithm a new population is created using the individuals from the population in the previous generation. The new population is bred from the previous individuals with the hopes of creating better individuals. Sometimes this is not the case and the population can end up losing valuable information from individuals that were not chosen during the selection process. To prevent this from happening the elitism operator was create. The elitism operator works by seeding the next generations population with the individuals with the best fitness score. Typically only the top 1% of individuals are copied into the next generation.

2.3 Recentering Genetic Algorithm

The recentering genetic algorithm (RGA) is a variation of the recentering-restarting genetic algorithm (RRGA) [9] [10] which has had success in avoiding local minima. The RRGA is used to avoid fixating on local optima. RRGA works by performing a series of standard GA runs. Each run uses the final population from the previous run as its starting population with some adjustments. At the beginning of a run the RRGA selects a center, which is a possible candidate solution to the problem, and at the end of each basic GA run the center is compared to the best individual in the population. If the best individual is better than the current center it is replaced with the best individual and the whole process is repeated. The center is used as a baseline for generating the population in the next run.

The RGA works similarly to the RRGA but there is no center for the population. Instead a basic GA is allowed to run until the population's fitness scores begins to converge. After the population has converged upon a minimum diversity, new individuals are introduced to the population. Duplicate individuals are removed from the population and new individuals that have not yet been in any population take their place. For example, if there is a population size of 100 and the convergence rate is 5% then after all the duplicates are removed there will only be 5 individuals remaining and 95 new individuals will be inserted into the population. Algorithm 1 shows the pseudo-code of the restarting method.

Algorithm 1 Restarting the population

```
if population has converged to minimum diversity then
    remove all duplicate individuals;
    while population not full do
        insert random draw from generated individuals into population;
    end while
end if
```

2.4 Particle Swarm Optimization

Particle swarm optimization (PSO) [11] [12] is a population based search metaheuristic designed to iteratively optimize a problem. PSO is well suited for problems con-

taining a nonlinear and non-differentialable continuous search space. The candidate solutions found within a PSO are known as *particles*. Each of these particles represents a candidate solutions position within the search space. The particles positions are updated to move around the search space based on a mathematical formula. The process of how a particle's position is updated is detailed in Subsection 2.4.3. During the *evolution* of the population each particle is updated once.

2.4.1 Particle

Each particle represents a candidate solution to the problem. An individual particle contains both a position (p), and a velocity (v). The position and velocity are each a vector of real numbers where the size of the vector depends on the problem. The position represents a possible solution to the problem.

Each particle also contains an archive of its personal best position ($pBest$). After a particle's position is updated based on the mathematical formula described in Subsection 2.4.3, it's fitness score (See Subsection 2.1.2) is updated. The new fitness score is compared with the fitness score of the current best position's fitness score. If the new position's fitness score is better than the current best position's fitness score, the new position becomes the current best position.

2.4.2 Global Best Position

The global best position ($gBest$) is the particle position that has produced the best fitness score. The $gBest$ is updated at the end of each iteration of the population.

2.4.3 Particle Update

Each *particle's* position vector is updated based on their current velocity vector. The velocity vector of each particle position is updated each generation based on the formula show in Equation 2.1, where $r_p, r_g \sim U(0,1)$, and ω, ϕ and φ are user defined. ω (inertia) controls the efficiency of the particle moving through the search space. ϕ (social) and φ (cognitive) control the magnitude of the force pulling the

particle towards the *pBest* and *gBest*. The particle's position vector is updated based on the particle's new velocity vector (See Equation 2.2).

$$v = \omega v + \phi r_p(pBest - p) + \varphi r_g(gBest - p) \tag{2.1}$$

$$p = p + v \tag{2.2}$$

2.5 Differential Evolution

Differential evolution (DE) [13] is a population based search metaheuristic designed to iteratively improve candidate soltions to a problem. DE is well suited for problems containing a nonlinear and non-differentialable continuous search space. DE works by creating a new candidate solution using existing candidate solutions in the population using a *mutation* operator. Once a new candidate solution is creating the fitness scores of each solution are compared and the candidate with the better fitness score is put into the new population. Candidate solutions in DE are refered to as *agents*. Figure 2.6 depicts how evolution occurs in DE and Subsections 2.5.2 and 2.5.3 describe the operators used.

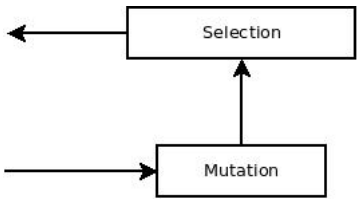

Figure 2.6: DE Evolution

2.5.1 Agents

The candidate solutions found in the population of a DE are refered to as agents. Each agent contains a vector of real numbers which represents their position within the search space.

2.5.2 Mutation

The mutation operator is used to create new individuals from the existing individuals within the population. Individuals are combined using a mathematical formula to create new individuals. Algorithm 2 describes the pseudocode for the mutation operator. The mutation operator is performed once for each agent in the population. To locate a new agent position first three different agents must be randomly selected from the population. The agent's position combined with the three other agents positions based on the mathematical formula shown in algorithm 2. The variable $F \in [0, 2]$ is known as the differential weight, and variable $CR \in [0, 1]$ is known as the crossover probability. Both of these variables are user defined.

Algorithm 2 Mutation

```
for each agent X in the population do
    pick three agents a, b, c randomly from the population
    pick random index R ∈ {1, ..., n}
    copy agent X_i to y
    for each position index y_j in [y_1, ..., y_n] do
        r_j = U(0, 1)
        if r_j < CR or R == j then
            y_j = a_j + F(b_j − c_j)
        else
            y_j = X_ij
        end if
    end for
    if f(y) < f(X_i) then
        X_i = y
    end if
end for
```

2.5.3 Selection

The selection operator determines whether the newly created agent from the mutation operator should be placed in the new population of candidate individuals. Once the new agent is created it is evaluated based on the evaluation function described in Subsection 2.1.2. If the new agent has an improved fitness score over the original agent, the new agent is placed in the new population. Otherwise, the original agent is place in the new population.

Chapter 3

Previous Research

Before we can begin explaining the techniques we used in the next chapter it is necessary that we explain related research in this field.

3.1 Quantum Mechanics/Molecular Mechanics

In previous work [14] the authors used DFT-QM/MM and R-QM/MM techniques to find close approximations of the experimental EXAFS spectrum of OEC in S_1. The EXAFS spectrum used in their calculations was at a poorer resolution compared to the spectra used in the experiments in our work.

Density functional theory quantum mechanics/molecular mechanics (DFT-QM/MM) [15] uses the atoms spatially dependent electron density to determine the position of each atom. Since DFT largely uses function approximations this approach is very limited.

To increase their accuracy the researchers used a refined quantum mechanics/molecular mechanics (R-QM/MM) technique. This approach iteratively adjusted the molecular structure of the molecule and attempted to minimize a scoring function defined in terms of the sum of squared deviations between the experimental and calculated EXAFS spectra. A quadratic penalty was applied to each atom to ensure that their positions did not deviate too far from their original position in order to keep the energy of the system at a minimum.

The researchers speculated that even though the R-QM/MM technique was able to generate an EXAFS spectra closer to the experimental spectra their solution was only a local solution because it was based on their original DFT-QM/MM solution.

Later in [16] the same research group repeated their original experiments performed in [14] with updated X-ray diffraction (XRD) data that had a resolution of 1.9Å. They has success rerunning the DFT-QM/MM experiment followed by the R-QM/MM experiment but still had the same speculations about remaining in a local solution.

3.2 Previous Applications of...

3.2.1 Genetic Algorithms

In [17] P. Comte uses a genetic algorithm to search for solutions to the side-chain packing problem. Each chromosome represents a list of amino-acid residues with a possible rotamer. Comte's method was a able to find improved low energy conformations over conventational methods.

The Laboratory of Crystallography in Zurich, Switzerland developed a method [18] for predicting stable crystal structures and low-energy structures using a genetic algorithm. Each chromosome represents a possible crystal structure, which is a set of atomic coordinates. The GA population was produced either randomly or by user input. Populations were also seeded by the best found crystal structures of previous GA experiments. The lab tested both tradition methods such as simulated annealing and basin hopping against their evolutionary algorithms but prefered the results of the evolutionary algorithm because of its ability to find solutions without knowledge of the problem itself and its ability to move out of local optima search.

3.2.2 Differential Evolution

In 2010 Kalegari, Diego Humberto and Lopes, Heitor Silverio used a differential evolution algorithm for protein structure optimization [19]. They used a simple representation for the protein structure known as hydrophobic/polar (HP). This allowed them

to contrain the system to minimize the search space. The individuals in the DE consisted of a vector of values between $[-\pi, \pi]$ which represent the angle between three monomers. The study was able to find the ground state energy values for problems with smaller sets of amino acids but the researchers found that DE had a tougher time finding the optimal solution as the problem size grew larger.

3.2.3 Particle Swarm Optimization

A research group attempted crystal structure prediction using particle swarm optimization [20]. PSO was selected for comparison against traditional evolutionary methods. The goal was to find optimal structures with the lowest energy. The initial population was generated randomly based on a starting structure. Each of the new random structures was optimized locally before starting the PSO experiment. The local optimization was done using traditional conjugate gradient algorithms. The researchers found that PSO was an efficient method for finding low energy atomic configurations.

Chapter 4

Methodology

4.1 Problem Encoding

A molecule consists of a number of atoms. Each of these atoms has its own 3-dimensional position within the molecule. For the structure refinement problem the individual 3-dimensional position values are not important. The important information about this problem is how the atoms are positioned with respect to each other. Two different forms of representation were used in this work. For each of these representations the number values are shown in Angstroms (Å).

4.1.1 Representation 1

The initial run of experiments used a representation that maintained the initial atomic positions of each atom. The 3-dimensional coordinates were treated as a list of coordinates as shown in Figure 4.1. Using this representation meant that during any form of crossover the tuple of X, Y and Z values would stay together.

X	Y	Z
14.451	-13.346	1.133
15.336	-13.488	2.014
13.005	-13.364	1.452
0.019	0.011	0.045
...	...	...

Figure 4.1: Representation 1

14.451
-13.346
1.133
15.336
-13.488
2.014
13.005
...

Figure 4.2: Representation 2

4.1.2 Representation 2

Algorithms such as particle swarm optimization and differential evolution called for a more flexible representation. The other representation used was simply a list of values. The initial list of 3-dimensional coordinates was converted to a single list of decimal points as shown in Figure 4.2. It is important to note that both of these representations are showing the same information. For fitness evaluation the list of number was converted back to a list of 3-dimensional coordinates by taking segments of three numbers to create a 3-dimensional position.

4.2 Population Generation

An initial population of different individuals needed to be created in order to begin refining the OEC atomic structure using an evolutionary algorithm. The initial OEC atomic structure came from the crystallographic photosystem II (PSII) structure [21]. It is available in the Protein Data Bank (PDB) [22] as PDB ID 3ARC. Two forms of population generation were used during the experiments conducted in this work: random, and molecular dynamics simulation. The atomic structure obtained from

14.451		14.494
-13.346		-13.370
1.133		1.132
15.336	$\Longrightarrow$	15.295
-13.488		-13.518
2.014		2.009
13.005		13.018
...		...

Figure 4.3: Randomly shifting atomic positions by 0.05Å

the PDB contained 1269 chemical elements. For the purposes of OEC structure refinement only 79 specific atoms were required for EXAFS analysis.

4.2.1 Random

A population can be generated randomly based on a starting molecule. To create a random candidate individual each atomic position within the atomic structure is randomly moved by a *user defined range*. This means that if the *user defined range* is 0.05Å each atomic position is randomly shifted ±0.05Å. Using Representation 2 as defined in Subsection 4.1.2 as an example, Figure 4.3 demonstrates how each atomic position is randomly shifted by 0.05Å.

4.2.2 Molecular Dynamics Simulation

An alternative method of population generation was needed to generate individuals that were usable in the experiments. To ensure that the atomic structure was as stable as possible, the structure was put into a molecular dynamics simulation. While in this simulation the molecule is allowed to act as if it were in the real world. The atoms were allowed to move freely in space until the overall temperature of the system was reasonably low. This acted as the baseline atomic structure for all tests. NAMD [7] was used to run the molecular dynamics simulations.

Once the atom structure was stable the temperature within the system was increased. The increased temperature causes the atoms to oscillate their positions but still remain chemically feasible. During this process stapshots of the molecules atomic struc-

ture were recorded. The simulation was allowed to run for 10 000 steps and 10 000 snapshots of the atomic structure were recorded. Each of these snapshots creates a feasible individual for the experiments.

Since 10 000 individuals is more than enough individuals to seed the populations the best individuals were picked. The generated atomic structures were run through IFEFFIT [6] and compared to the target EXAFS spectra. The top 3% (roughly 300) individuals were used to generate the initial populations in the evolutionary algorithms.

4.3 Genetic Operators

4.3.1 Crossover

The basic one-point crossover operator was chosen for the experiments (Described in Subsection 2.2.2). One point crossover is generally less destructive to the individuals than other forms of crossover.

4.3.2 Mutation

For the mutation operator a single atomic coordinate will be moved. A random atomic coordinate is selected from the individual and its position is moved randomly by 0.05Å using Euclidean distance. The resulting position will be 0.05Å away from its original position. In order to determine how much distance the atomic position should be moved, an analysis was needed to learn more about how changing atomic positions affects the calculated EXAFS spectra.

The analysis consisted of moving each atom, individually, in a variety of directions and calculating its RMSD score. Each atom was moved in a total of six directions (±X, ±Y, and ±Z), at a variety of distances (0.001Å, 0.005Å, 0.01Å, 0.025Å, 0.05Å, 0.1Å, 0.5Å, 1Å, and 5Å). This was done to determine how much movement was required of an atom to make a significant change to the RMSD score. Table 4.1 shows results of how much movement is required to produce a 1% and 5% change to their RMSD

Element	1% Difference	5% Difference
O	0.025Å	0.5Å
Mn	0.01Å	0.5Å
Ca	1Å	5Å
C	0.5Å	5Å
N	0.5Å	5Å
H	5Å	5Å

Table 4.1: Minimum Move Required at 1%

scores. Since there is more than one instance of each chemical element in OEC, the distance chosen was the first distance that produced the minimum change because the goal was to find the absolute minimum for each chemical element.

The value of 0.05Å was chosen for the experiments as a middle ground that could be applied to each chemical element. It should be noted that the value of 0.05Å is particular to OEC. A similar analysis could be done to determine the minimum move distance for each element in another chemical complex.

4.3.3 Selection

Tournament selection was used as the selection operator for the genetic algorithms. A tournament size of 3 was used in all experiments.

4.4 Fitness: EXAFS Spectra

The goal of structure refinement as shown in Section 1.4 is to find a calculated EXAFS spectrum that matches the experimental EXAFS spectrum. To calculate how close the calculated EXAFS spectra is to the experimental EXAFS spectra, the root-mean-square deviation (RMSD), see Equation 4.1, will be computed between the calculated and experimental EXAFS spectra. Each spectrum is recorded at an increment of 0.05 k / A^{-1} which allows $EXAFS\chi k^3$ to be compared at each increment. The goal is to get the RMSD value as low as possible because then the calculated and experimental EXAFS spectra match as closely as possible. It is not reasonable to expect the RMSD to be zero, because the experimental EXAFS spectra is not perfect. The environment

in which the EXAFS spectra is recorded creates small errors in the result.

$$RMSD = \sqrt{\frac{\sum_{t=1}^{n} (x_{1,t} - x_{2,t})^2}{n}} \tag{4.1}$$

Chapter 5

Experimental Design

This chapter contains the design details of each experiment within this work. Each experiment includes the purpose of running the experiment and the system parameters used during the experiment.

5.1 Genetic Algorithm: Viability

5.1.1 Purpose

The purpose of this study is to determine the viabiltiy of using an evolutionary algorithm to solve the structure refinement problem (Section 1.4). Previous studies [14] [16] have shown that iterative algorithms work well in finding candidate solutions to the problem. This study will demonstrate how well the population based search algorithms GA, and RGA perform on the structure refinement problem.

5.1.2 Population

During the initial stages of testing a random population of candidate solutions was generated using the technique described in Subsection 4.2.1. The candidate solutions that were generated using the random method had either high fitness scores or were

Experiment Set	1	2	3	4	5	6
Crossover Rate	80	80	70	80	70	80
Mutation Rate	20	10	30	10	30	20
Elitism	False	False	False	True	True	True
Generations	30	30	30	30	30	30
Population Size	50	50	50	50	50	50

Table 5.1: System parameters for the basic GA runs

Experiment Set	1	2	3	4	5	6
Crossover Rate	80	80	80	80	70	70
Mutation Rate	20	20	20	20	30	30
Elitism	True	True	True	True	True	True
Population Size	50	50	50	50	50	50
Convergence Rate	10	5	10	5	5	10
Number of Restarts	3	3	5	5	5	5

Table 5.2: System parameters for the RGA runs

too chemically infeasible and an EXAFS spectrum could not be generated. Candidate solutions that were able to generate EXAFS spectra were repeatedly generated until there were enough to fill the GA population but the experiments were unable to produce a candidate solution that had an EXAFS spectrum that improved upon the starting candidate.

5.1.3 System Parameters

Two evolutionary algorithms were used in this experiment: GA, and RGA. The system parameters for the GA can be seen in Table 5.1, and the system parameters for the RGA can be seen in Table 5.2. The fitness function used for the GA, and RGA is defined in Subsection 4.4.

Each GA type was given a random selection of individuals from the population generated using the molecular dynamics simulation (Subsection 4.2.2). The RGA was given access to the entire population of 300 candidate solutions to be used during the restarting process. The number of generations could not be specified for the RGA experiments because of the restarting process. Details of the genetic operators are outlined in Section 4.3. Representation 1 (Subsection 4.1.1) was chosen for the individuals as a direct mapping to the problem.

5.2 Genetic Algorithm: Post-Optimization

5.2.1 Purpose

The purpose of this study is to improve upon the results found in the study discussed in Section 5.1. The results of the GA viability experiment suggested that the candidate solutions found could be improved upon. Two evolutionary algorithms, differential evolution, and particle swarm optimization, were chosen to perform a local search of the search space to locate improved candidate solutions. These algorithms were selected based on their success with continuous space problems.

5.2.2 Population

The initial population for the DE, and PSO were generated using the random generation technique described in Subsection 4.2.1. The seed candidate solution for the random generation was the best found candidate solution from the RGA experiments, which had an RMSD of 1.046. The random generation technique was used in this experiment because it is suspected as the best technique to find the local optima in this situation.

5.2.3 System Parameters

Two evolutionary algorithms were used in this experiment: DE, and PSO. The system parameters for the DE can be seen in Table 5.3, and the system parameters for the PSO can be seen in Table 5.4. The *initial movement radius* shown in the system parameters tables defines the *user defined range* that was used to generate the initial population. The fitness function used for the DE, and PSO is defined in Subsection 4.4.

An alternative individual representation was used for this experiment. DE, and PSO are algorithms that are better suited for problems that can be represented as a vector of real numbers. The individual representation that was used in the GA viability experiment was translated into a vector of real numbers (See Subsection 4.1.2).

Experiment Set	1	2
Initial Movement Radius	0.05	0.25
Generations	30	30
Population Size	50	50

Table 5.3: System parameters for the Post-Optimization DE runs

Experiment Set	1	2
Initial Movement Radius	0.05	0.25
Generations	30	30
Population Size	50	50

Table 5.4: System parameters for the Post-Optimization PSO runs

5.3 Evolutionary Algorithms

5.3.1 Purpose

The purpose of this study is to determine how well DE, and PSO perform on the structure refinement problem. In Section 5.2 DE, and PSO were used as a post-optimization of the results found in Section 5.1. In this study DE, and PSO will be run with the same individuals used in the initial populations in Section 5.1.

5.3.2 System Parameters

Two evolutionary algorithms were used in this experiment: DE, and PSO. The system parameters for the DE can be seen in Table 5.6, and the system parameters for the PSO can be seen in Table 5.5. The fitness function used for the DE, and PSO is defined in Subsection 4.4. The *velocity parameter* found in Table 5.5 represents the random range that was used to generate the initial *velocity vector* of each individual. For example, a *velocity parameter* of 0.01 means the velocity was generated using the possible values ± 0.01.

Representation 1 (See Subsection 4.1.1) was chosen as the individual represetation for both the DE, and PSO. Since each index within the individual is a 3-dimensional coordinate during the *evolution* of each individual vector arithmetic was performed at each index.

Exp. Set	1	2	3	4	5	6	7	8	9	10	11	12
Pop. Size	50	50	50	50	50	50	100	100	100	100	100	100
Gens.	100	100	100	250	250	250	100	100	100	250	250	250
Velocity	0.01	0.05	0.1	0.01	0.05	0.1	0.01	0.05	0.1	0.01	0.05	0.1

Table 5.5: System parameters for the PSO runs

Experiment Set	1	2	3	4
Population Size	50	50	100	100
Generations	100	200	100	200

Table 5.6: System parameters for the DE runs

5.4 Atom Subsets

5.4.1 Purpose

The purpose of this study is to attempt to reduce the search space of the structure refinement problem. Table 5.7 outlines the number of atoms within the OEC. If we could reduce the number of atoms needed to move during the evolutionary process it would reduce the search space. This study will show how a GA performs on the structure refinement problem when certain chemical elements are kept rigid. A rigid chemical element means that the chemical element will not be evolved during the run. The position of the rigid chemical elements will be the same in each individual in the population.

Chemical Element	**Sum**
Mn	4
Ca	1
O	26
C	14
N	6
H	28

Table 5.7: Chemical Element Breakdown

Runs	10
Population size	50
Crossover rate	0.7
Mutation rate	0.3
Elitism	True
Number of restart attempts	5
Max convergence percentage before restarting	5%

Table 5.8: GA Subset Parameters

Exp. Set	1	2	3	4	5	6	7	8	9
Flexible Atoms	Mn, Ca, C, O, N, H	Mn, Ca, C, O, N	Mn, Ca, C	Mn, Ca, O	Mn, Ca, N	Mn, Ca, C, O	Mn, Ca, C, N	Mn, Ca, O, N	Mn, Ca
Rigid Atoms		H	H, N, O	H, N, C	H, C, O	H, N	H, O	H, C	H, C, O, N

Table 5.9: Experiments with different subsets

5.4.2 System Parameters

A basic GA was used in this experiment. The system parameters used during the experiment are shown in Table 5.8. Table 5.9 outlines the different experiments that were run. Each experiment contains a different combination of rigid and flexible chemical elements but each experiment used the same GA system parameters. The fitness function used for GA is defined in Subsection 4.4.

Chapter 6

Analysis and Discussion

This chapter contains the discussion of results found from the previous chapters experiments.

6.1 Genetic Algorithm: Viability

6.1.1 Analysis

Table 6.1 and Table 6.2 provide a summary of the results from the basic GA, and RGA experiments respectively. Comparing the results shows that the RGA experiments were able to find a better candidate solution than the basic GA experiments.

A closer look at the data revealed that the basic GA experiments were converging early on local optima. The RGA experiments initially converged on similar optima but were able to find new optima after each restarting phase. Figure 6.1 shows what a typical RGA run looks like at each generation. One would notice that there are several *spikes* in the graph where the average fitness jumps. These *spikes* represents a *restart* in the population. During each *restart* the average fitness of the population is disrupted but the fitness quickly improved showing an overall trend downward.

The number of generations in each RGA run varied based on when the population

Experiment Set	1	2	3	4	5	6
Best RMSD	1.2471	1.3315	1.1610	1.1880	**1.1173**	1.2287
Average Best RMSD	1.3448	1.3784	1.2792	1.3315	**1.2626**	1.3336

Table 6.1: Basic GA Results

converged. Table 6.2 contains the average number of generations each run had for each experiment.

At first it may seem biased that the RGA was able to find a better candidate solution than the GA because it was allowed to see more unique individuals and ran for more generations but the GA had soem pitfalls when it was allowed the same privileges. Increasing the size of the GA's populations had little affect on its ability to find improved candidate solutions. Increasing the number of generations for the GA's also did not affect the solutions. The GA runs would converge quickly and become stuck in a local optima.

The GA experiments favoured having *elitism* enabled.

The experiments that produced the best candidate solutions for both the GA, and RGA were experiments that ran with a crossover rate of 70% and a mutation rate of 30% (WHY???).

The best candidate solution found by the RGA can be seen in Figure 6.2. The *humps* on the calculated EXAFS spectrum appear to be getting close to the experimental EXAFS spectrum except in a few instances. Where k is between 7.5 and 10 the experimental EXAFS spectrum is more chaotic and the calculated EXAFS spectrum is having a tougher time conforming. The chaotic nature of the data may be due to the margin of error in collecting the experimental EXAFS spectrum. A future technique may be to smooth out the experimental EXAFS spectrum in order to get a more accurate RMSD comparison. Another structure refinement technique could be to reduce the range on k for comparison. Comparing the RMSD between $k = [1, 7.5]$ may produce improved results because the fitness function would contains less erroneous data points.

Experiment Set	1	2	3	4	5	6
Average Number of Generations	67	81	107	124	67	81
Best RMSD	1.1297	1.1545	1.1132	1.0913	**1.046**	1.0953
Average Best RMSD	1.2281	1.2150	1.2095	1.2189	**1.2031**	1.2171

Table 6.2: RGA Results

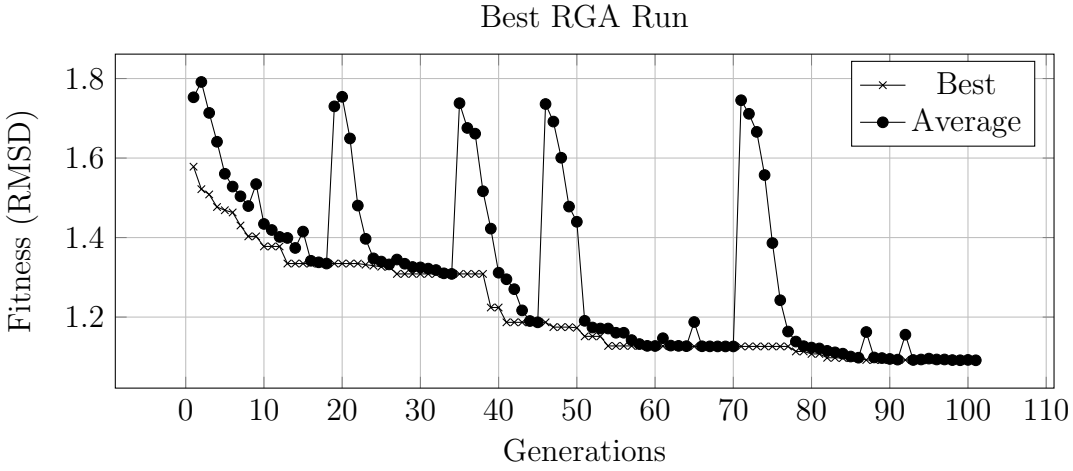

Figure 6.1: Example Run of a Restarting Genetic Algorithm

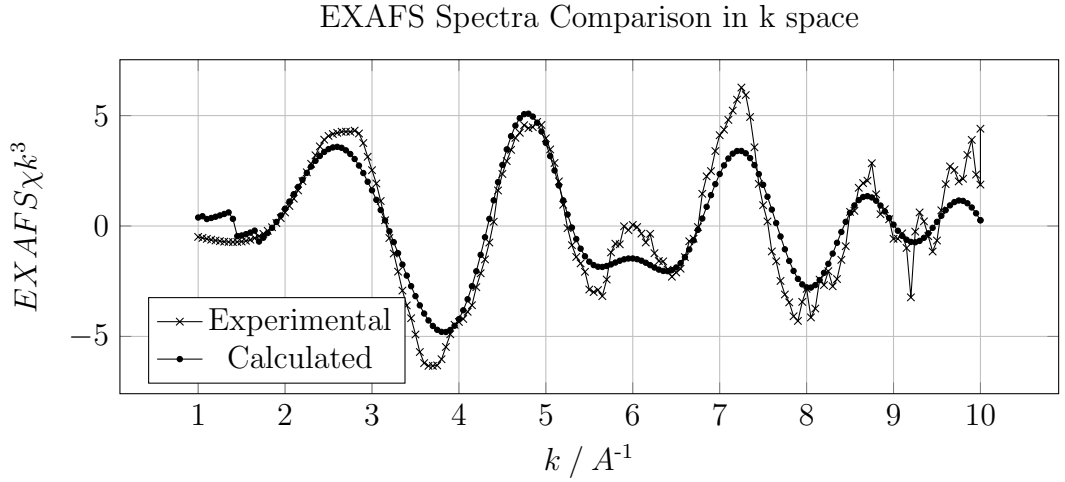

Figure 6.2: Best OEC EXAFS Spectra Comparison from RGA

Experiment Set	1	2
Average Best RMSD	1.1386	1.7267

Table 6.3: Results of DE Post-Optimization

Experiment Set	1	2
Average Best RMSD	0.9001	1.2445

Table 6.4: Results of PSO Post-Optimization

6.2 Genetic Algorithm: Post-Optimization

6.2.1 Analysis

Table 6.3 and Table 6.4 provide a summary of the results from the DE, and PSO experiments respectively. Each experiment was run a total of 30 times to obtain the average. The two algorithms were performed using individuals generated using two different *initial movement radius*. Using the larger *initial movement radius* of ± 0.25Å had a negative impact on the DE, and PSO candidate solutions. The large *initial movement radius* caused the experiments to move from the initial local optima, that was found using the RGA, and move into sub-optimal solutions. As we decreased the *initial movement radius* the results started to improve. Using the smaller *initial movement radius* of ± 0.05Å provided the best results.

The results from the DE experiments were never able to improve upon the seed candidate solution. Decreasing the *initial movement radius* improved the results from the DE but it was not enough to local improved candidate solutions.

In contrast, the results from the PSO experiments were very successful. Using the *initial movement radius* of ± 0.05Å the PSO was able to greatly improve upon the seed candidate solution. Both the DE, and PSO were run for 30 generations. The DE experiments had reached a convergence but the PSO data was still on a downward slope. The PSO was run for 200 generations and it ended up locating even better solutions. Figure 6.3 shows the best EXAFS spectra after optimization from the PSO. The RMSD score of the candidate solution was significantly reduced which shows that PSO works very well as a post-optimization strategy.

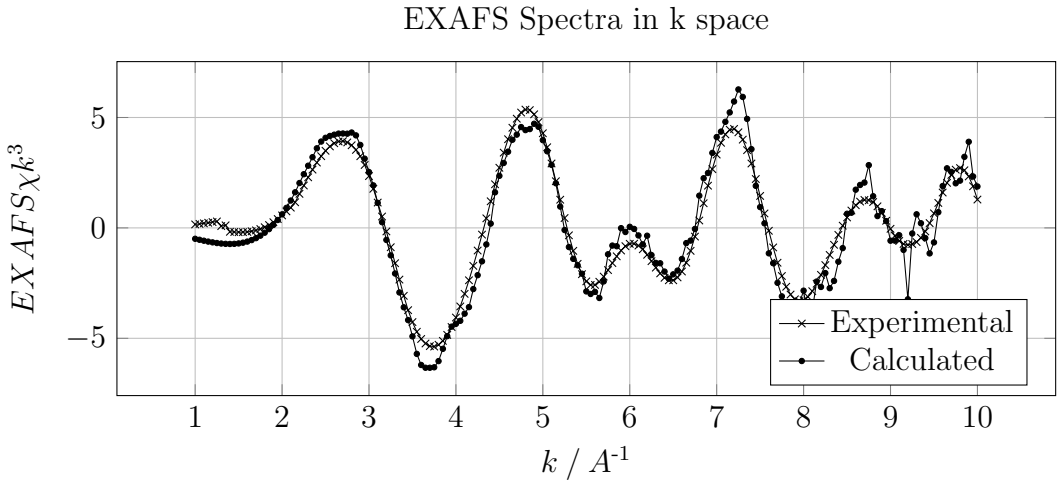

Figure 6.3: OEC EXAFS Spectra Comparison

Exp. Set	1	2	3	4	5	6
Best RMSD	0.7735	0.6840	0.7498	**0.6058**	0.6568	0.6429
Average RMSD	0.9121	0.9046	0.9090	0.7743	0.7643	0.7731

Exp. Set	7	8	9	10	11	12
Best RMSD	0.7392	0.6881	0.7306	0.6440	0.6398	0.6504
Average RMSD	0.8724	0.8876	0.8836	0.7563	0.7441	**0.7369**

Table 6.5: Results for the PSO runs

6.3 Evolutionary Algorithms

6.3.1 Analysis

After viewing the results from the DE, and PSO experiments in Section 6.2 these algorithms were suspected to perform well on the structure refinement problem. Table 6.6 shows a summary from the DE experiments and Table 6.5 provide a summary of the results from the PSO experiments. Each experiment was run a total of 30 times to obtain the average.

Experiment Set	1	2	3	4
Best RMSD	0.9793	**0.9405**	1.0357	0.9646
Average Best RMSD	1.1055	**1.0554**	1.1456	1.0635

Table 6.6: Results for the DE runs

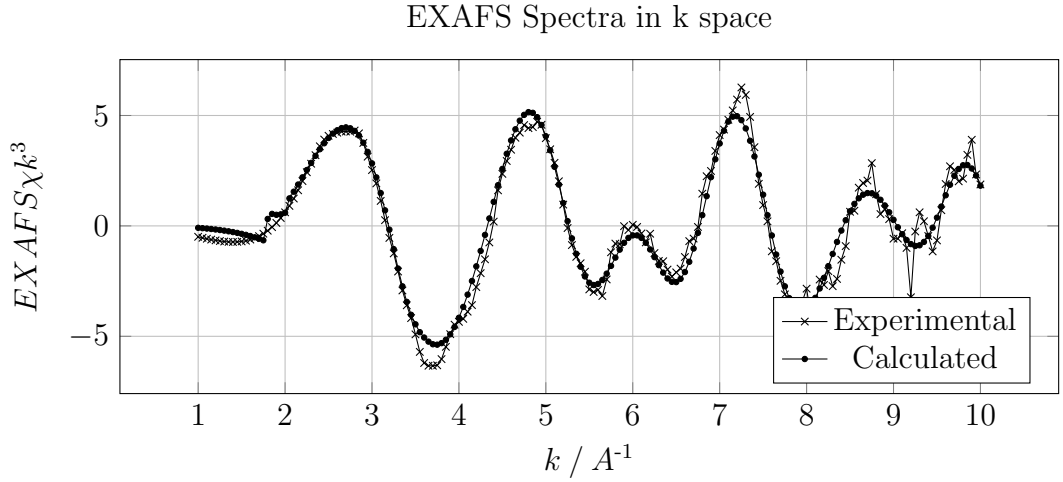

Figure 6.4: OEC EXAFS Spectra Comparison

The DE, and PSO experiments were initially run using a variety of different system parameters. The results shown in Table 6.6 and Table 6.5 used the following tuning parameters. The PSO algorithm has three tunable user defined parameters: inertia, social and cognitive. Based on the work performed in the research paper [23] the parameters social and cognitive was set to 1.496180, while inertia was set to 0.729844. The DE algorithm has two tunable user defined parameters: differential weight, and crossover probability. Several different combinations of values for differential weight, and crossover probability were used but the parameters that produced the best results were from a research paper from Hvass Laboratories [24] where differential weight was set to 0.4717, and crossover probability was set to 0.8803.

6.4 Atom Subsets

6.4.1 Analysis

The results of the atom subset experiments revealed some interesting insights into the structure refinement of atomic structures. Figure 6.7 contains a summary of the results from the atom subset experiments. The most significant result was that keeping the Hydrogen elements rigid actually had little affect on the final results. This is not surprising since the study in Subsection 4.3.2 already revealed that moving a Hydrogen element had very little impact on the fitness score.

Exp. Set	1	2	3	4	5	6	7	8	9
Best RMSD	1.2031	1.1730	2.4481	1.2566	2.4951	**1.1681**	2.5213	N/A	2.4986
Average RMSD	1.2615	1.2656	N/A	N/A	N/A	N/A	2.5720	N/A	2.5158

Table 6.7: Experiments with different subsets

Knowing that the Hydrogen element has little impact on the results of atomic structure refinement means that it could be removed from the individuals that re evolved. Removing the Hydrogen element would decrease the chromosome length from 79 to 51. The reduced chromosome length would allow for more different combinations to be attempted and reduced degrees of freedom.

Since the Manganese (Mn), and Calcium (Ca) are at the core of the OEC molecule these chemical elements could not be left rigid during the experiments. Leaving any of the other chemical elements rigid during the experiments showed little improvement. Keeping the Carbon (C), Oxygen (O), or Nitrogen (N) elements rigid during the experiments either caused the RGA run to become stuck in an early local optima or created atomic structures that would become unable to perform the EXAFS calculations. The N/A's within Figure 6.7 represent results that were unable to be calculated successfully. The populations of most of the runs were becoming polluted with invalid atomic configurations that could not produce EXAFS spectra.

In order to make the atomic structure refinement work using only a subset of the atoms would require the assistance of a molecular dynamics simulation such as NAMD [7]. Once the candidate individuals were allowed to evolve for a few generations some corrections to their atomic structures would have to be made using the molecular dynamics simulation.

Chapter 7

Conclusions and Future Work

7.1 Conclusion

7.2 Future Work

Bibliography

[1] D. J. Vinyard, G. M. Ananyev, and G. C. Dismukes, "Photosystem II: The reaction center of oxygenic photosynthesis.," *Annual Review of Biochemistry*, vol. 82, pp. 577 – 606, 2013.

[2] J. Yano and J. Kern, "Manganese: The oxygen-evolving complex and models," *Encyclopedia of Inorganic and Bioinorganic Chemistry*, 2006.

[3] M. Newville, "Fundamentals of xafs," *Consortium for Advanced Radiation Sources, University of Chicago (USA)[http://xafs. org]*, 2004.

[4] W. D. Cornell, P. Cieplak, C. I. Bayly, I. R. Gould, K. M. Merz, D. M. Ferguson, D. C. Spellmeyer, T. Fox, J. W. Caldwell, and P. A. Kollman, "A second generation force field for the simulation of proteins, nucleic acids, and organic molecules," *Journal of the American Chemical Society*, vol. 117, no. 19, pp. 5179–5197, 1995.

[5] J. W. Ponder, D. A. Case, *et al.*, "Force fields for protein simulations," *Advances in protein chemistry*, vol. 66, pp. 27–86, 2003.

[6] T. U. of Chicago, "Ifeffit: Interactive xafs analysis." Accessed: 2014-04-01.

[7] Theoretical and C. B. Group, "Namd - scalable molecular dynamics." Accessed: 2014-04-01.

[8] Theoretical and C. B. Group, "Namd energy plugin." Accessed: 2014-04-01.

[9] J. Hughes, S. Houghten, and D. Ashlock, "Recentering, reanchoring & restarting an evolutionary algorithm," in *Nature and Biologically Inspired Computing (NaBIC), 2013 World Congress on*, pp. 76–83, IEEE, 2013.

[10] J. Hughes, J. A. Brown, S. Houghten, and D. Ashlock, "Edit metric decoding: Representation strikes back," in *Evolutionary Computation (CEC), 2013 IEEE Congress on*, pp. 229–236, IEEE, 2013.

[11] J. Kennedy, "Particle swarm optimization," in *Encyclopedia of Machine Learning*, pp. 760–766, Springer, 2010.

[12] R. Poli, J. Kennedy, and T. Blackwell, "Particle swarm optimization," *Swarm intelligence*, vol. 1, no. 1, pp. 33–57, 2007.

[13] R. Storn and K. Price, "Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces," *Journal of global optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[14] E. M. Sproviero, J. A. Gascón, J. P. McEvoy, G. W. Brudvig, and V. S. Batista, "A model of the oxygen-evolving center of photosystem II predicted by structural refinement based on exafs simulations," *Journal of the American Chemical Society*, vol. 130, no. 21, pp. 6728–6730, 2008.

[15] R. G. Parr and W. Yang, *Density-functional theory of atoms and molecules*, vol. 16. Oxford university press, 1989.

[16] S. Luber, I. Rivalta, Y. Umena, K. Kawakami, J.-R. Shen, N. Kamiya, G. W. Brudvig, and V. S. Batista, "S_1-state model of the O_2-evolving complex of photosystem II," *Biochemistry*, vol. 50, no. 29, pp. 6308–6311, 2011.

[17] P. Comte, "Bio-inspired optimization & sampling technique for side-chain packing in mcce," 2010.

[18] A. R. Oganov and C. W. Glass, "Crystal structure prediction using ab initio evolutionary techniques: Principles and applications," *The Journal of chemical physics*, vol. 124, no. 24, p. 244704, 2006.

[19] D. H. Kalegari and H. S. Lopes, "A differential evolution approach for protein structure optimisation using a 2d off-lattice model," *International Journal of Bio-Inspired Computation*, vol. 2, no. 3, pp. 242–250, 2010.

[20] Y. Wang, J. Lv, L. Zhu, and Y. Ma, "Crystal structure prediction via particle-swarm optimization," *Physical Review B*, vol. 82, no. 9, p. 094116, 2010.

[21] Y. Umena, K. Kawakami, J.-R. Shen, and N. Kamiya, "Crystal structure of oxygen-evolving photosystem ii at a resolution of 1.9 Å," *Nature*, vol. 473, no. 7345, pp. 55–60, 2011.

[22] "Crystal structure of oxygen-evolving photosystem ii at 1.9 angstrom resolution."

[23] R. C. Eberhart and Y. Shi, "Comparing inertia weights and constriction factors in particle swarm optimization," in *Evolutionary Computation, 2000. Proceedings of the 2000 Congress on*, vol. 1, pp. 84–88, IEEE, 2000.

[24] M. E. H. Pedersen, "Good parameters for differential evolution," *Magnus Erik Hvass Pedersen*, 2010.