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 proven somewhat successful in structure refinement but have not been successful in finding the global minima.

Multiple population based algorithms, including a genetic algorithm, a restarting genetic algorithm, differential evolution, and particle swarm optimization, are studied for their effectiveness in structure refinement of EXAFS. The oxygen-evolving complex in S_1 is used as a benchmark for comparing the algorithms. These algorithms were successful in finding new atomic structures that produced improved calculated EXAFS spectra over atomic structures previously found.

Acknowledgements

First and foremost, I would like to thank my supervisor, Dr. Sheridan Houghten for being the best mentor I could have asked for. Without her guidance and patience I would not have been able to achieve the success I have today. I would also like to thank my advisory committee Dr. Doug Bruce, Dr. Brian Ross, and external examiner Dr. Dan Ashlock.

Special thanks goes to the Brock University Computer Science Department. Especially, Donna Phelps for her help navigating the paper work and Cale Fairchild for taking the time to ensure that my experiments ran smoothly and efficiently.

I would like to show my appreciation for Sergey Vassiliev. He helped with the understanding of the biological background and was a vital asset in the success of this work.

Finally, I would like to thank my friends and family, especially Gary and Cathie Price. They supported and motivated me during my research. Without their continuous support I would not be the person I am today.

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. 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 transition 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 is an important biological blueprint for water spliting. 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. A visualization of the inorganic core of OEC in S_1 is shown in Figure 1.1. This figure was generated using the Visual Molecular Dynamics software [3].

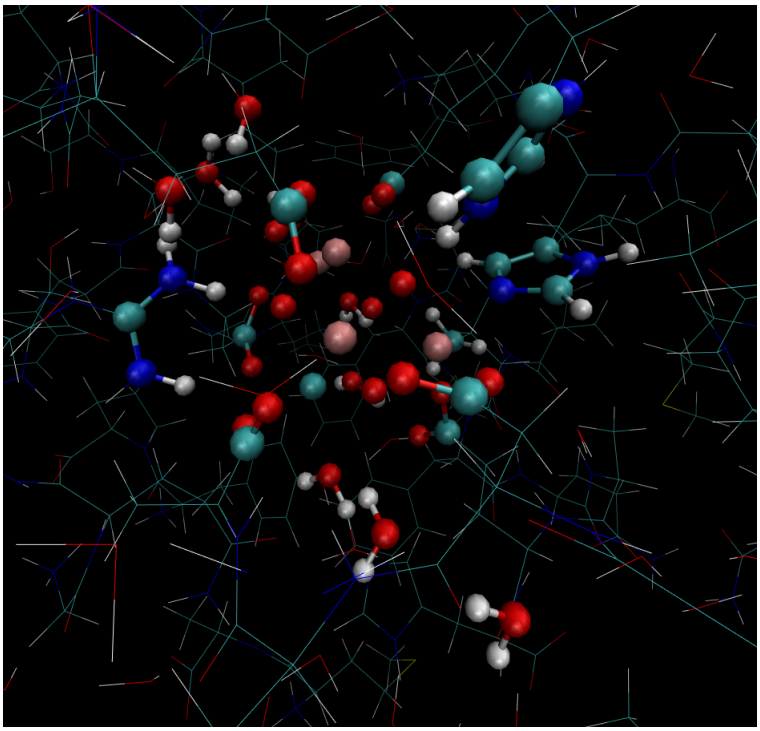

Figure 1.1: OEC Atomic Structure in S_1

1.2 X-ray Absorption Spectroscopy

The following overview is based on information contained in Matthew Newville's *Fundamentals of XAFS* (2004) [4]. X-Ray absorption fine structure (XAFS) is a

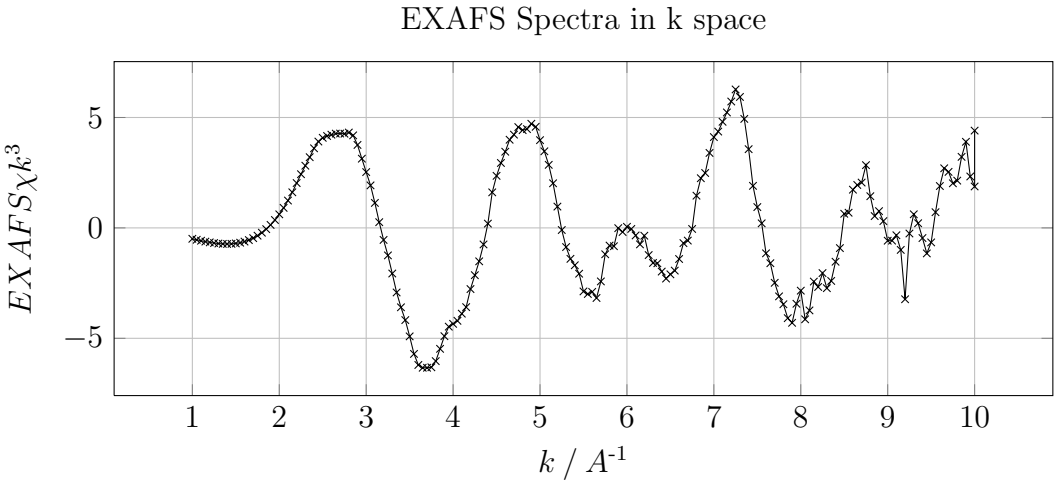

Figure 1.2: EXAFS Spectra of OEC in S_1

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 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 spectrum for OEC in S_1 is shown in Figure 1.2.

1.3 Force Fields

The atoms within a molecule are consistently interacting with each other. 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, 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) [5] force fields for the energy calculations. AMBER force fields are widely used with proteins and related systems [6].

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. The algorithm must be able to find an atomic structure whose EXAFS spectrum matches the experimental EXAFS spectrum and has relatively low energy.

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. Although an EXAFS spectrum allows you to identify how far apart atoms are from each other, it 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, which means 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 feasible. Having a set of candidate solutions will improve the odds of finding the actual solution.

The IFEFFIT XAFS data analysis suite [7] 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 [8] will be used for the energy calculations. The NAMD Energy Plugin [9] 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 gives an overview of the various search algorithms used in this work. This chapter gives an overview of evolutionary algorithms, and provides details on the different implementations of genetic algorithms, restarting genetic algorithms, differential evolution, and particle swarm optimization. Chapter 3 discusses previous research that has been performed on the structure refinement problem and how other algorithms were used on similar problems. Chapter 4 provides implementation details for the various algorithms, such as the problem encoding and population generation. Chapter 5 outlines the different experiments that will be performed. Chapter 6 discusses the results from the experiments performed and provides detailed analysis of the findings. Chapter 7 summarizes the research conducted and outlines possible future work. Appendices are also included in this thesis. The appendices contain summary tables of the best fitness score found in each run for all experiments.

Chapter 2

Background

The purpose of this chapter is to assist the reader in understanding the search techniques used in this thesis. Several different population based search algorithms, including a genetic algorithm, restarting genetic algorithm, differential evolution, and particle swarm optimization are defined in this chapter.

2.1 Genetic Algorithm

A genetic algorithm (GA) is a population-based metaheuristic optimization algorithm that is based on Darwin's theory of natural evolution. Over a period of time a population of individuals mate and create offspring. Darwin theorized that not all offspring are created equally and that eventually the weaker individuals would die off as a result of not being well adapted to their environment, leaving the strong to survive and reproduce. This same principle can be applied to a search algorithm as a heuristic. A GA contains a population of individuals that are evolved to find improved candidate solutions.

Figure 2.1 demonstrates how the basic GA 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.

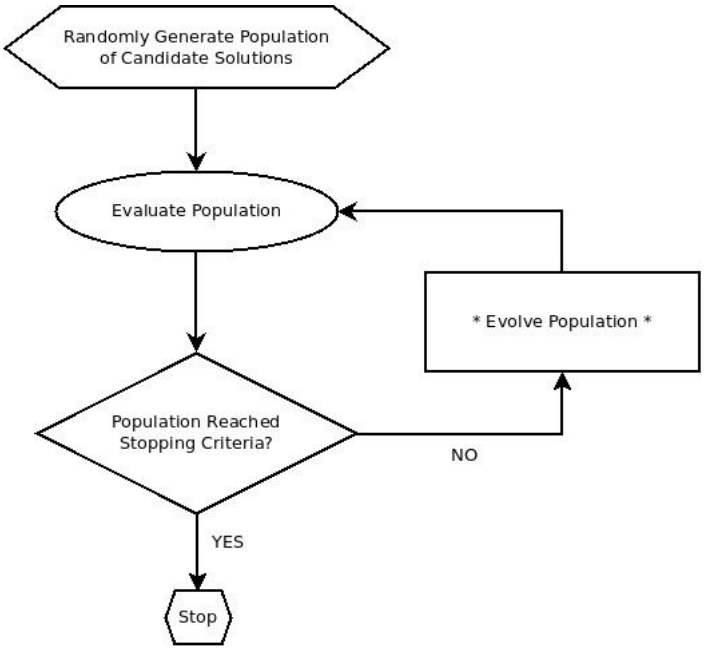

Figure 2.1: Basic GA Flowchart

2.1.1 Population

The population is a key piece to a GA. 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 from which to evolve solutions.

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 separately,

and replication copies the individuals. These 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 2.1.4 for more details on the operators discussed.

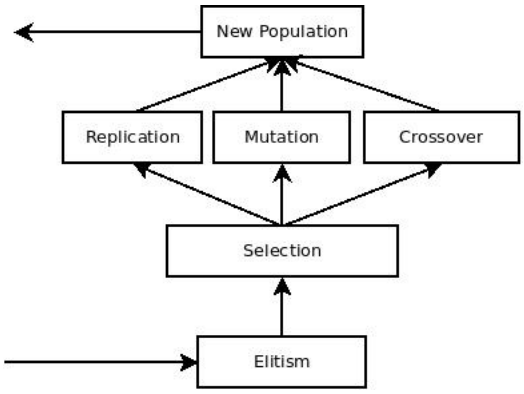

Figure 2.2: GA Evolution

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 sometimes 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 GA 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 an acceptably high quality solution has been found.

2.1.4 Genetic Operators

Each of the following operators is a component of a genetic algorithm. They facilitate the evolutionary process in the effort to find better candidate solutions. Only a few of the different methods will be described.

Selection Operator: 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 continuing on to the next population. During the selection process two individuals are chosen using a selection method and then are either bred together, mutated, or replicated and placed in the next population. There are several varieties of selection methods but only tournament selection will be explained as this is the method used here.

Tournament selection works by randomly selecting k individuals from the population, where k is usually between 2 to 7, 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.

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.

For example, one-point crossover could be used. This crossover method works by splitting the two individuals apart based on a randomly selected pivot point and swapping the pieces. In Figure 2.3, the pivot point was selected between index 2 and 3. The information after the pivot point is swapped between the parents to create the children. Each child will contain a piece of information from both parents.

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

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

Figure 2.3: One-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 new areas of 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 a given individual is modified. Figure 2.4 demonstrates single-point mutation.

Individual	0	1	1	0	1	0	1	0

Mutant	0	1	1	1	1	0	1	0

Figure 2.4: 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 is used. The elitism operator works by seeding the next generation's population with the individuals with the best fitness scores from the previous generation. Only a very small number of individuals (at most 1%) are copied into the next generation.

2.2 Recentering Genetic Algorithm

The recentering genetic algorithm (RGA) is a variation of the recentering-restarting genetic algorithm (RRGA) [10] [11] which has had success in avoiding fixation 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 begin to converge. After the population has converged upon a minimum diversity, new individuals are introduced to the population. Individuals are considered to be the same if they share the same fitness score. 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
        generate new individual
        insert new individual into the population
    end while
end if
```

2.3 Differential Evolution

Differential evolution (DE) [12] is a population based search metaheuristic designed to iteratively improve candidate solutions to a problem. DE is well suited for problems containing a nonlinear and non-differentiable 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 created the fitness scores of each solution are compared and the candidate with the better fitness score is put into the new population. Subsection 2.3.1 describes the operators used.

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

2.3.1 Mutation and Selection

The mutation operator is used to create new individuals from the existing individuals. Individuals are combined using the mathematical formula, shown in Algorithm 2, 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's position first three different agents must be randomly selected from the population. The agent's position is combined with the three other agents' positions to create a new position. Each position index within the agent is updated either based on the three other agents selected or the value from the previous agent is copied. The new agent's position is evaluated based on the fitness function. If the agent's new position has resulted in an improved fitness score the new position replaces the old one. If not, the new position is discarded. 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
```

2.4 Particle Swarm Optimization

Particle swarm optimization (PSO) [13] [14] is a population based search metaheuristic designed to iteratively optimize a problem. PSO is well suited for problems containing a nonlinear and non-differentiable continuous search space. The candidate solutions found within a PSO are known as *particles*. Each of these particles represents a candidate solution's 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 Sub-

section 2.4.3, its 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 shown in Equation 2.1.

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

In this equation $r_p, r_g \sim U(0, 1)$, and ω, ϕ and φ are user defined. The parameter ω (inertia) controls the efficiency of the particle moving through the search space. The parameters 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 as shown in Equation 2.2.

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

Chapter 3

Previous Research

Before we can begin explaining the techniques we used in the next chapter, it is necessary that we survey related research in this field. Section 3.1 reviews the previous work that has been done on the structure refinement of OEC and Section 3.2 reviews research that has been performed on structure refinement in other applications.

3.1 Quantum Mechanics/Molecular Mechanics

In previous work [15] the authors used density functional theory quantum mechanics/molecular mechanics (DFT-QM/MM) and refined quantum mechanics/molecular mechanics (R-QM/MM) 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. DFT-QM/MM [16] 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 R-QM/MM. 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 the atoms' positions did not deviate too far from their original positions

Algorithm	DFT-QM/MM [17]	R-QM/MM [17]
Best RMSD	1.2679	1.2437

Table 3.1: Results of Previous Work

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 spectrum closer to the experimental spectrum their solution was only a local solution because it was based on their original DFT-QM/MM solution.

Later in [17] the same research group repeated their original experiments performed in [15] with updated X-ray diffraction (XRD) data that had a closer resolution of 1.9Å. They had success in rerunning the DFT-QM/MM, and R-QM/MM experiment but still had the same speculations about remaining in a local optimum. Their paper included the best atomic structures they were able to achieve. We have analyzed these structures using the same EXAFS spectra fitness score (see Section 4.4) and included them in Table 3.1.

3.1.1 Genetic Algorithm

A study conducted in [18] had success in EXAFS fitting using an annealing evolutionary algorithm. The researchers combined a genetic algorithm with a simulated annealing algorithm in order to locate the global optima. During each generation new candidate solutions were either accepted or rejected according to the Metropolis criterion, which is used to control the distribution of the population. For their EXAFS spectrum analysis the researchers were able to generate an EXAFS spectrum using an equation. The equation was able to generate an EXAFS spectrum based on five structural control parameters: coordination distance (r), coordination number (N), Debye-Waller factor (σ), electron mean free path (λ) and ΔE_0. The control parameters were randomly generated using specific constraints for each. Least squares fitting was used as an objective function between the experimental and calculated EXAFS spectra. The experimental EXAFS spectrum was generated from two Cu samples. Using the annealing evolutionary algorithm, the researchers were able to find more accurate results than existing methods at the time.

3.2 Previous Applications

In this section, we look at previous applications of GA, DE, and PSO to biological problems, specifically concentrating on those that attempt to identify structures.

3.2.1 Genetic Algorithms

In [19] a genetic algorithm is used to search for solutions to the side-chain packing problem. Each chromosome represented a list of amino-acid residues with a possible rotamer. The method was able to find improved low energy conformations over conventional methods.

The Laboratory of Crystallography in Zurich, Switzerland developed a method for predicting stable crystal structures and low-energy structures using a genetic algorithm [20]. Each chromosome represented 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 traditional methods such as simulated annealing and basin hopping against their evolutionary algorithms, but preferred 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.

3.2.2 Differential Evolution

In [21] a differential evolution algorithm for protein structure optimization. They used a simple representation for the protein structure known as hydrophobic/polar (HP). This allowed them to constrain the system to minimize the search space. The individuals in the DE consisted of a vector of values between $[-\pi, \pi]$ which represent the angles 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

In [22] particle swarm optimization was used to prediction crystal structures. 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

In this chapter we describe the methodologies we used with the various evolutionary algorithms in our research. Section 4.1 explains how a molecule's atomic structure is translated to a more usable encoding. The different methods used for population generation are outlined in Section 4.2. Section 4.3 defines the genetic operators used in the GA and RGA and Section 4.4 reviews how each individual will be evaluated.

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 co-

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

ordinates 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 if crossover is suitably implemented.

4.1.2 Representation 2

Algorithms such as particle swarm optimization and differential evolution called for a more flexible representation. Therefore, 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 numbers 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 [23]. It is available in the Protein Data Bank (PDB) [24] 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 the PDB contained 1269 chemical elements. For the purposes of OEC structure refinement only 79 specific atoms were required for EXAFS spectra 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 then each atomic position will be randomly moved to a new atomic position that is a euclidean distance 0.05Å away from its original position.

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 [8] 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 molecule's atomic structure were recorded. The simulation was allowed to run for 10000 steps and

10000 snapshots of the atomic structure were recorded. Each of these snapshots creates a feasible individual for the experiments.

Since 10000 individuals is more than enough individuals to seed the populations the best individuals were picked. The generated atomic structures were run through IFEFFIT [7] 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

In this section each of the genetic operators will be explained briefly. The parameters are summarized in Table 4.1.

Crossover	One-point
Mutation	0.05
Tourament Size	3

Table 4.1: GA Operators

4.3.1 Crossover

The basic one-point crossover operator was chosen for the experiments, as described in Subsection 2.1.4. 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 altered 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

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 4.2: Minimum Move Required at 1%

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 4.2 shows the results of how much movement is required to produce a 1% and 5% change to their RMSD 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 performed 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 the energy levels ($EXAFS\chi k^3$) 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. For each experiment, the system parameters and experimental setup are discussed.

5.1 Genetic Algorithms

5.1.1 Purpose

The purpose of this study is to determine how well a genetic algorithm performs on the structure refinement problem (Section 1.4). Previous studies [15] [17] 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 chemically infeasible and an EXAFS spectrum could not be generated. Candidate

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 Size	0	0	0	1	1	1
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

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.

This led to a change in the way the initial population was created. The initial population for the basic GA, and RGA was created from a random sampling of the 300 candidate solutions that were generated using the molecular dynamics simulation discussed in Subsection 4.2.2. During the restarting process of the RGA new candidate solutions were randomly selected from the remaining candidate solutions within the 300.

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. These parameters were empirically determined. The fitness function used for the GA, and RGA is defined in Subsection 4.4.

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.

Experiment Set	1	2	3	4	5	6	7	8
Crossover Rate	80	80	80	80	70	70	70	70
Mutation Rate	20	20	20	20	30	30	30	30
Elitism Size	1	1	1	1	1	1	1	1
Population Size	50	50	50	50	50	50	50	50
Convergence Rate	10	5	10	5	10	5	10	5
Number of Restarts	3	3	5	5	3	3	5	5

Table 5.2: System parameters for the RGA runs

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 experiments (see Section 6.1) 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 optimum in this situation.

5.2.3 System Parameters

Two 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

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

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).

5.3 Alternative 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 more directly compared with the algorithms used in Section 5.1.

5.3.2 Population

The initial population for both the DE, and PSO will be created from a random sampling of the 300 candidate solutions that were generated using the molecular dynamics simulation discussed in Subsection 4.2.2.

Exp. Set	1	2	3	4	5	6
Pop. Size	50	50	50	50	50	50
Gens.	100	100	100	200	200	200
Velocity	0.01	0.05	0.1	0.01	0.05	0.1

Exp. Set	7	8	9	10	11	12
Pop. Size	100	100	100	100	100	100
Gens.	100	100	100	200	200	200
Velocity	0.01	0.05	0.1	0.01	0.05	0.1

Table 5.5: System parameters for the PSO runs

5.3.3 System Parameters

Two 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.

Representation 1 (See Subsection 4.1.1) was chosen as the individual representation for both the DE, and PSO. Since each index within the individual is a 3-dimensional coordinate, these values are updated using standard vector arithmetic at each index.

The *velocity parameter* found in Table 5.5 represents the random range that was used to generate the initial *velocity vector* of each particle in the PSO. For example, a *velocity parameter* of 0.01 means that each velocity vector was generated using random values between -0.01 and 0.01.

The DE, and PSO experiments were initially run using a variety of different tuning parameters. The PSO algorithm has three tunable user defined parameters: *inertia*, *social* and *cognitive*. The values used in this experiment were selected based on the work performed by Eberhart, and Shi [25] and are shown in Table 5.7. 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 by Hvass Laboratories [26] and are shown in Table 5.8.

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

Inertia	0.729844
Social	1.496180
Cognitive	1.496180

Table 5.7: Algorithm parameters for the PSO 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 5.9 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.

5.4.2 System Parameters

A basic GA was used in this experiment. The system parameters used during the experiment are shown in Table 5.10. Table 5.11 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.

Differential Weight	0.4717
Crossover Probability	0.8803

Table 5.8: Algorithm parameters for the DE runs

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

Table 5.9: Chemical Element Breakdown

Runs	10
Population size	50
Crossover rate	0.7
Mutation rate	0.3
Elitism Size	1

Table 5.10: GA Subset Parameters

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

Exp. Set	6	7	8	9
Flexible Atoms	Mn, Ca, C, O	Mn, Ca, C, N	Mn, Ca, O, N	Mn, Ca
Rigid Atoms	H, N	H, O	H, C	H, C, O, N

Table 5.11: Experiments with different subsets

Chapter 6

Analysis and Discussion

This chapter contains the discussion of results found from the previous chapter's experiments. The complete results for each experiment can be found in the Appendices.

6.1 Genetic Algorithms

Table 6.1 and Table 6.2 provide a summary of the results from the basic GA, and RGA experiments respectively. Full results can be found in Appendix A. The bolded values represent the experiments with the best fitness scores. Each experiment in the results table was run 30 times.

The Mann-Whitney U test [27] is a nonparametric test to determine if two groups are equivalent without assuming the groups have a normal distribution. Mann-Whitney U tests were performed on the results of the basic GA, and RGA experiment sets using a confidence interval of 95%. The results of the basic GA experiments in Table 6.1 showed very little statistical significance. Experiments 3 and 5 show the best statistical performance over the other experiments which coincides with these

Experiment Set	1	2	3	4	5	6
Best RMSD	1.2471	1.1880	1.1173	1.2349	**1.0533**	1.2287
Average Best RMSD	1.3518	1.3610	**1.2942**	1.3658	1.3044	1.3294

Table 6.1: Basic GA Results

Experiment Set	1	2	3	4
Avg. Num. Generations	61	73	86	106
Best RMSD	1.1297	1.1174	1.0388	**0.9649**
Average Best RMSD	1.2532	1.2468	1.2252	1.2149

Experiment Set	5	6	7	8
Avg. Num. Generations	72	83	100	133
Best RMSD	1.1170	1.0012	1.0353	0.9992
Average Best RMSD	1.2229	1.2119	**1.1808**	1.1856

Table 6.2: RGA Results

experiments having the best fitness and average fitness scores. The results of the RGA experiments revealed that experiment 8 was able to perform statistically better than the other experiments. Experiments 3, 4, 5, 6, and 7 show little statistical difference and experiments 1, and 2 performed the worst. A comparison of the basic GA, and RGA experiments showed that the RGA performed statistically better than the basic GA.

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 represent a restart in the population. During each restart the average fitness of the population is disrupted but the fitness quickly improved showing an overall downward trend.

The number of generations in each RGA run varied based on when the population 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 some pitfalls when it was allowed the same privileges. Increasing the size of the GA's population had little affect on its ability to find improved candidate solutions. Increasing the number of generations for the GAs also did not affect the solutions. The GA runs would typically converge quickly and become stuck in a local optimum. Figure 6.2 shows the average best fitness for GA experiment 3.

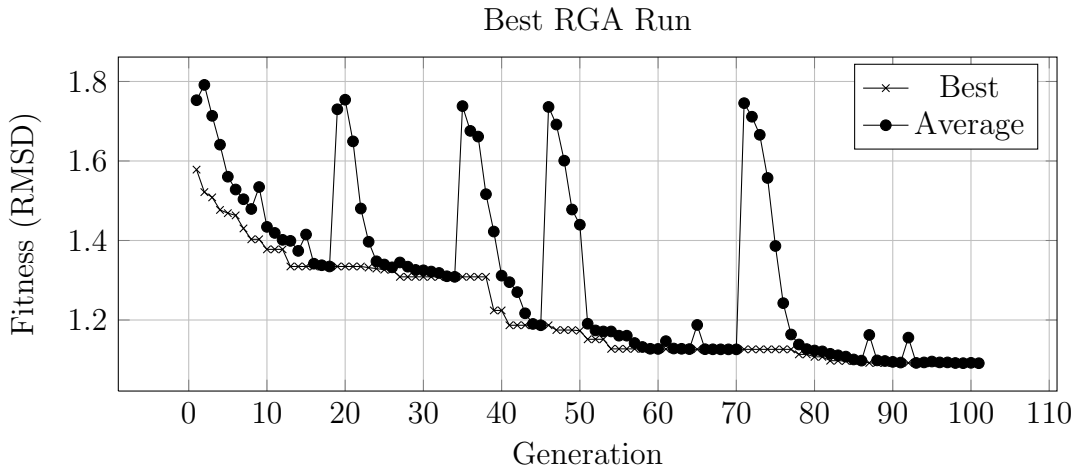

Figure 6.1: Example Run of a Restarting Genetic Algorithm

The experiments that produced the best candidate solutions for both the GA, and RGA were experiments that contained a crossover rate of 70% and a mutation rate of 30% compared to the other experiments that contained a crossover rate of 80% and a mutation rate of 20%. Since the higher mutation rate performed better than the higher crossover rate this may indicate that the algorithms favoured receiving new information, through exploration, than exploiting the information already contained within the population. Using a higher mutation rate may also indicate that the problem is better suited to be solved using a less discrete search method such as PSO since the candidate solutions operate on a continuous space.

The best candidate solution found by the RGA can be seen in Figure 6.3. 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 contain fewer erroneous data points.

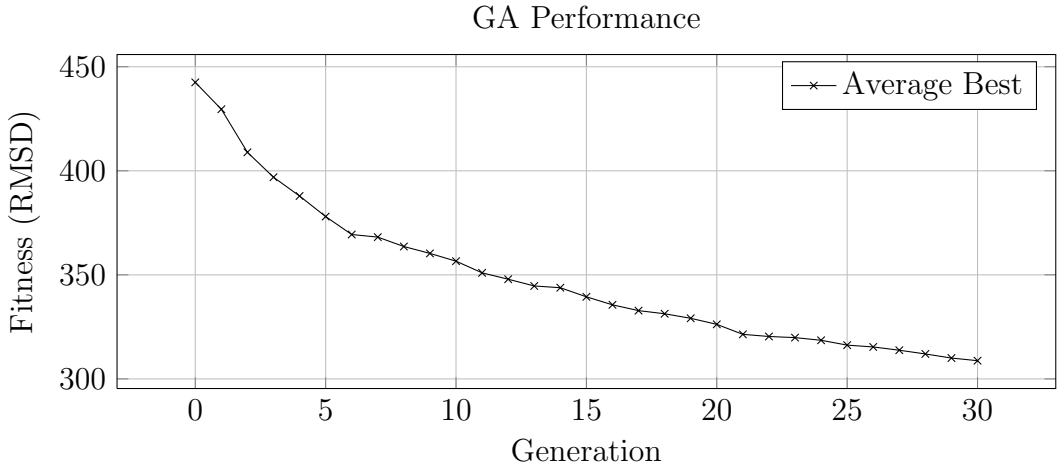

Figure 6.2: Performance of GA Experiment 3

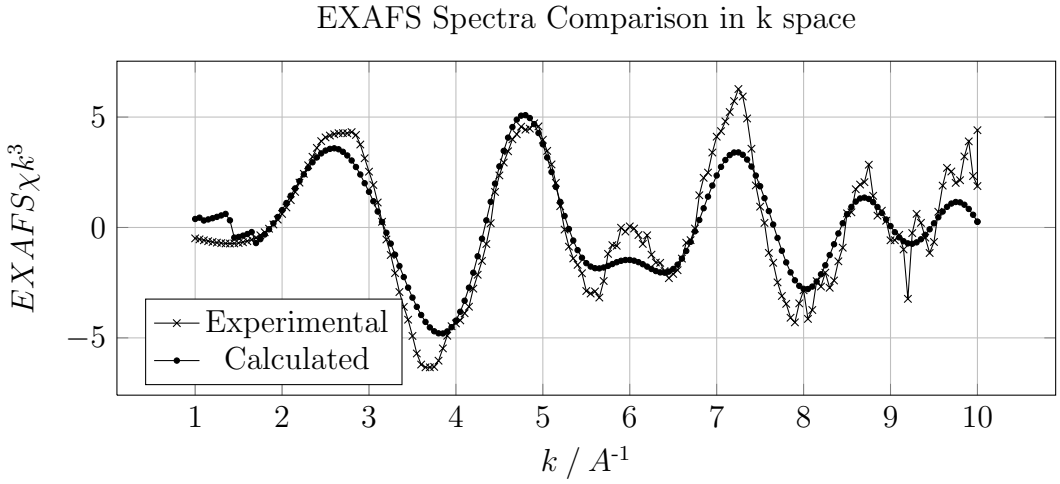

Figure 6.3: Best OEC EXAFS Spectra Comparison from RGA

Experiment Set	1	2
Best RMSD	**0.9973**	1.4118
Average Best RMSD	**1.1386**	1.7267

Table 6.3: Results of DE Post-Optimization

Experiment Set	1	2
Best RMSD	**0.7977**	0.9296
Average Best RMSD	**0.9001**	1.2445

Table 6.4: Results of PSO Post-Optimization

6.2 Genetic Algorithm: Post-Optimization

Table 6.3 and Table 6.4 provide a summary of the results from the DE, and PSO experiments respectively. Full results can be found in Appendix A. The bolded values represent the experiments with the best fitness scores. Each experiment in the results table was run 30 times.

The two algorithms were performed using individuals generated using two different *initial movement radii*. 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 optimum, that was found using the RGA, 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 for the DE but it was still unable to produce results that improved upon the seed candidate solution.

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. Figure 6.4 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.

Mann-Whitney U tests were performed on the results of the DE, and PSO experiment

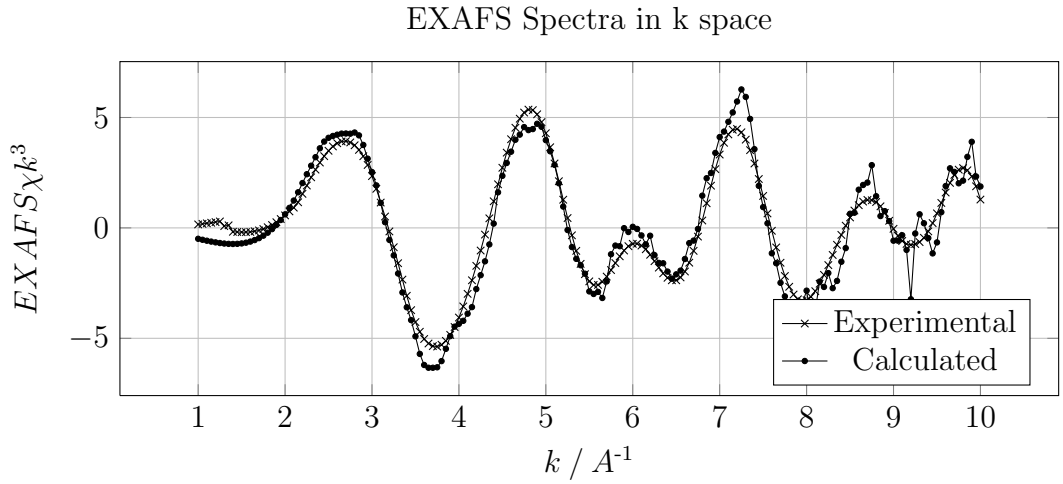

Figure 6.4: OEC EXAFS Spectra Comparison

sets and the results from the RGA experiment sets in Section 6.1 using a confidence interval of 95%. The results of the PSO experiments were statistically better than the results of the DE experiments. The improved results found using the reduced *initial movement radius* were statistically consistent. Comparing the PSO experiment results with the RGA experiment results showed that the PSO performed statistically better than the RGA. It is worth noting that although the DE experiments could not improve upon the seed candidate solution the results of the DE compared to the RGA were statistically better.

6.3 Alternative Algorithms

Table 6.5 and Table 6.6 provide a summary of the results from the DE, and PSO experiments respectively. Full results can be found in Appendix A. The bolded values represent the experiments with the best fitness scores. Each experiment in the results table was run 30 times.

Experiment Set	1	2	3	4
Best RMSD	0.9793	**0.9405**	1.0357	0.9646
Average Best RMSD	1.1120	**1.0540**	1.1453	1.0624

Table 6.5: Results for the DE runs

Exp. Set	1	2	3	4	5	6
Best RMSD	0.7735	0.6840	0.7498	**0.6109**	0.6653	0.6621
Average Best RMSD	0.9136	0.9004	0.9049	0.8025	0.7907	0.7933

Exp. Set	7	8	9	10	11	12
Best RMSD	0.7392	0.6881	0.7306	0.6546	0.6571	0.6688
Average Best RMSD	0.8775	0.8856	0.8836	0.7750	**0.7714**	0.7594

Table 6.6: Results for the PSO runs

The DE, and PSO experiments performed very well on the structure refinement problem. These algorithms were able to succeed in finding better candidate solutions to the problem than the results from the RGA in Section 6.1 and the PSO in Section 6.2.

Mann-Whitney U tests were performed on the results of the DE, and PSO experiment sets using a confidence interval of 95%. The results from testing only the PSO experiment set revealed that there were two clear statistical groupings. The two groups consisted of the experiments with a generation count of 100, and the experiments with a generation count of 200. The results within each of these two groups had no statistical difference but the group with a higher generation count performed statistically better than the other group. Modifying the initial velocity speeds had little statistical effect on the solutions. The results of the DE showed the same outcomes. Two statistical groups formed from the experiments that had the same number of generations. The experiments with the greater number of generations performed statistically better than the other group.

The two experiment sets shared exactly the same statistical conclusions. The population sizes had no effect on the results of the experiments. The experiments with a population size of 50 performed statistically the same as the ones with a population size of 100. This characteristic could be caused by the method by which the populations are initialized. The molecular dynamics simulation causes the individual atoms to oscillate back and forth. The individuals used in the initial populations may have been similar enough that there was some overlap in the search space. Having two statistical groups forming around the number of generations only makes sense. The group with the greater generation count was allowed more time to explore the search space.

Mann-Whitney U tests were also performed on the results of the DE, and PSO experiment sets, and also the experiment sets from the RGA in Section 6.1. The tests

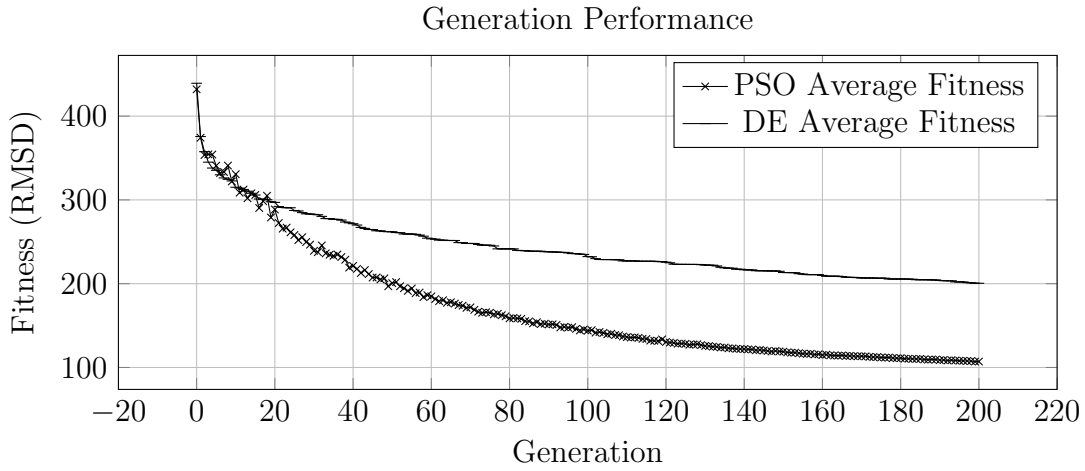

Figure 6.5: Performance of PSO Experiment 11 and DE Experiment 2

showed that PSO performed statistically better than DE, and RGA performed statistically worse than DE, and PSO. These results were actually not that surprising considering the results from the analysis done in Section 6.2 showed that PSO worked well on the structure refinement problem.

Figure 6.6 demonstrates one of the best candidate solutions. It can be seen that the candidate solution has produced an EXAFS spectrum that is a close approximation of the experimental EXAFS spectrum. The remaining differences in the EXAFS spectra may be due to errors in the experimental EXAFS spectrum. Figure 6.5 shows the performce of the best performing experiments for PSO and DE. Both algorithms show a steep downward trend within the first 20 generations but the PSO is able to progress at a faster rate.

The results found in this section indicate that algorithms that operate on a continuous space perform better than those that use a more discrete search space. We suspect that this may be true only for smaller search spaces. The complex analyzed in this work is relatively small with only 79 atoms required for EXAFS spectra comparison. If one was attempting to optimize the EXAFS spectrum and the force fields involved, the search space would grow to 1269 atoms. This is an exponential increase is degrees of freedom. In this case a genetic algorithm may be better suited.

Table 6.7 provides a comparison of the results found in a previous study [17] to the results found in this work. Each of the algorithms used was able to find a better candidate solution for OEC in S_1. The major difference between the candidate solu-

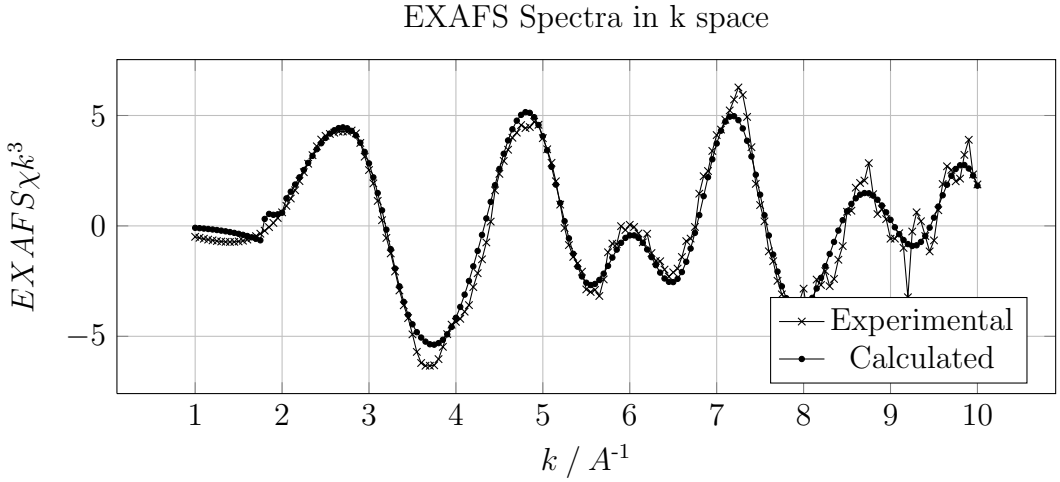

Figure 6.6: OEC EXAFS Spectra Comparison

Algorithm	Best RMSD
DFT-QM/MM [17]	1.2679
R-QM/MM [17]	1.2437
GA	1.0533
RGA	0.9649
Post-Optimized PSO	0.7977
DE	0.9405
PSO	**0.6109**

Table 6.7: Summary of Best Candidate Solutions

tions found in the previous study [17] and this work is that their candidate solutions have also had their force fields optimized. We suspect that the previous researchers' goals of optimizing both the force fields and EXAFS spectrum is what caused their candidate solutions to become stuck in a local optimum. Force field calculations do play an important role in creating a stable candidate solution but these alterations to the atomic structure may be better suited as a post-optimization.

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 6.8 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 effect 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.

Knowing that the hydrogen element has little impact on the results of atomic structure refinement means that it could be removed from the individuals. 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 optium or created atomic structures that would become unable to perform the EXAFS calculations. The N/A's within Figure 6.8 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 [8]. 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.

Exp. Set	1	2	3	4	5
Best RMSD	1.2031	1.1730	2.4481	1.2566	2.4951
Average RMSD	1.2615	1.2656	N/A	N/A	N/A

Exp. Set	6	7	8	9
Best RMSD	**1.1681**	2.5213	N/A	2.4986
Average RMSD	N/A	2.5720	N/A	2.5158

Table 6.8: Experiments with different subsets

Chapter 7

Conclusions and Future Work

This thesis studies the performance of multiple population-based search algorithms on the structure refinement problem using EXAFS spectra comparison. The experimental EXAFS spectrum for the oxygen-evolving complex was used for testing. We have shown that a basic GA, and RGA perform well at finding close candidate solutions to the experimental EXAFS spectrum. DE and PSO were performed as a post-optimization of the basic GA, and RGA in order to improve upon their results. The PSO was successful in finding an improved optima. DE and PSO were then run as a more direct comparison to the basic GA and RGA. These algorithms were able to outperform the GA and RGA results showing that algorithms that perform on a continuous space are better suited for this type of problem. Although these algorithms were successful in finding a new optimum for the oxygen-evolving complex in S_1, future testing on other states of the oxygen-evolving complex or other complexes should be performed.

This work was primarily a study of the computational side of the structure refinement problem. The atomic structures produced in this work were only validated against the experimental EXAFS spectrum. A biologist will still have to verify which candidate solutions are potential solutions to the problem. Future research into using population based search algorithms on the structure refinement problem would benefit from including force field calculations on the candidate solutions. A multi-objective approach where each atomic structure's EXAFS spectrum is refined as well as the potential energy of the atomic structure is minimized should be investigated.

A molecular dynamics simulation was used to generate individuals that were appropriate to seed the initial populations of the algorithms examined in this work. Future work might benefit from repeated use of the molecular dynamics simulation. Once a candidate solution is located this candidate could be placed back into the molecular dynamics simulation where a new batch of candidate solutions could be generated to seed another round of refinement. This method would allow the search algorithm to continue to be single objective because the molecular dynamics simulation would correct any issues with the atomic structures force fields.

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] Theoretical and C. B. Group, "Vmd - visual molecular dynamics." `http://www.ks.uiuc.edu/Research/vmd/`. Accessed: 2014-04-01.

[4] M. Newville, "Fundamentals of xafs," *Consortium for Advanced Radiation Sources, University of Chicago (USA)[http://xafs.org]*, 2004. http://xafs.org/Tutorials?action=AttachFile&do=get&target=Newville_xas_fundamentals.pdf.

[5] 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.

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

[7] T. U. of Chicago, "Ifeffit: Interactive xafs analysis." `http://cars9.uchicago.edu/ifeffit/Ifeffit`. Accessed: 2014-04-01.

[8] Theoretical and C. B. Group, "Namd - scalable molecular dynamics." `http://www.ks.uiuc.edu/Research/namd/`. Accessed: 2014-04-01.

[9] Theoretical and C. B. Group, "Namd energy plugin." `http://www.ks.uiuc.edu/Research/vmd/plugins/namdenergy/`. Accessed: 2014-04-01.

[10] 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.

[11] 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.

[12] 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.

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

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

[15] 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.

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

[17] 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.

[18] W. Cai, L. Wang, Z. Pan, and X. Shao, "Analysis of extended x-ray absorption fine structure spectra using annealing evolutionary algorithms," *Anal. Commun.*, vol. 36, no. 8, pp. 313–315, 1999.

[19] P. Comte, "Bio-inspired optimization & sampling technique for side-chain packing in mcce," *MSc. Thesis, Brock University*, 2010.

[20] 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.

[21] 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.

[22] 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.

[23] 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.

[24] K. Umena and K. Shen, "Crystal structure of oxygen-evolving photosystem ii at 1.9 angstrom resolution." `http://www.rcsb.org/pdb/explore/explore.do?structureId=3arc`, 2011.

[25] 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.

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

[27] P. E. McKnight and J. Najab, "Mann-whitney u test," *Corsini Encyclopedia of Psychology*, 2010.

Appendix A

Summary of Results

Run	Exp. 1	Exp. 2	Exp. 3	Exp. 4	Exp. 5	Exp. 6
1	1.3428	1.2424	1.2945	1.3701	1.3247	1.3709
2	1.2471	1.4306	1.2552	1.3575	1.2755	1.3693
3	1.2992	1.1880	1.2144	1.3787	1.2790	1.3007
4	1.2942	1.3885	1.3071	1.3315	1.1618	1.3522
5	1.3993	1.2563	1.2518	1.4015	1.3237	1.3224
6	1.3332	1.4274	1.3800	1.4087	1.1610	1.4091
7	1.3636	1.3646	1.1959	1.3598	1.2899	1.4123
8	1.3438	1.3635	1.1173	1.4316	1.3449	1.2376
9	1.4919	1.3106	1.3297	1.3641	1.3053	1.2287
10	1.3178	1.3209	1.2607	1.3783	1.3115	1.3195
11	1.2693	1.4062	1.2732	1.4141	1.0533	1.2933
12	1.3241	1.4196	1.2254	1.4146	1.3025	1.3349
13	1.2751	1.3636	1.3250	1.3602	1.3073	1.3483
14	1.3123	1.2769	1.2198	1.3791	1.2695	1.3819
15	1.3578	1.4808	1.3391	1.2739	1.2064	1.3494
16	1.4347	1.2823	1.3326	1.4350	1.2647	1.3639
17	1.3475	1.4272	1.2402	1.3066	1.2606	1.3500
18	1.3017	1.4155	1.3969	1.3482	1.3344	1.2709
19	1.4571	1.2273	1.2620	1.3697	1.3756	1.3409
20	1.3927	1.3743	1.3054	1.2994	1.2210	1.2913
21	1.3439	1.4167	1.4121	1.3950	1.2945	1.2885
22	1.3002	1.4028	1.2122	1.3428	1.3311	1.3075
23	1.4578	1.3585	1.2837	1.3818	1.4682	1.2890
24	1.2869	1.2723	1.3317	1.3860	1.2917	1.4294
25	1.2641	1.4774	1.3325	1.3635	1.4501	1.3429
26	1.4350	1.4298	1.2616	1.2349	1.3040	1.2469
27	1.3986	1.3895	1.3587	1.2431	1.3799	1.3247
28	1.2988	1.3234	1.3017	1.3468	1.4099	1.3936
29	1.5162	1.3651	1.4062	1.4813	1.3823	1.2933
30	1.2920	1.3698	1.3462	1.3864	1.3653	1.2905

Table A.1: Best RMSD for GA Experiments

Run	Exp. 1	Exp. 2	Exp. 3	Exp. 4
1	1.2063	1.1545	1.3276	1.3535
2	1.2706	1.3501	1.1927	1.2102
3	1.2187	1.2426	1.2577	1.1344
4	1.1297	1.2421	1.2153	1.2447
5	1.2374	1.2040	1.1132	1.2955
6	1.3025	1.2099	1.2803	1.1923
7	1.2421	1.2348	1.1689	1.2160
8	1.2530	1.2545	1.1457	1.2248
9	1.2328	1.2873	1.2249	1.0913
10	1.2511	1.2604	1.1526	1.2068
11	1.2877	1.3057	1.2437	1.2651
12	1.3789	1.1638	1.1807	1.0660
13	1.3211	1.2366	1.2122	1.2409
14	1.2915	1.2242	1.2830	1.2336
15	1.2111	1.2352	1.2853	1.2984
16	1.3023	1.1174	1.1367	1.2162
17	1.2594	1.2762	1.2255	1.0912
18	1.1899	1.2912	1.4292	1.2518
19	1.3177	1.2182	1.0388	1.2069
20	1.2636	1.3225	1.2516	1.2213
21	1.2414	1.2857	1.1815	1.1947
22	1.2334	1.2494	1.1230	1.2673
23	1.1386	1.2704	1.2364	1.2601
24	1.2662	1.1787	1.1568	1.1883
25	1.2814	1.1977	1.2121	0.9649
26	1.2450	1.2847	1.4148	1.1812
27	1.1692	1.2193	1.2055	1.2463
28	1.2842	1.3093	1.2208	1.2043
29	1.2352	1.2636	1.2778	1.3322
30	1.2996	1.2837	1.2801	1.2700

Table A.2: Best RMSD for RGA Experiments

Run	Exp. 5	Exp. 6	Exp. 7	Exp. 8
1	1.1343	1.2256	1.1484	1.1490
2	1.2821	1.1523	1.2118	1.2394
3	1.4133	1.2554	1.1787	1.0956
4	1.2993	1.1361	1.2471	1.2195
5	1.1944	1.2295	1.1991	1.2188
6	1.1386	1.1675	1.2531	1.2680
7	1.2247	1.2105	1.2039	1.2358
8	1.2763	1.2174	1.1696	1.0953
9	1.2225	1.2553	1.2451	1.3076
10	1.2309	1.2084	1.1692	1.3195
11	0.9992	1.1406	1.4056	1.2381
12	1.1867	1.2006	1.1978	1.1554
13	1.2022	1.1954	1.2086	1.1898
14	1.0740	1.1270	1.1835	1.2645
15	1.2094	1.1233	1.2236	1.2669
16	1.1295	1.1478	1.1672	1.3418
17	1.2555	1.0729	1.2260	1.1329
18	1.1199	1.1670	1.2066	1.0012
19	1.0545	1.2809	1.3594	1.2314
20	1.0443	1.1192	1.1395	1.2430
21	1.0532	1.1698	1.1891	1.1267
22	1.1794	1.1955	1.1170	1.2810
23	1.1758	1.1470	1.2526	1.3144
24	1.2687	1.2614	1.3395	1.0967
25	1.2854	1.2504	1.2935	1.2893
26	1.1247	1.0353	1.2713	1.2048
27	1.0728	1.0982	1.3057	1.2983
28	1.2216	1.2005	1.1184	1.0877
29	1.2834	1.1705	1.1771	1.2324
30	1.1015	1.2206	N/A	1.1261

Table A.3: Best RMSD for RGA Experiments

Run	Exp. 1	Exp. 2
1	1.1609	1.7515
2	1.1319	1.5903
3	1.1182	1.7561
4	1.1205	1.7175
5	1.1934	1.6874
6	1.1322	1.7442
7	1.1222	1.6023
8	1.1722	1.8597
9	1.1717	1.8060
10	1.0968	1.7440
11	1.1343	1.4118
12	1.1545	1.7996
13	1.0831	1.8263
14	1.1588	1.6057
15	1.1657	1.7369
16	1.1458	1.6848
17	1.1698	1.6669
18	1.1936	1.6874
19	1.1171	1.8532
20	1.1182	1.7480
21	1.1347	1.8118
22	1.1066	1.6816
23	1.0855	1.7415
24	1.2093	1.8370
25	0.9973	1.7543
26	1.1234	1.7459
27	1.1482	1.7013
28	1.1784	N/A
29	1.1409	N/A
30	1.1513	N/A

Table A.4: Best RMSD for Post-Optimized DE Experiments

Run	Exp. 1	Exp. 2
1	0.9151	0.9296
2	0.8902	1.2441
3	0.9391	1.1869
4	0.8935	1.3929
5	0.9083	1.1275
6	0.8565	1.0846
7	0.8802	1.1160
8	0.9659	1.2459
9	0.8876	1.2899
10	0.9062	1.3036
11	0.8453	1.2866
12	0.8882	1.1360
13	0.9810	1.2542
14	0.9469	1.0714
15	0.8264	1.2136
16	0.8939	1.1847
17	0.9315	1.5407
18	0.9200	1.1791
19	0.9031	1.3381
20	0.8645	1.2664
21	0.8973	1.5527
22	0.8743	1.1913
23	0.9037	1.2747
24	0.9206	1.2567
25	0.9425	1.3737
26	0.9557	1.2934
27	0.7977	1.2228
28	0.8896	1.0576
29	0.9214	1.3527
30	0.8299	1.1649

Table A.5: Best RMSD for Post-Optimized PSO Experiments

Run	Exp. 1	Exp. 2	Exp. 3	Exp. 4
1	1.0947	1.0369	1.1534	1.0627
2	1.1838	1.1350	1.1317	1.0745
3	1.0542	0.9405	1.2284	1.0905
4	1.1576	0.9870	1.0357	1.0362
5	1.0615	0.9695	1.1977	1.1983
6	1.0462	1.0739	1.1583	0.9944
7	1.1524	0.9966	1.2570	1.1107
8	1.0714	1.0861	1.1033	1.0569
9	1.1384	0.9984	1.1251	1.0422
10	1.1492	0.9429	1.2231	1.0622
11	1.0862	1.0276	1.1332	1.1233
12	1.0723	1.1215	1.1791	1.0546
13	1.1322	0.9958	1.1528	1.0272
14	0.9945	1.0380	1.1504	1.0770
15	0.9793	0.9829	1.1058	1.1498
16	1.2365	1.1555	1.0402	1.0309
17	1.0162	1.0647	1.2093	1.0624
18	0.9918	1.1114	1.1733	1.0123
19	1.1503	0.9863	1.0506	1.0933
20	1.1360	1.0286	1.1565	1.0631
21	1.0469	0.9740	1.1425	0.9867
22	1.2461	1.1057	1.1287	0.9860
23	1.1370	1.1796	1.1840	1.1172
24	1.1202	1.0440	1.0851	1.1081
25	1.1940	1.0880	1.1462	1.0102
26	1.1322	1.0817	1.1781	1.0887
27	1.2220	1.0682	1.0399	0.9646
28	1.0922	1.0686	1.0731	1.0709
29	0.9973	1.1550	1.1918	1.0569
30	1.1940	1.1176	1.1818	1.0252

Table A.6: Best RMSD for DE Experiments

Run	Exp. 1	Exp. 2	Exp. 3	Exp. 4	Exp. 5	Exp. 6
1	0.8197	0.7489	0.9836	0.8171	0.8470	0.6556
2	1.0025	0.9738	0.9658	0.7014	0.8052	0.7628
3	0.7872	0.9279	0.9754	0.7798	0.7241	0.8674
4	0.8424	0.6840	0.9000	0.7208	0.8335	0.7605
5	0.8995	1.0466	0.7647	0.7617	0.7355	0.8063
6	0.9506	0.8541	0.9073	0.7812	0.8220	0.8825
7	0.8528	1.0103	0.9102	0.7378	0.7840	0.7886
8	0.8986	0.8520	0.8979	0.9230	0.7311	0.6846
9	0.9755	0.8705	0.8353	0.8099	0.7756	0.7687
10	0.9383	0.8888	0.9919	0.9178	0.6710	0.7855
11	0.9011	0.7625	0.8584	0.6058	0.6837	0.7721
12	0.9912	0.9123	0.8404	0.6938	0.8538	0.7805
13	0.7735	0.8613	0.8323	0.7980	0.8345	0.8875
14	0.8914	0.8162	0.8897	0.7468	0.6688	0.8332
15	0.9159	0.8654	0.8637	0.7844	0.7967	0.8173
16	0.9804	0.9103	0.8477	0.7551	0.7858	0.7499
17	0.8443	0.8958	0.9534	0.8967	0.7419	0.6823
18	0.9484	0.9315	0.8881	0.7634	0.7249	0.7694
19	1.0433	0.8812	0.7498	0.8150	0.7001	0.7305
20	0.8857	0.9461	0.9195	0.6873	0.6568	0.7606
21	0.8449	0.9020	0.8648	0.7039	0.7107	0.7979
22	0.9429	0.9750	0.8999	0.7616	0.7578	0.7790
23	0.8644	0.9197	1.0113	0.7503	0.7884	0.7927
24	0.9459	0.9313	0.9418	0.8278	0.6979	0.7586
25	1.0301	0.7882	0.9926	0.7165	0.7547	0.7584
26	0.9285	0.9310	0.9590	0.7146	0.7541	0.7428
27	1.0634	0.8417	0.9220	0.7324	0.7986	0.6429
28	0.9197	1.0061	0.9515	0.7003	0.8918	0.7439
29	0.7917	1.0398	0.9433	0.8628	0.7662	0.7845
30	0.8423	0.9242	0.8145	0.9149	0.8186	0.7857

Table A.7: Best RMSD for PSO Experiments

Run	Exp. 7	Exp. 8	Exp. 9	Exp. 10	Exp. 11	Exp. 12
1	0.9270	0.8076	0.9662	0.7244	0.6398	0.6932
2	0.7905	0.8177	0.8218	0.7407	0.7283	0.6880
3	0.9417	0.9338	0.9220	0.6915	0.7256	0.7270
4	0.8550	0.8481	0.8518	0.7126	0.7067	0.7775
5	0.8616	0.7913	0.8582	0.6661	0.7306	0.7404
6	0.9856	1.0768	0.9277	0.7531	0.8182	0.7138
7	0.9107	1.0613	1.1638	0.7320	0.7380	0.7533
8	0.8585	0.9062	0.9856	0.7270	0.6905	0.7260
9	0.8223	0.9639	0.9816	0.7079	0.7306	0.6772
10	0.8071	0.9237	0.7306	0.7501	0.8087	0.6601
11	1.0090	0.7786	0.8236	0.6771	0.7501	0.7306
12	1.0178	0.9102	0.9673	0.7748	0.7043	0.7589
13	0.8550	0.9328	0.8439	0.6876	0.7324	0.6743
14	0.8156	0.8014	0.8637	0.6440	0.7555	0.7693
15	0.7392	0.8737	0.8853	0.6973	0.7604	0.7208
16	0.9616	0.6881	0.7609	0.8127	0.7124	0.7813
17	0.8477	0.8879	0.8338	0.6749	0.7411	0.7694
18	0.8817	1.0310	0.8838	0.7172	0.7446	0.6504
19	0.8584	0.7231	0.8812	0.8473	0.7526	0.7592
20	0.9008	0.9182	0.7702	0.8641	0.8013	0.6955
21	0.8728	0.8939	0.7786	0.7013	0.8071	0.8554
22	0.8521	0.7832	0.8041	0.7332	0.7241	0.7648
23	0.7795	0.8741	0.8847	0.7919	0.7109	0.7622
24	0.8601	0.8460	0.8822	0.7956	0.7456	0.8122
25	0.7956	1.0012	0.9158	0.8009	0.7591	0.7142
26	0.8813	0.9367	0.8132	0.8801	0.6765	0.6888
27	0.9053	0.7580	0.8559	0.8477	0.8089	0.6963
28	0.9357	0.9317	0.9111	0.7787	0.7570	0.7423
29	0.7806	0.8921	0.9109	0.8378	0.7654	0.7802
30	0.9345	0.8313	0.9112	0.7535	0.7615	0.8331

Table A.8: Best RMSD for PSO Experiments