12-2014

Ranking Methods for Global Optimization of Molecular Structures

John Norman McMeen Jr
East Tennessee State University

Ranking Methods for Global Optimization of Molecular Structures

---

A thesis

presented to

the faculty of the Department of Computing

East Tennessee State University

In partial fulfillment

of the requirements for the degree

Master of Science in Computer Science

---

by

John N. McMeen Jr.

December 2014

Dr. Frank B. Hagelberg, Co-chair

Dr. Martin L. Barrett, Co-chair

Dr. Phillip E. Pfeiffer, IV

Keywords: Computer-Aided Molecular Design, Heuristics, Evolutionary Algorithms, Molecular Dynamics

ABSTRACT

Ranking Methods for Global Optimization of Molecular Structures

by

John N. McMeen Jr.

This work presents heuristics for searching large sets of molecular structures for low-energy, stable systems. The goal is to find the globally optimal structures in less time or by consuming less computational resources. The strategies intermittently evaluate and rank structures during molecular dynamics optimizations, culling possible weaker solutions from evaluations earlier, leaving better solutions to receive more simulation time. Although some imprecision was introduced from not allowing all structures to fully optimize before ranking, the strategies identify metrics that can be used to make these searches more efficient when computational resources are limited.

TABLE OF CONTENTS

Page

ABSTRACT 2
LIST OF TABLES 5
LIST OF FIGURES 6

Chapter
1. INTRODUCTION 7
The Problem 8
Approach 8
Results 10
Overview 10
2. BACKGROUND 11
Computer Aided Molecular Design 11
Evolutionary Algorithms 12
Fitness Heuristics 14
Evolutionary CAMD 15
VASP 15
3. METHODS 17
Goals 17
Experimental Design 18
Implementation 20
Computing Platform 20
Controller Application 20
Initialization 20
Random Molecule Generation 21
Silicon Clusters 24
Heuristic Testing 24
Control Method 24
Fitness Heuristics 24
Performance 26

4. RESULTS 28
Test Configurations 28
Test Configuration 1 29
Test Configuration 2 30
Test Configuration 3 30
5. ANALYSIS 32
Run Times 32
Accuracy 36
Overall Performance 38
Parallelization 39
Si7 Clusters 40
6. CONCLUSION 42
REFERENCES 43
APPENDICES 45
APENDIX A; SELECTED TERMS 45
APPENDIX B; DATABASE SCHEMA 46
APENDIX C; EXPERIMENTAL RESULTS 47
APPENDIX D; EXAMPLE INPUT 55
VITA 57

LIST OF TABLES

Table Page

1. Accuracy calculation example 27
2. Simulation group test configurations 28
3. Test configuration 1 overall results 29
4. Test configuration 2 overall results 30
5. Test configuration 3 overall results 31
6. Test configuration comparison 39
7. Test configuration 1 - accuracy percentage per heuristic and simulation group 47
8. Test configuration 1 - time in VASP (seconds) per simulation group and heuristic 48
9. Test configuration 1 - each heuristic's percentage of time in VASP relative to control method's VASP time 49
10. Test configuration 1 - accuracy percentage and VASP time percentage per heuristic and simulation group 50
11. Test configuration 2 - accuracy percentage per heuristic and simulation group 51
12. Test configuration 2 - time in VASP (seconds) per simulation group and heuristic 51
13. Test configuration 2 - each heuristic's percentage of time in VASP relative to control method's VASP time 52
14. Test configuration 2 - accuracy percentage and VASP time percentage per heuristic and simulation group 52
15. Test configuration 3 - accuracy percentage per heuristic and simulation group 53
16. Test configuration 3 - time in VASP (seconds) per simulation group and heuristic 53
17. Test configuration 3 - each heuristic's percentage of time in VASP relative to control method's VASP time 54
18. Test configuration 3 - accuracy percentage and VASP time percentage per heuristic and simulation group 54

LIST OF FIGURES

Figure Page

1. Controller application overview 9
2. The general scheme of an EA as flow chart 13
3. Two-dimensional representation of generateNeighboringAtom method 22
4. Random coordinate generation 23
5. Heuristic control flow diagram 25
6. Test configuration 1 - control method vs. heuristics' VASP time (seconds)..... 33
7. Test configuration 2 - control method vs. heuristics' VASP time (seconds)..... 34
8. Test configuration 3 - control method time vs. heuristics' VASP time (seconds) 35
9. Test configuration 1 - heuristics' accuracy percentage per simulation group 36
10. Test configuration 2 - heuristics' accuracy percentage per simulation group 37
11. Test configuration 3 - heuristics' accuracy percentage per simulation group 38
12. Si4 and Si5 geometric configurations 41
13. Si7 structures identified with control method and heuristics..... 41
14. Example controller application input file 55
15. Example VASP input, POSCAR file with random atomic coordinates for Si7 cluster.... 56

CHAPTER 1

INTRODUCTION

Computational chemistry, a subfield of chemistry, has accelerated the discovery of new materials by replacing trial-and-error laboratory experimentation with simulated models of molecular structures. These simulations are first calculated using software packages such as the Vienna Ab-initio Simulation Package (VASP), then used to predict molecular properties of molecules of interest. Such properties can include the molecules' color, conductivity, hardness, melting/boiling points, luster, and malleability.

Computational models of molecular structures can require considerable effort to calculate. As early as 1929, Paul Dirac observed, “The underlying physical laws necessary for the mathematical theory of a large part of physics and the whole of chemistry are thus completely known, and the difficulty is only that the exact application of these laws leads to equations much too complicated to be soluble. It therefore becomes desirable that approximate practical methods of applying quantum mechanics should be developed, which can lead to an explanation of the main features of complex atomic systems without too much computation [1].”

In spite of subsequent advances in quantum mechanical theories and the development of high performance computers, calculations of even simple molecules can be time consuming. Researchers have tried to make these calculations more efficient with simplified equations of molecular behavior to obtain sufficiently accurate predictions of a molecule’s physical properties. Algorithms have been developed that speed the search for feasible configurations of molecules of interest by evaluating potential configurations for these molecules in parallel. Packages like VASP have been designed to run on parallel, distributed architectures like the East Tennessee State University (ETSU) Knightrider cluster, where VASP is primarily used for

researching nanoelectronic materials such as silicon clusters. Moreover, research into more efficient strategies for modeling molecules continues.

The Problem

This research investigated strategies for increasing the efficiency of a class of packages for exploring molecular structures. These packages are computer-aided molecular design (CAMD) packages that use evolutionary algorithms (EA) to generate and evaluate candidate structures and employ packages like VASP for fitness evaluations. Essentially, the packages use a series of independent, parallel computations to generate a final pool of candidate structures for a molecule of interest. Each round in this series of computations generates an intermediate pool of candidate structures, encoded as VASP-generated output files that rate each encoded structure's quality. After each non-final round of computations, these packages discard some the pool's less stable structures, using the remaining structures to initiate the next round of computation.

The work described here sought to identify computationally economical heuristics for eliminating unfavorable candidates from pools of candidate structures by terminating less favorable VASP simulations. As the execution times of VASP jobs tend to be long, eliminating poor candidates from pools of solutions early would reduce the number of computations whose relevance is low.

Approach

A VASP-based CAMD package, similar to ones in current use, was developed that generates possible structures for a molecule of interest. This package includes a controller task that uses VASP data to periodically cull a percentage of the lowest ranked structures from the pool of candidate solutions. This task uses one of three heuristics for culling these structures. The

first heuristic ranks evaluations by a molecule's total energy, favoring molecules with lower total free energy, which are more likely to be recreated in a particular environment. The second ranks evaluations by the rate of decrease between intermediate VASP steps, favoring candidate structures with high rates of decrease over molecules that were close to optimal; i.e., that had a lower chance for further improvement. The third ranks evaluations using a metric that combines a molecule's total energy (first heuristic) and speed of optimization (second heuristic). This heuristic attempted to balance molecular stability with rate of change.

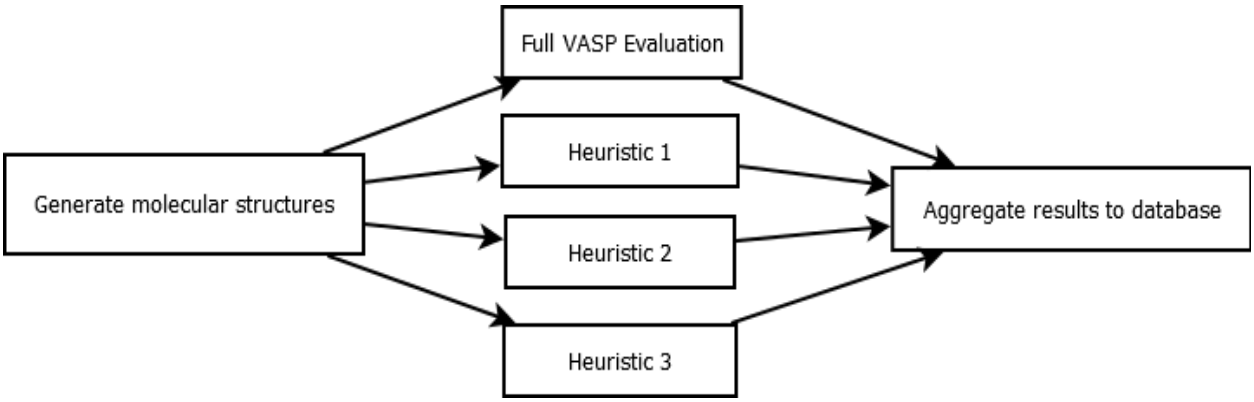


Figure 1. Controller application overview

To test these heuristics, the controller generated random atomic coordinates for candidate molecules and their respective VASP input files. The controller submitted the VASP simulations for the full VASP evaluations and each heuristic to Knightrider. After each VASP simulation, the controller logged VASP output to a database for analysis (Figure 1). To evaluate each heuristic's effectiveness, the results obtained from culling were compared with the results obtained with a full VASP optimization on each candidate structure in a starting pool.

Results

The heuristics decreased VASP simulation execution times, at the cost of discarding some molecules that would have performed better than those that were kept. Overall, no culling method appeared to be superior in terms of run time or accuracy. On a case-by-case basis, the optimization rate culling seemed to outperform the other methods by a very small margin when coupled with a more aggressive culling factor. The overall load on the system was reduced due to less time in VASP operations, but because the heuristics added a non-parallelizable dependency wall clock times were longer.

Overview

The remainder of this work is split into five sections. Section 2 is a background section that describes computer-aided molecular design and evolutionary computing. Section 3 examines the experimental framework and fitness approximations created for reducing VASP time. Section 4 presents the results of each fitness approximation. Section 5 analyzes and compares the heuristics and implications of use. Lastly, Section 6 reviews the work and suggests further research.

CHAPTER 2

BACKGROUND

Computer Aided Molecular Design

Molecular design is an area of study where new molecular structures are created or simulated to obtain materials with required properties [2]. Molecular design drives materials design research and has applications in electronic, biochemical, and pharmaceutical research. Traditional molecular design methods are expensive and time-consuming, requiring design, synthesis, and evaluation experiments in a laboratory. CAMD software, which uses computations to simulate atomic interaction, can be used to calculate optimal molecular geometries based on atomistic simulation. CAMD software advances research by replacing environments that are potentially difficult or expensive to replicate in a laboratory. By avoiding the need to manufacture candidate molecular structures, these simulations accelerate the discovery of previously unknown organic and inorganic structures.

Gani describes CAMD as the use of "a set of building blocks and a specified set of target properties [to] determine the molecule or molecular structure that matches these properties [3]." One such target property is a molecule's total energy: molecules with a lower free energy level, closer to ground state, are more stable. Stable molecules are more likely to be physically fabricated after simulation. CAMD algorithms attempt to optimize a molecule's geometric configuration by defining structures of lowest total energy.

Molecular structure optimization is difficult for complex molecules. Increasing the number of atoms in a system of molecules greatly increases the number of possible geometries for that system, creating a complex, non-linear search space. Venkatasubramanian et al. discuss several CAMD techniques for structural optimization, including random search, heuristic enumeration,

mathematical programming, knowledge-based systems, and graphical reconstruction [4]. Though these approaches have been successful with smaller molecular systems, they do not scale well to larger molecular geometries that have potentially huge search spaces. For larger search spaces, adaptive systems such as EAs have proved useful.

Evolutionary Algorithms

EAs identify solutions to problems by emulating the effect of natural evolution on a population of candidate solutions [5]. This involves the repeated perturbation of the population, using operations that evaluate, modify, combine elements of, or remove solutions from the population, based in part on predefined criteria that rate the desirability, or fitness, of the population's members.

EAs work well with large search spaces. They can be adapted to many different simulation problems from various fields of study [3]. They can be tailored to achieve tradeoffs between breadth and depth of search based on techniques for balancing between randomization of search and prioritization of likely areas for good solutions.

In general, EAs follow these steps (Figure 2).

1. Initialize a population with random candidate solutions
2. Evaluate each candidate
3. Repeat until a termination condition is satisfied
    a. Select parents
    b. Combine or mutate the resulting offspring
    c. Evaluate new candidates' fitness
    d. Select individuals for the next generation
    e. Check termination condition

4. End [5]

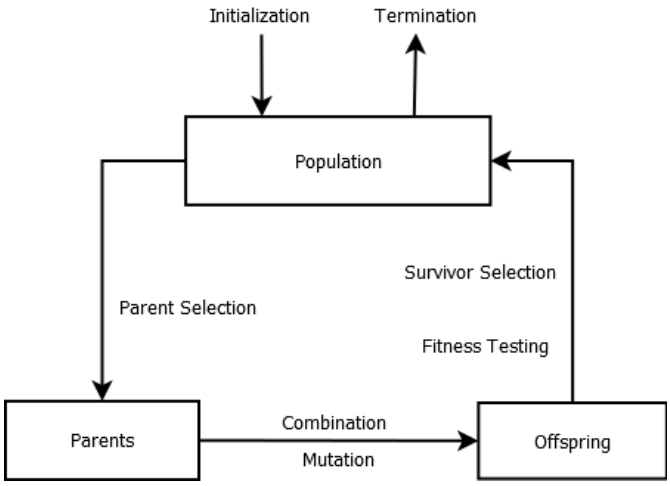


Figure 2. The general scheme of an EA as flow chart

EAs attempt to evolve an initial population of individuals to obtain better individuals, as measured by a fitness function [5]. These individuals, known as genotypes, consist of characteristic attributes, called phenotypes, which fitness functions use to determine a genotype's quality. A genotype with a higher fitness is a solution better adapted to a problem's requirements. Genotypes are evolved by changing their constituent phenotypes. These changes are implemented by operations that select genotypes for further evolution, mutate their phenotypes, and recombine phenotypes from pairs of genotypes to obtain new genotypes.

A survivor selection procedure promotes candidate genotypes to the next round of mutation or parent selection. This procedure attempts to balance the need to identify novel genotypes with the need to preserve a quality population. Choosing only the fittest parents each round would cause the EA to converge too quickly, without exposing many candidate solutions before arriving at an acceptable fitness.

Survivors selected from the population pass on traits through combination (multiple parents) or mutation (single parent). The resulting genotypes are offspring for new populations. This

process repeats until enough optimal candidates have been generated or another predefined termination condition has been met.

Termination conditions for EAs vary by application. Some common conditions for termination limit CPU time, total number of fitness evaluations, minimum rate of fitness improvement, and minimum population diversity [5]. In addition, an EA could be signaled to terminate when an optimum has been reached, as determined by a well-defined set of criteria.

Fitness Heuristics

Fitness heuristics are computationally inexpensive approximations to more computationally expensive fitness functions. One such approximation, described by Hemberg et al., was employed by ECStar, an evolutionary system that predicts changes in intensive care unit patients' arterial blood pressure [6]. ECStar uses a database called MIMIC that stores encoded physiological signals (electrocardiogram, arterial blood pressure, and pulmonary artery pressure) and documented clinical data from previous ICU settings. These data are used for inferential analysis to compare fitness candidates to previous scenarios.

ECStar distributes the work of computing successive generations of candidate solutions among multiple, volunteer compute nodes, which use spare CPU cycles to run ECStar's EA processes. This design allows cheaper commodity systems like home desktops to contribute to large scale processing. While this model is cost effective, its efficiency is limited by technical considerations such as limited RAM and disk space, number of volunteered compute cycles, and uncontrollable network latency.

Given the size and complexity of the MIMIC data and ECStar's unpredictable resources, fitness approximations were used to reduce the overall load on the system and time in fitness calculations. Hemberg et al.'s approach iteratively evaluated and removed potentially

underachieving candidates from the population during complex calculations based on various intermediate fitness evaluation metrics. Removing individuals from the pool before a fitness evaluation was completed reduced the overall system load. However, the strategy's failure to test all candidates equally resulted in some higher-value candidates being discarded prematurely because their intermediate fitness score was not fully representative of their final score when evaluated, leading to an imprecise fitness ranking. For complex, time-consuming fitness evaluations, the efficiency gained from using a heuristic could be worth introducing some imprecision into fitness ranking.

Evolutionary CAMD

CAMD EAs create new molecular geometries by evolving interatomic distances and bond angles within a molecule [1]. EAs have been integrated with quantum dynamic packages. These packages simulate atomic interaction, optimize molecular geometries, and drive evaluative fitness testing.

EAs in CAMD have been used to discover a variety of organic and inorganic molecular structures [7]. Rata et al. describe an EA that optimizes silicon (Si) clusters, which are small Si isomers in the range of approximately 5-100 atoms in size [8]. Oganov et al.'s Universal Structure Predictor: Evolutionary Xtallography (USPEX) method specifically targets the discovery of new crystalline structures [9]. Wong et al. present a more generalized method, EvoMD, which explores a broader range of chemical composition such as new drugs [2].

VASP

VASP is a software package used for atomistic modeling and examining materials; it is often used in CAMD EAs [10]. Given a molecular structure's geometric representation and a representation of its environment, VASP uses density functional theory [11] to approximate that

structure's energy. VASP can also optimize structures by geometrically reconfiguring them to produce a lower ground state energy molecule. Throughout a VASP simulation, free energy decreases as a molecular geometry is relaxed to a ground state. A molecule's free energy can vary considerably with just small changes to its geometric configuration; this makes the potential energy surface jagged, with many local minima. VASP will optimize a molecule to a local minimum; the goal then of an EA using VASP would be to find the global minimum.

VASP's functionality is useful in evolutionary CAMD suites that use free energy as fitness criteria. The Universal Structure Predictor: Evolutionary Xtallography (USPEX) method, which predicts stable crystalline structures, uses VASP to obtain ground state energies during evaluation and to stabilize or optimize parent structures evaluation [12]. VASP based fitness tests generally optimize all candidate geometries before ranking a population by the criterion of total free energy; the majority of time spent in fitness evaluation is in VASP simulations [13].

CHAPTER 3

METHODS

Goals

Fitness tests using VASP are time consuming and resource intensive. Given similar resources and starting molecules as the experiments described in this work, full VASP simulations run on average 35 minutes. In some cases, VASP simulations took much longer and had not completed after 10 hours. The number of atoms in a molecule, the population size, initial molecular geometric configurations, and the availability of computational resources all affect VASP run times. Minimizing the time spent in VASP simulations is favorable for EAs or any large-scale energy landscape search running on limited or unpredictable computational resources. This will reduce overall run times and the impact on shared resources on smaller computer clusters.

Some EAs, such as USPEX, discard the lowest ranked members of a population after fitness testing, selecting all parents from a top ranked percentage of candidates [13]. With this approach, computational resources are wasted on VASP simulations of molecules that do not survive to reproduce. Using a heuristic to automatically discard underachieving molecules during fitness testing would reduce the number of VASP simulations and/or improve the value of the simulations being performed.

During a VASP execution, intermediate results are written to output files. From these files, one can determine each molecule's current free energy and the rate at which its energy is decreasing. This research sought to use these values to identify and remove underperforming molecules from a population during fitness testing in an attempt to reduce time in VASP. Given

a population of molecules, the heuristic should identify a user-defined percentage of top ranked candidates and spend less time in VASP simulations.

In the most effective culling, the worst candidate structures would be removed early leaving the best to spend the most time in VASP. Since early calculations of a molecule's energy can't necessarily be used to predict a molecule's final energy, removing simulations before they are fully optimized by VASP will tend to remove some potentially high value molecules along with poorer ones. The extent to which this tradeoff between speed and accuracy is acceptable is a consideration end users will need to balance for their own needs or application.

Experimental Design

Three fitness heuristics were developed for this study as well as a procedure for computing a control population for comparing each heuristics' efficiency and effectiveness. The control procedure uses VASP to optimize all of an initial population's candidate molecules then ranks each based on lowest free energy. The control procedure does not remove underperforming molecules from the population during fitness testing. This method is computationally intensive due to the time spent in VASP calculations for each molecule.

Ideally, a fitness heuristic should produce approximately the same set of top ranked, optimized molecular structures as the control population in less VASP run time. Given the random nature of EA recombination, an approximate ranking should not affect the efficacy of an EA using a heuristic. The heuristics' accuracy would need to be considered when designing a survival selection mechanism, which would select candidates for reproduction based on the heuristics' outcomes. The problem of determining how choice of heuristics should influence strategies for evolving new candidate molecules, however, is outside the scope of this study.

The three heuristics that this study investigated were intended to identify top ranked candidates. Criteria for ranking candidates included a molecule's free energy, a molecule's average decrease in free energy per ionic step, and a hybrid of these two metrics, as follows:

- The first, energy value culling (EVC) heuristic ranks molecules based on free energy. The EVC heuristic removes higher energy, less stable molecules from populations. Free energy is a good indicator of a molecule's stability and regularly used as a quality indicator. This heuristic is intended to naively use the same quality indicator used in fitness ranking for intermediate quality scoring for culling.
- The second, optimization rate culling (ORC) heuristic ranks molecules based on the average amount of energy decrease per ionic step. The ORC heuristic removes molecules with less energy decrease from populations. Overall, VASP simulations with a higher average energy decrease per ionic step have lower final free energies. This method is intended to investigate average energy decrease per ionic step as an effective alternative intermediate quality indicator for culling.
- The third, energy and optimization score culling (EOSC) heuristic ranks molecules based on a score obtained by multiplying performance scores for energy and optimization rate. These scores are determined by rating a molecule's performance in each dimension relative to those of other molecules in the pool. Each score is scaled from 1 (worst) to 2 (best) then multiplied to get a score from 1 (highest energy, slowest optimization rate) to 4 (lowest energy, fastest optimization rate). EOSC-based culling removes high-energy molecules that are also not decreasing in energy much per ionic step.

To test the heuristics, randomly generated molecular structures were used for test groups. The molecules used for testing, small silicon clusters, are well-documented structures with known optimal geometries. A controller application automated all VASP simulations and then stored the VASP results and performance metrics to a database for further analysis.

Implementation

Computing Platform

The heuristics' performance was tested on ETSU's Knightrider computer cluster. Knightrider currently supports 45 nodes, with each node having two Intel Xeon six core X5650 2.6 GHz CPUs. Each node has 12 GB of memory, 1 GB dedicated to each processor core. Nodes are linked via an InfiniBand low latency network connection. Knightrider's operating system is Rocks Cluster Distribution 5.4.

Controller Application

A controller application was developed to simplify testing. The application generates random molecule populations, builds VASP IO files, monitors VASP simulations, polls for VASP simulation state changes, and writes the output data to a database. The application was written using the Java SE Development Kit 7 and uses a SQLite database. It generates scripts to interface with the Moab Resource Manager to submit VASP simulations and query their execution state.

Initialization

The controller application's initialization component generates random atomic coordinates for each population's molecular candidates and creates scripts to run each VASP simulation on Knightrider. The application uses a configuration file (Appendix 7.4) to create a simulation

group object that holds setup data for initialization and heuristic testing processes. The configuration file specifies the following parameters for simulation group generation.

- Workspace path – a directory to host VASP input and output files
- Simulation name – a simulation name or ID that is used as a directory name
- Molecule count – the number of molecules in a population
- Atom count – the number of atoms in each molecule generated
- Minimum atomic distance – the minimum distance allowed between two atoms being generated for testing (angstroms)
- Maximum atomic distance – the maximum distance allowed between two atoms being generated for testing (angstroms)
- Compute node count – the number of compute nodes to use on the computer cluster
- Process count – the number of processes to spawn per node
- Maximum wall time – the maximum amount of time a VASP simulation is allowed to run
- Max ionic steps – the maximum number of intermediate results (*ionic steps*) that each VASP optimization may generate

Random Molecule Generation

To test the heuristics, an initialization algorithm was created to generate test molecules as VASP input files. The algorithm generates random coordinates for a molecule's constituent atoms, which are represented using an array of coordinate vectors relative to a central atom. The algorithm places the molecule's other atoms in random locations around this central atom. The three-dimensional Cartesian coordinates are stored in a VASP input file for processing.

The initialization algorithm uses a method, generateNeighboringAtom, to populate a molecule's atoms. This method accepts three inputs: the coordinates of a reference atom and the

minimum and maximum distance in angstroms from the reference atom at which the neighboring atom should be generated. The algorithm places the reference atom at the center of a unit sphere with a radius of 1 angstrom. It generates a random point for the new atom on the sphere's surface, and then positions the atom along the vector from the reference atom to the random point at a random distance within the minimum and maximum distance range. This random distance is calculated using a Java implementation of the Mersenne Twister [14].

Figure 3 shows a generic range as defined by the reference atom and distance range. The distance range will change for various atom types and is specified by the user.

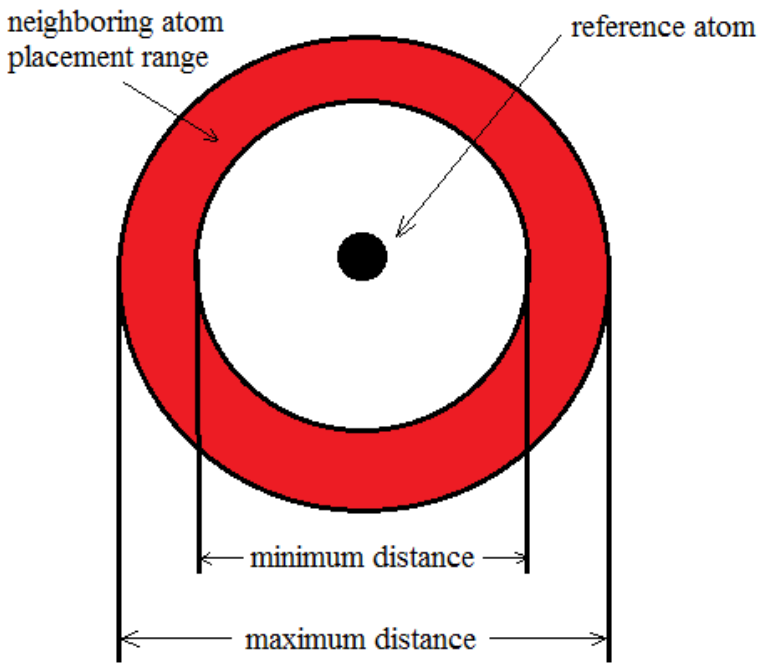


Figure 3. Two-dimensional representation of generateNeighboringAtom method

The algorithm starts by placing the central atom at coordinate (0,0,0). It then uses generateNeighboringAtom to generate N-1 more atoms. After each new coordinate is generated, the algorithm determines if the newly generated atom is too close to a preexisting atom in the coordinate set based on the specified distanced range. If so, the function continues to produce

random coordinates until an appropriate atom is generated or there is no space left for a new coordinate. If no space is left, the process is restarted with the central atom. The process concludes when N atoms are generated, all of which are positioned at an appropriate distance from each other.

Since this algorithm uses a central reference point, it generates molecules that are compact. Figure 4 illustrates this algorithm; the reference atom (0,0,0) is marked with a white dot.

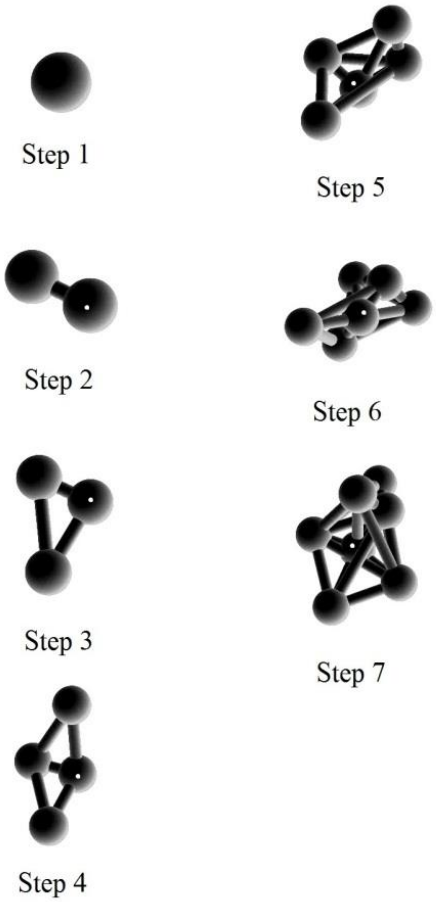


Figure 4. Random coordinate generation

Silicon Clusters

Tests were initialized with seven-atom silicon (Si7) clusters. Silicon clusters are a good test candidate because they have been studied extensively and the most stable geometries are well known. Silicon clusters are of great scientific interest because silicon is the most important electronic material. It is important to understand how one can modify silicon crystals, guided by the idea of creating novel forms of silicon that are of higher electronic efficiency.

Heuristic Testing

The controller application initializes a population and then ranks each population using the control method and each fitness heuristic (simulation group). Upon completion, the controller application logs the simulation group's data to the database including molecule energies, ionic steps, and total time spent in VASP operations.

Control Method

The control method provides the most accurate rank based on free energy, as it fully optimizes a population's geometries to a stable ground state energy. The controller application submits and monitors each molecule in the population to a VASP simulation pool. After full optimization of each molecule, the population is ranked by free energy. The ranking results, final optimized molecular geometries, and performance metrics are then stored to a database.

Fitness Heuristics

The controller application executes VASP optimizations for each molecule and monitors the VASP simulation pool. The controller evaluates VASP jobs at intervals, culling underperforming simulations from the job pool. The culling heuristics use four variables to control the simulations' behavior.

- Initial ionic steps – the initial number of ionic steps to perform before a culling action
- Steps per round – after the initial ionic steps, the number of ionic steps to run before another culling action
- Round kill percentage (RKP) – the percentage of molecules to cull each round. The percentage of VASP simulations to terminate and discard each round
- Optimization percentage – the percentage of top ranked candidate molecules that will be fully optimized

The heuristics analyze the progress of VASP simulations at user defined ionic step intervals over several rounds (Figure 5). A round consists of VASP simulations that run for a user defined number of steps and then culls a percentage of the jobs from the VASP simulation pool.

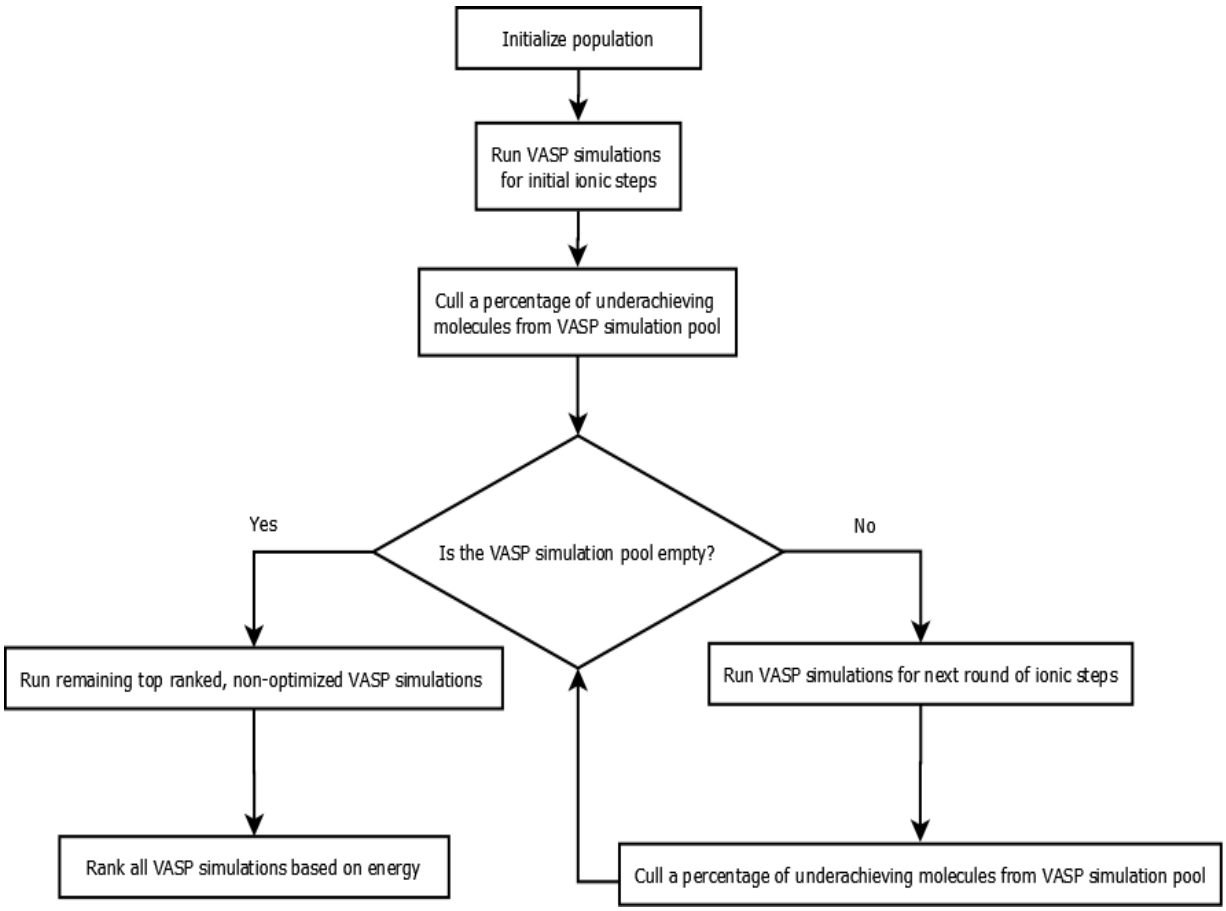


Figure 5. Heuristic control flow diagram

In each test of the fitness heuristics, the controller runs all VASP simulations for a specified number of initial ionic steps before any culling occurs. This gives candidate geometries more time to stabilize before culling, thereby providing more accurate performance measurements for the first and subsequent culling actions. After the initial ionic steps, a percentage of underperforming molecules is removed (round kill percentage) from the VASP simulation pool. This ends the first round.

The VASP simulations then run for a determined number of ionic steps (steps per round) before another culling action. This process repeats until the maximum number of ionic steps is reached or the pool is emptied. If any simulations have not reached the maximum number of ionic steps, the molecules are ranked once more based on the approximation's performance metric and the top ranked molecules are resubmitted to continue optimization. Once fully optimized, all molecules are ranked based on free energy.

Performance

The heuristics' improvement on VASP run time was evaluated by comparing each heuristic's time in VASP to the control procedure's time. The heuristics' accuracy was calculated by comparing each heuristic's top ranked candidates to the control population's top ranked candidates. More accurate approximations placed more top candidates in the control population's list of top ranked candidates.

Rank	Control Method	Fitness Heuristic
1	Candidate 1	Candidate 1
2	Candidate 2	Candidate 2
3	Candidate 3	Candidate 4
4	Candidate 4	Candidate 5
5	Candidate 5	Candidate 3
6	Candidate 6	Candidate 8
7	Candidate 7	Candidate 7
8	Candidate 8	Candidate 6
Optimization Percent = 50%		
Overall Accuracy = 75%		

Table 1. Accuracy calculation example

A representative accuracy calculation is depicted in Table 1. In this example, the optimization percent is 50%, so candidates ranked 5-8 would be discarded. The fitness approximation correctly identified three of the candidates from the control method's top candidates (candidates 1, 2, and 4) but incorrectly ranked candidate 5. Since three of four top ranked candidates were identified, the fitness approximation is considered 75% accurate.

The test configurations described in chapter 4 attempted to maximize heuristic accuracy and minimize time in VASP by adjusting the aggressiveness in which VASP simulations are removed from a pool, which is controlled by the round kill percentage variable. RKP was tested with three different values to see how RKP affected run time and accuracy.

CHAPTER 4

RESULTS

The full results for each test configuration are given in Appendix 7.3. The appendix lists all time measurements in seconds but were rounded to hours for this section

Test Configurations

For this research, the heuristic tests were run with three different simulation group configurations. The major differences in each configuration are highlighted in Table 2 below.

	Value		
Parameter	**Test Configuration 1**	**Test Configuration 2**	**Test Configuration 3**
workspace_path	si7	si7	si7
sim_name	si7-compact	si7-compact	si7-compact
num_molecules	100	40	40
num_atoms	7	7	7
min_distance	1	1	1
max_distance	2	2	2
num_nodes	2	3	3
num_processes	12	12	12
walltime	10:00:00	10:00:00	10:00:00
max_ionic_steps	500	500	500
initial_ionic_steps	8	8	8
steps_per_round	2	2	2
round_kill_percentage	100	50	75
optimization_percentage	50	50	50

Table 2. Simulation group test configurations

The RKP parameter was tested with values of 100, 75, and 50 percent. Setting RKP to 100% removed all structures from the testing pool just after the initial ionic steps, yielding a quick estimate of molecular quality. Then the highest ranked (optimization percent) structures were resubmitted to continue for full VASP optimization. This left lower ranked structures only partially optimized. Test configurations 2 and 3 were less aggressive, culling 50 and 75 percent

of the testing pool per round, respectively. This let the VASP simulations continue longer, giving each molecular candidate more time in VASP.

Initially it was planned to only alter the RKP for each test configuration. Test configuration 1 ran VASP simulations for 100 test structures in each simulation group, each simulation processing on two compute nodes. Due to time and shared resource constraints on Knightrider, test configurations 2 and 3 were reduced to 40 molecular candidates but given three nodes per VASP simulation. This reduced the overall workload for heuristic testing for these two configurations.

Test Configuration 1

The control procedures spent a combined total of 1160 hours in VASP calculations. The EVC approximation methods ran with a total combined time in VASP of 729 hours, a 37% reduction in time. The ORC approximation methods ran with a total combined time in VASP of 715 hours, a 38% reduction in time. The EOSC approximation methods ran with a total combined time in VASP of 721 hours, a 38% reduction in time. The average accuracy percentage for EVC, ORC, and EOSC were 67%, 68%, and 67%, respectively. On average, each fitness approximation ran in about 2/3 the time as the control procedure. Overall, the approximations were about 2/3 as accurate as the control procedure (Table 3).

	Energy Value Culling		**Optimization Rate Culling**		**Energy-Optimization Score Culling**	
	Accuracy %	**Time %**	**Accuracy %**	**Time %**	**Accuracy %**	**Time %**
Average	67%	63%	68%	62%	67%	62%
Best	74%	44%	76%	41%	78%	40%
Worst	60%	111%	62%	97%	60%	111%

Table 3. Test configuration 1 overall results

Test Configuration 2

The control procedures spent a combined total of 277 hours in VASP calculations. The EVC approximation methods ran with a total combined time in VASP of 195 hours, a 29% reduction in time. The ORC approximation methods ran with a total combined time in VASP of 175 hours, a 36% reduction in time. The EOSC approximation methods ran with a total combined time in VASP of 202 hours, a 27% reduction in time. The average accuracy percentage for EVC, ORC, and EOSC were 66%, 64%, and 59%, respectively. On average, each fitness approximation ran in about 7/10 the time of the control procedure. Overall, the approximations were about 5/8 as accurate as the control procedure (Table 4).

	Energy Value Culling		**Optimization Rate Culling**		**Energy-Optimization Score Culling**	
	Accuracy %	**Time %**	**Accuracy %**	**Time %**	**Accuracy %**	**Time %**
Average	66%	71%	64%	64%	59%	73%
Best	85%	29%	75%	24%	70%	26%
Worst	50%	160%	50%	168%	50%	176%

Table 4. Test configuration 2 overall results

Test Configuration 3

The control procedures spent a combined total of 296 hours in VASP calculations. The EVC approximation methods ran with a total combined time in VASP of 139 hours, a 53% reduction in time. The ORC approximation methods ran with a total combined time in VASP of 137 hours, a 54% reduction in time. The EOSC approximation methods ran with a total combined time in VASP of 153 hours, a 48% reduction in time. The average accuracy percentage for EVC, ORC, and EOSC were 67%, 68%, and 67%, respectively. On average, each fitness approximation ran in about 1/2 the time as the control procedure. Overall, the approximations were about 2/3 as accurate as the control procedure (Table 5).

	Energy Value Culling		Optimization Rate Culling		Energy-Optimization Score Culling	
	Accuracy %	**Time %**	**Accuracy %**	**Time %**	**Accuracy %**	**Time %**
Average	67%	47%	68%	46%	67%	52%
Best	80%	29%	75%	27%	75%	26%
Worst	60%	106%	50%	89%	55%	91%

Table 5. Test configuration 3 overall results

CHAPTER 5

ANALYSIS

Run Times

While heuristic computations spent less time overall in VASP simulations, in some instances they required more time to complete than the control procedures. Analysis was difficult because run times could have been affected by Knightrider's overall network load during testing. Baker notes that network communication accounts for most of VASP's run time [15]. To further analyze the heuristics' impact on VASP run time, tests should be run on a dedicated cluster or with a module that records network load during each test.

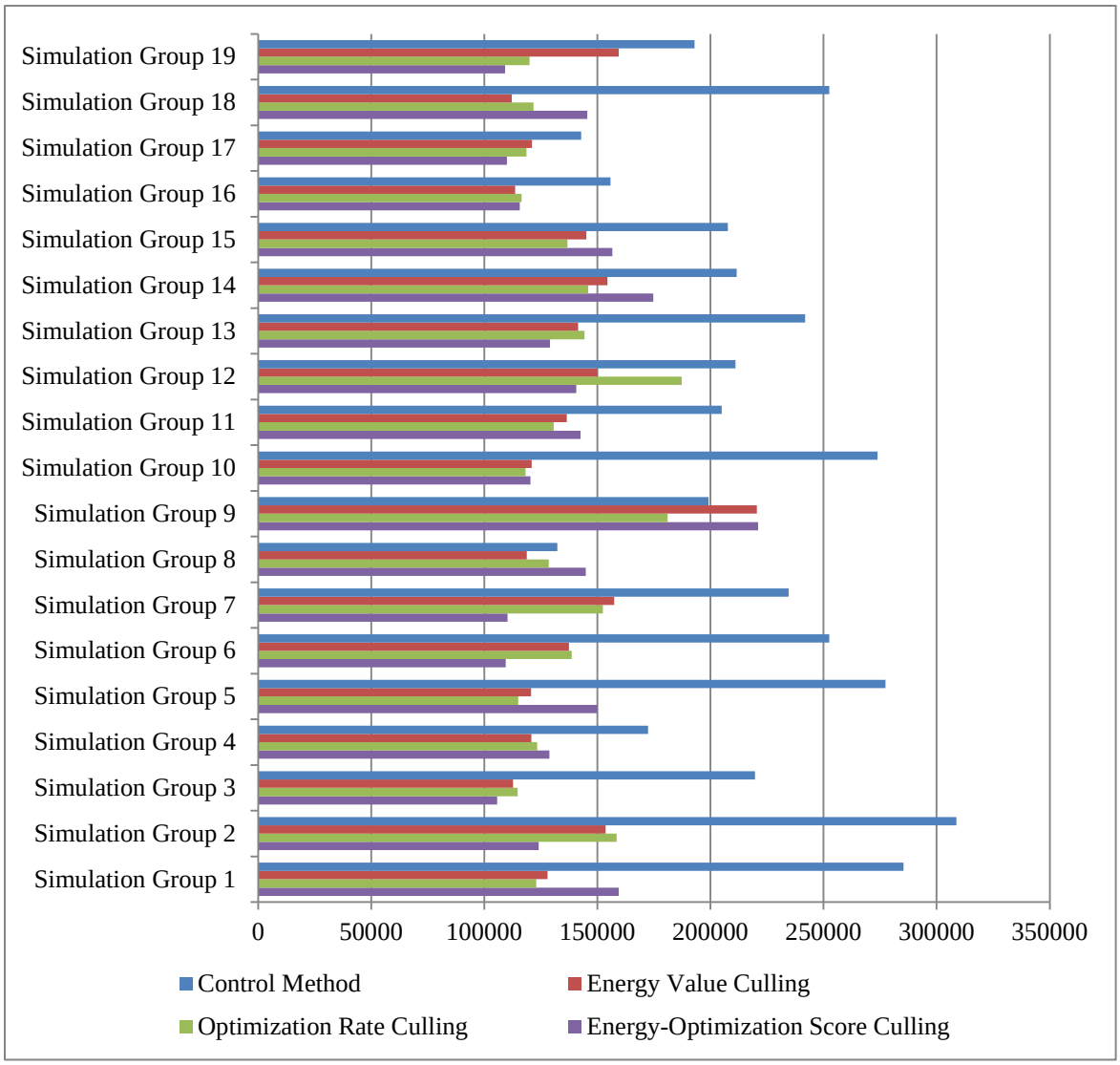


Figure 6. Test configuration 1 - control method vs. heuristics' VASP time (seconds)

Test configuration 1's VASP run times (Figure 6) were mostly consistent over each simulation group. Out of 57 VASP simulation pools (19 simulations groups * 3 heuristics), the heuristics' VASP run times were longer than the control method's in three instances. The EOSC heuristic ran longer twice and the EVC method once. Despite the EOSC method running longer in two cases, it did outperform the other methods in 8 out of 19 simulation groups; ORC had the best time in 7 out of 19 groups, and EVC was only the best in 4 out of 19 groups.

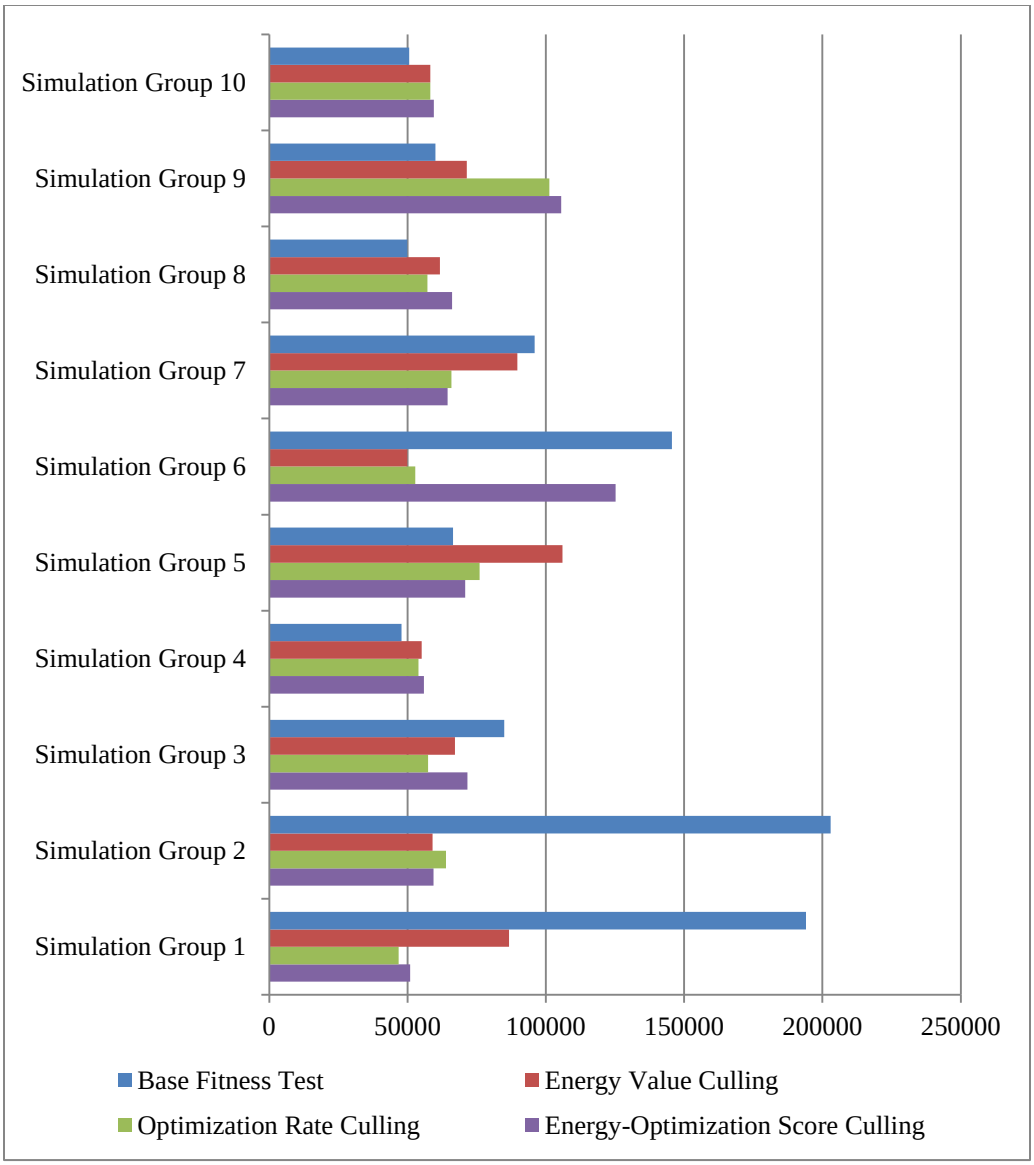


Figure 7. Test configuration 2 - control method vs. heuristics' VASP time (seconds)

Test configuration 2's VASP run times (Figure 7) were inconsistent compared to test configuration 1. Each heuristic ran longer than the control method in five cases for a total of fifteen of 30 VASP simulation groups. An analysis of the VASP files found no indication that these inconsistencies resulted from erroneous input. EVC and ORC tied for the best performance in 4 out of 10 simulation groups while EOSC only outperformed the others in two cases.

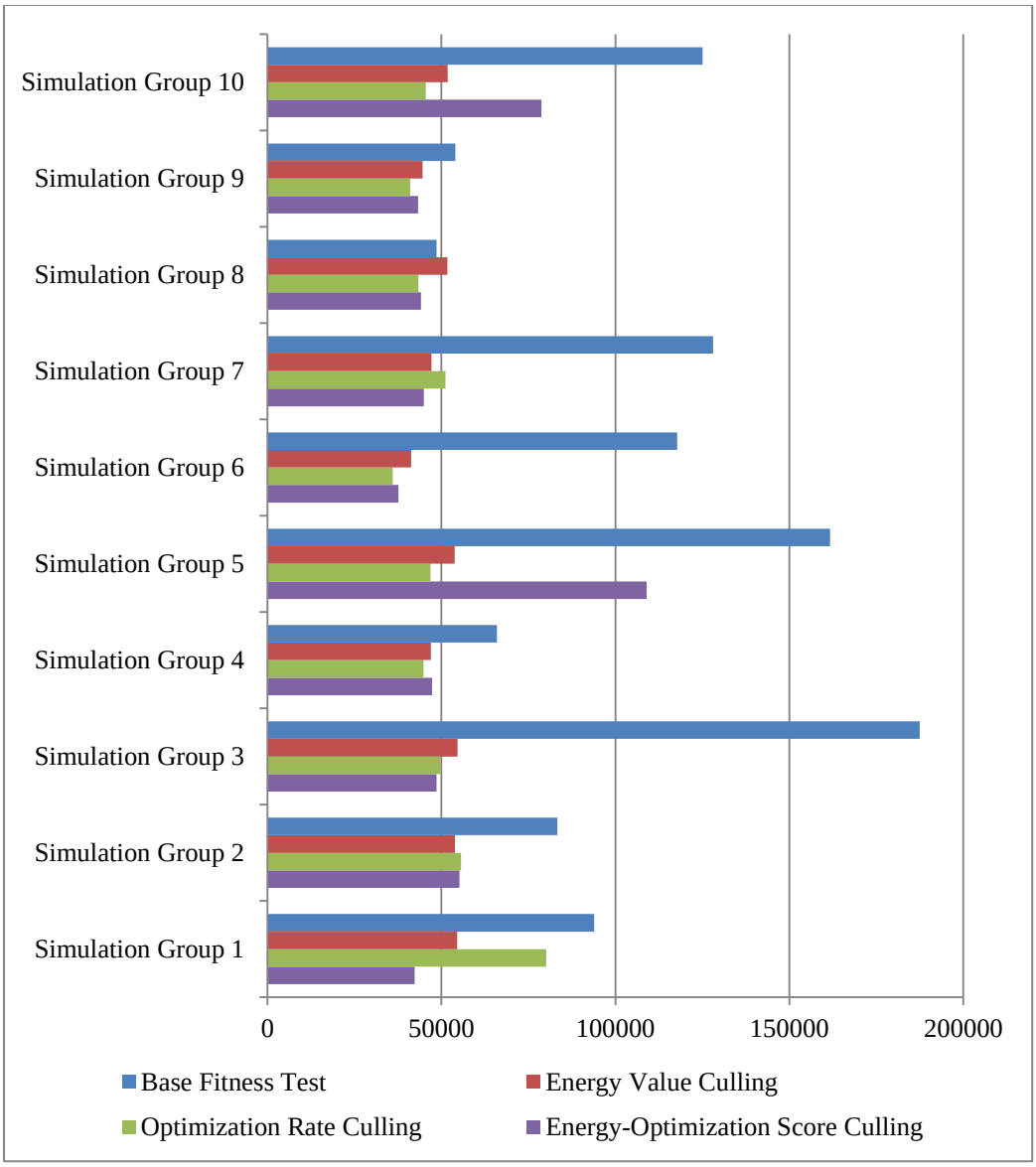


Figure 8. Test configuration 3 - control method time vs. heuristics' VASP time (seconds)

Test configuration 3's VASP run times were the most consistent across each simulation group (Figure 8). Only one VASP simulation pool out of 30 ran longer than the control method, the EVC method. The ORC method outperformed the others in 6 out of 10 simulation groups; EOSC had the best time in 3 out of 10 groups, and EVC was only the best in 1 out of 10 groups.

Accuracy

Heuristic accuracies ranged from 59% to 68% over each test configuration. Each heuristic's accuracy varied over each simulation group from around 10% to 30%. Some variance was expected and is most likely due to variations in the initial populations' quality. If a population has more candidates closer to an optimal structure, then the heuristics are more likely to make better guesses early. On average, the EVC, ORC, and EOSC heuristics performed about the same in each test configuration.

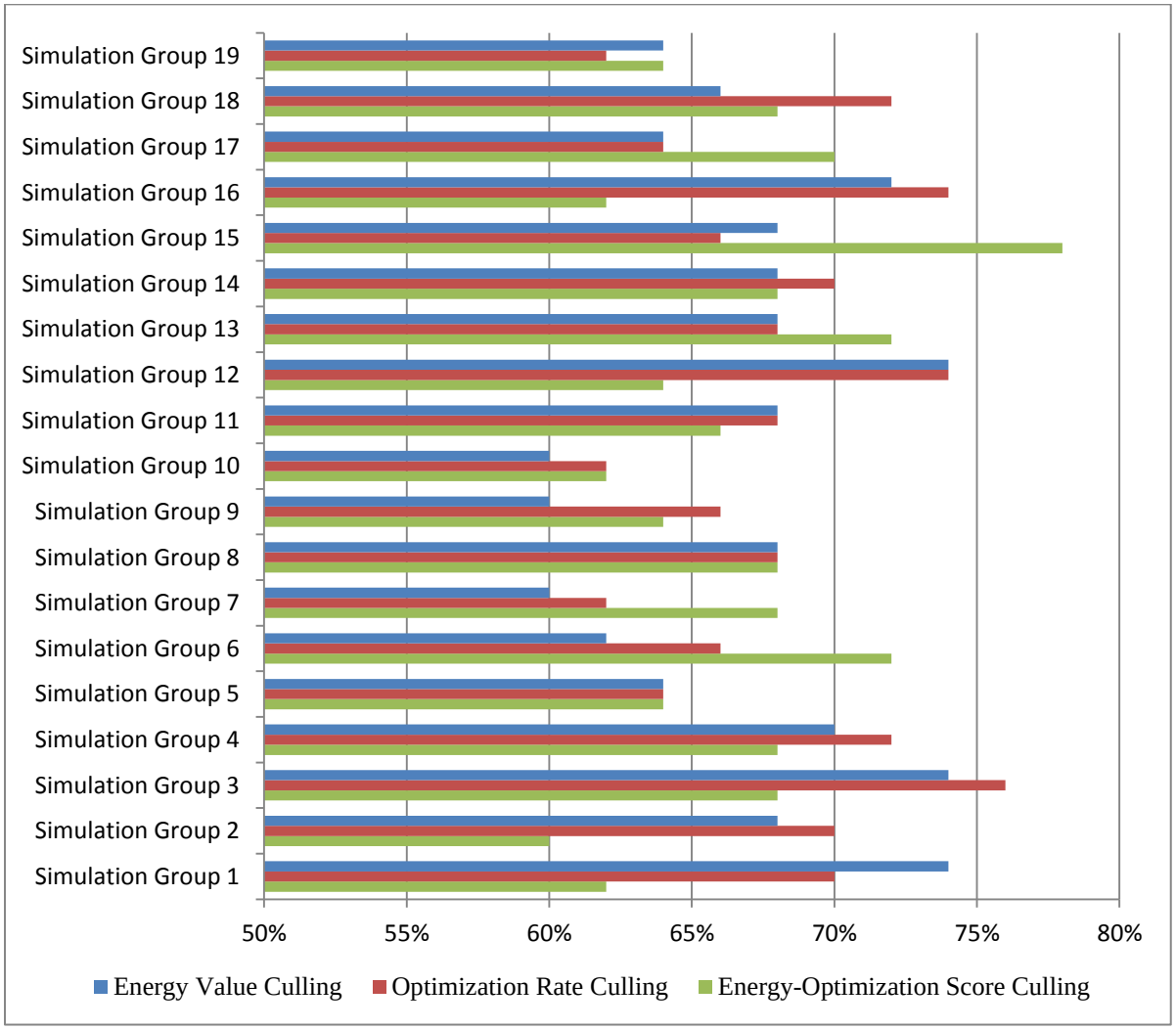


Figure 9. Test configuration 1 - heuristics' accuracy percentage per simulation group

Overall, test configuration 1's heuristics (Figure 9) were about 67% accurate with a median accuracy percentage of 68%. Per simulation group, the ORC was more accurate more times than EVC and EOSC, but not by a very high margin. In 12 of 19 simulation groups, the ORC method had the highest accuracy percentage or tied for highest accuracy; EOSC was 9 of 19, and EVC was 6 of 19.

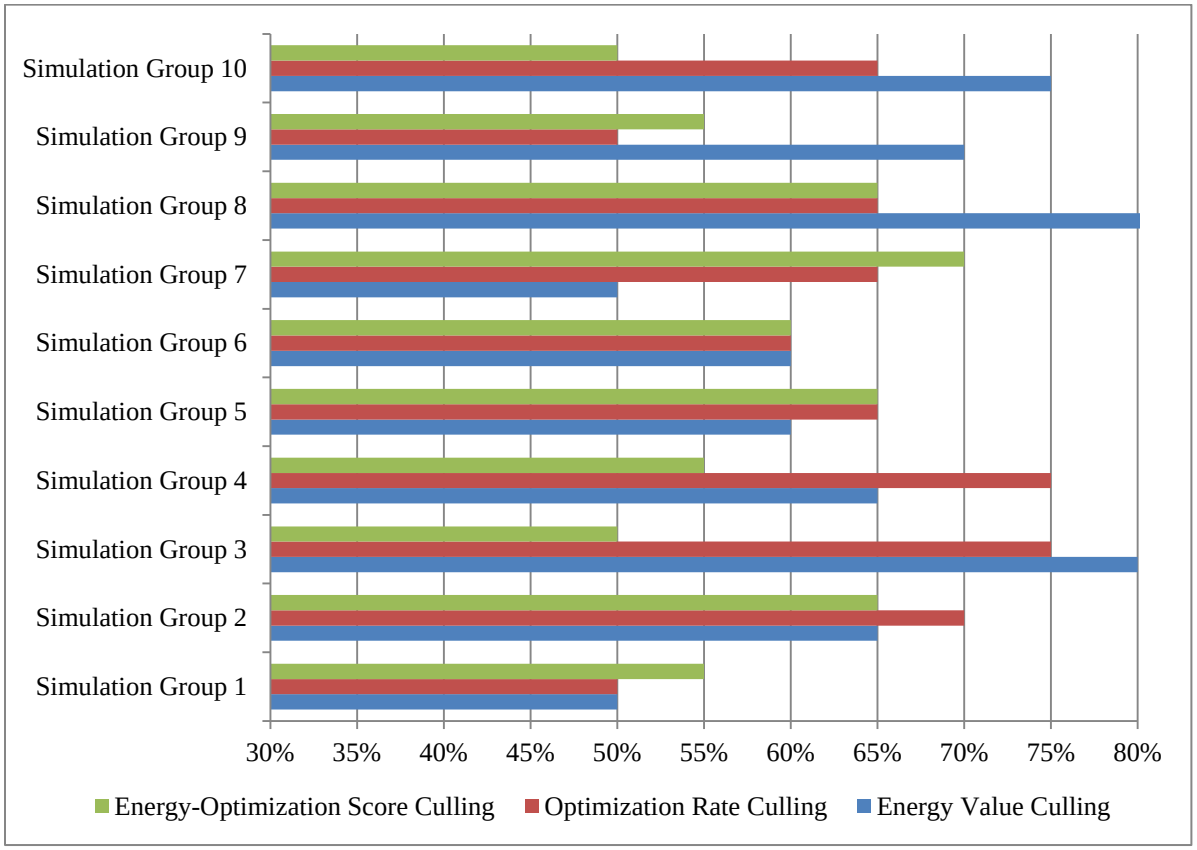


Figure 10. Test configuration 2 - heuristics' accuracy percentage per simulation group

On average, test configuration 2's heuristics (Figure 10) were about 63% accurate with median accuracy percentages of 65%, 65%, and 58% for EVC, ORC, and EOSC methods, respectively. Each method had the highest accuracy or tied for the highest in about half the simulation groups with no method proving to be more accurate overall.

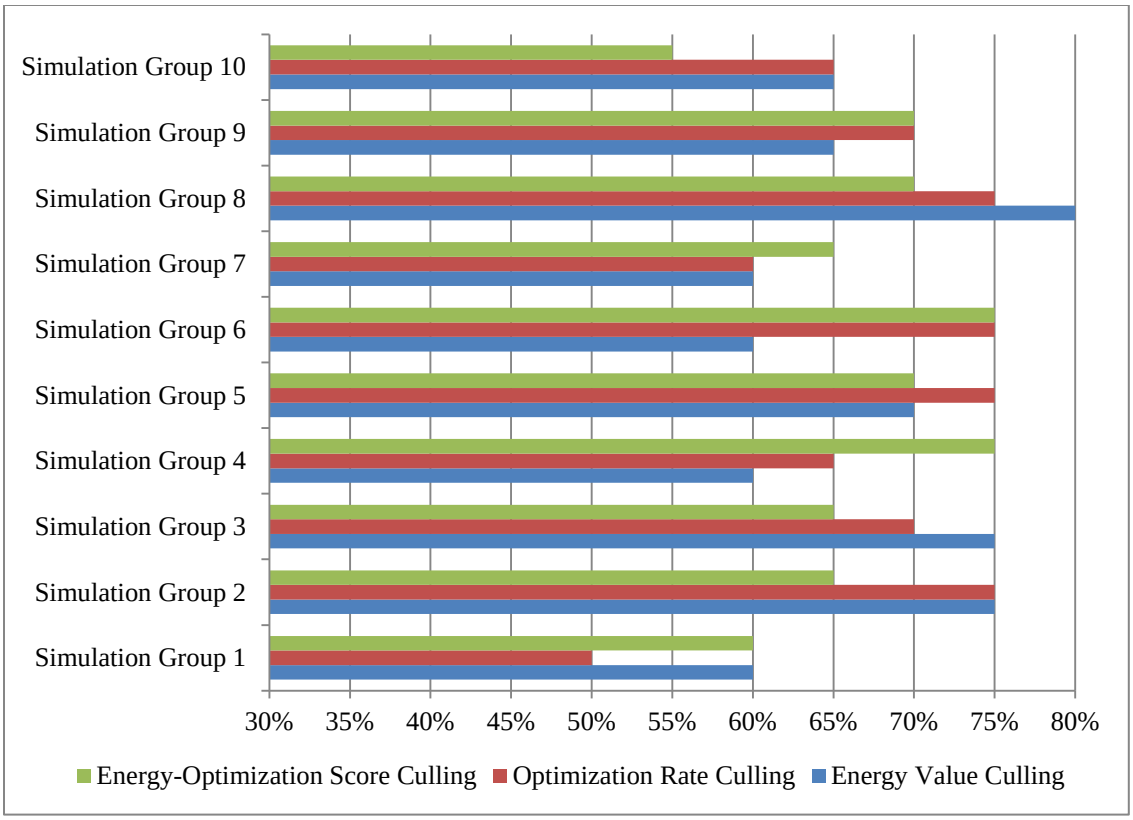


Figure 11. Test configuration 3 - heuristics' accuracy percentage per simulation group

On average, test configuration 3's heuristics (Figure 11) were about 67% accurate with median accuracy percentages of 65%, 70%, and 68% for EVC, ORC, and EOSC methods, respectively. Like test configuration 2, each method had the highest accuracy or tied for the highest in about half the simulation groups with no method showing superior accuracy.

Overall Performance

On average, test configurations 1 and 3 spent less time in VASP. This can most likely be attributed to using a more aggressive RKP (Table 6). Test configuration 2 was less aggressive and therefore spent more time in VASP operations, as it culled fewer simulations per round.

		Energy Value Culling		Optimization Rate Culling		Energy-Optimization Score Culling	
		Accuracy %	**Time %**	**Accuracy %**	**Time %**	**Accuracy %**	**Time %**
Test Configuration 1 (RKP = 100%)	***Average***	67%	63%	68%	62%	67%	62%
	Best	74%	44%	76%	41%	78%	40%
	Worst	60%	111%	62%	97%	60%	111%
Test Configuration 2 (RKP = 50%)	***Average***	66%	71%	64%	64%	59%	73%
	Best	85%	29%	75%	24%	70%	26%
	Worst	50%	160%	50%	168%	50%	176%
Test Configuration 3 (RKP = 75%)	***Average***	67%	47%	68%	46%	67%	52%
	Best	80%	29%	75%	27%	75%	26%
	Worst	60%	106%	50%	89%	55%	91%

Table 6. Test configuration comparison

The ORC method performed slightly better per simulation group than EVC and EOSC (ignoring any possible effects of RKP). The ORC method had the lowest run times in 17 simulation groups and the best accuracy or tied in 21 groups. Of these groups, ORC was the most accurate and fastest running in ten cases. The EOSC method was only slightly less consistent than ORC. The EOSC method had the lowest run times in 13 simulations groups and the best accuracy or tied in 18 groups. The EOSC method was the fastest and most accurate in nine cases. The EVC method was the fastest and most accurate in five instances.

Parallelization

Despite a decrease in VASP calculation time, overall wall clock time for the heuristics in most cases were longer than the control method. To rank each candidate fairly, the heuristics executed culling steps only after each candidate reached the same number of ionic steps. In the case where some VASP simulations couldn't run concurrently, as was the case for these trials, the culling step added a non-parallelizable dependency that forced the controller application to wait for simulations to finish before allowing jobs to optimize further. Along with the non-

parallelizable portions, starting and stopping the VASP simulations added overhead, resulting in longer wall clock time and slightly longer individual VASP times.

Using these heuristics on computer clusters with sufficient resources to run all jobs in parallel should result in faster fitness evaluations and a reduced overall impact on resources. Users of the heuristics on smaller computer clusters would still benefit from the reduced load of VASP simulations, but at the cost of longer wall clock times. This would still be desirable in the case where an administrator has limited the amount of processing time or cycles one can use in a given time span.

Si7 Clusters

Throughout heuristic testing, many intermediate forms of Si7 were observed, though Si7's optimal geometric configuration was never found. Since this work only focused on small, random populations' initialization and energy rankings, finding the optimal configuration was not expected. This geometry would likely be found if a complete EA process was executed.

The most consistently observed optimal Si7 structures identified are essentially fusions between two known structures [16] as seen in Figure 12.

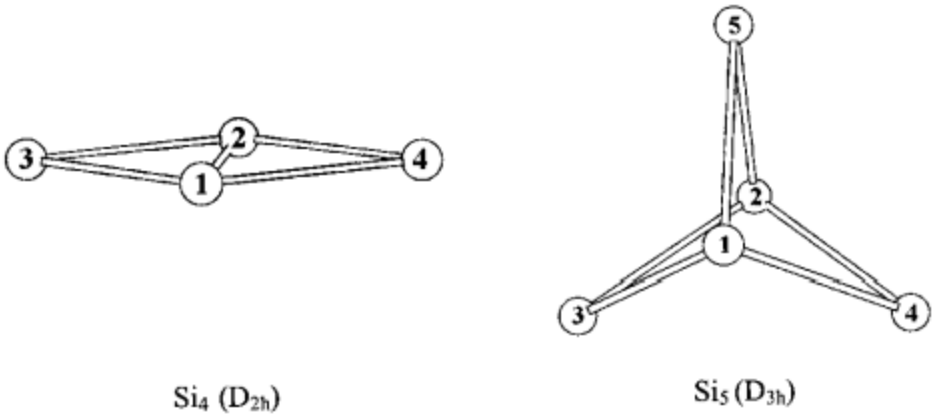


Figure 12. Si4 and Si5 geometric configurations

This fusion consists of a Si4 (planar rhombus) and a Si5 (double pyramid), shown below (Figure 13), which were derived from both control method and heuristics.

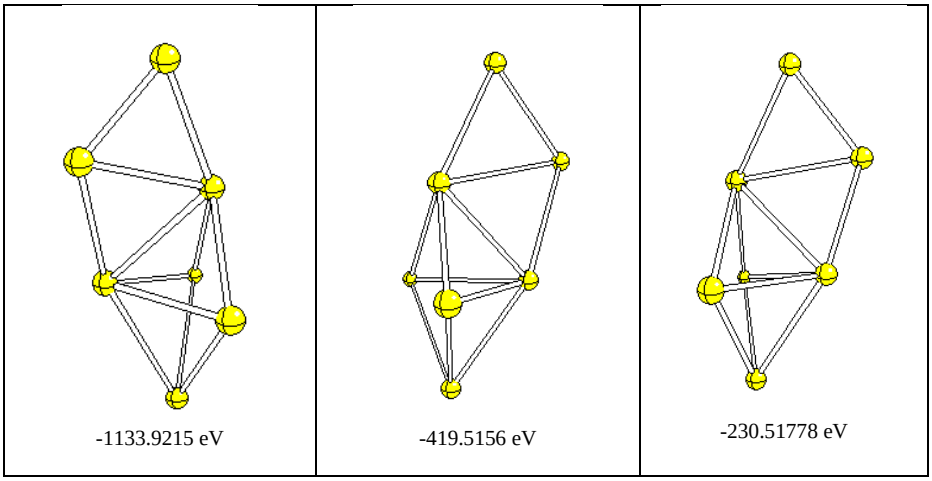


Figure 13. Si7 structures identified with control method and heuristics

CHAPTER 6

CONCLUSION

The three heuristics that were tested in this research were intended to reduce VASP simulation times when ranking molecular populations by a free energy. These heuristics employed a culling mechanism to remove lower value simulations from a VASP simulation pool, based on three different metrics for evaluation: free energy, optimization rate, and a score produced from the previous metrics. No metric proved wholly superior to any other. The ORC heuristic was slighter better per simulation group in terms of both speed and accuracy but only by a small margin. A more aggressive RKP seemed to decrease VASP simulation time without a major effect on accuracy.

Future studies of these algorithms should test the heuristics in isolation on a dedicated cluster to remove possible anomalies in test results related to network load. Given that Knightrider is a shared resource, this might not be feasible at ETSU. Additional modules could be developed that log network load so this factor can be considered when comparing heuristic run time results to the control method. Moving the studies to a computer cluster with more resources would also be favorable to test the heuristics' ability to speed up EA calculations to mitigate the effects of the added non-parallelizable code regions. In addition, the heuristics should be tested with various molecular components, such as larger silicon clusters, or clusters of different basic elements.

REFERENCES

[1] P. A. M. Dirac, "Series A, Containing Papers of a Mathematical and Physical Character," *Proceedings of the Royal Society of London,* vol. 123, no. 792, 1929.

[2] S. S. Wong, W. Luo and K. C. Chan, "EvoMD: An Algorithm for Evolutionary Molecular Design," *IEE/ACM Transactions on Computational Biology and Bioinformatics,* pp. 987-1003, 2011.

[3] R. Gani , "CAMD: Computer Aided Molecular Design – Examples of Applications," 2004. [Online]. Available: http://www.capec.kt.dtu.dk/documents/overview/camd-overview-2004.pdf. [Accessed 15 September 2014].

[4] V. Venkatasubramanian, K. Chan and J. M. Caruthers, "Evolutionary Design of Molecules with Desired Properties," *Journal of Chemical Information and Computer Sciences,* vol. 35, no. 2, pp. 188-195, 1995.

[5] A. E. Eiben and J. E. Smith, Introduction to Evolutionary Computing, Berlin: Springer-Verlag, 2003.

[6] E. Hemberg, K. Veeramachaneni, F. Dernoncourt, M. Wagy and U.-M. O'Reilly, "Imprecise Selection and Fitness Approximation in a Large-Scale Evolutionary Rule Based System for Blood Pressure Prediction," in *Genetic and Evolutionary Computation Conference Companion*, Amsterdam, 2013.

[7] D. E. Clark and D. R. Westhead, "Evolution Algorithms In Computer-Aided Molecular Design," *Journal of Computer-Aided Molecular Design,* vol. 10, pp. 337-358, 1996.

[8] I. Rata, A. A. Shvartsburg, M. Horoi, T. Frauenheim, K. M. Siu and K. A. Jackson, "Single-Parent Evolution Algorithm and the Optimization of Si Clusters," *Physical Review Letters,* vol. 85, no. 3, pp. 546-549, 2000.

[9] A. R. Oganov, A. O. Lyakhov and M. Valle, "How Evolutionary Crystal Structure Prediction Works and Why," *Accounts of Chemical Research,* pp. 227-237, 2010.

[10] J. Furthmüller, J. Hafner and G. Kresse, "Ab-initio calculation of the structural and electronic properties of carbon and boron nitride using pseudopotentials," *Physical Review B,* no. 50, 9 September 1994.

[11] R. Parr and W. Young, Density-Functional Theory of Atoms and Molecules, Oxford: Oxford University Press, 1994.

[12] A. O. Lyakhov, A. R. Oganov and M. Valle, "How to predict very large and complex crystal structures," *Computer Physics Communication,* no. 181, pp. 1623-1632, 2010.

[13] A. R. Oganov, Modern Methods of Crystal Structure Prediction, Weinheim: WILEY-VCH Verlag & Co. KGaA, 2011.

[14] S. Luke, "George Mason University Department of Computer Science," [Online]. Available: http://www.cs.gmu.edu/~sean/research/mersenne/. [Accessed 17 October 2014].

[15] M. B. Baker, "A Study of Improving the Parallel Performance of VASP," 2010. [Online]. Available: http://dc.etsu.edu/etd/1739.

[16] C. Xiao, F. Hagelberg and W. A. Lester, "Geometric, energetic, and bonding properties of neutral and charged copper-doped silicon clusters," *Physical Review B,* vol. 66, no. 7, p. 075425, 2002.

APPENDICES

APENDIX A
SELECTED TERMS

Atomistic simulation – methods that model molecular structures at the atomic level

Vienna Ab-Initio Software Package (VASP) – a program used for modelling molecular structures on an atomic scale, performing electronic structure calculations and quantum-mechanical molecular dynamics

Ionic step – intermediate steps in a VASP optimization

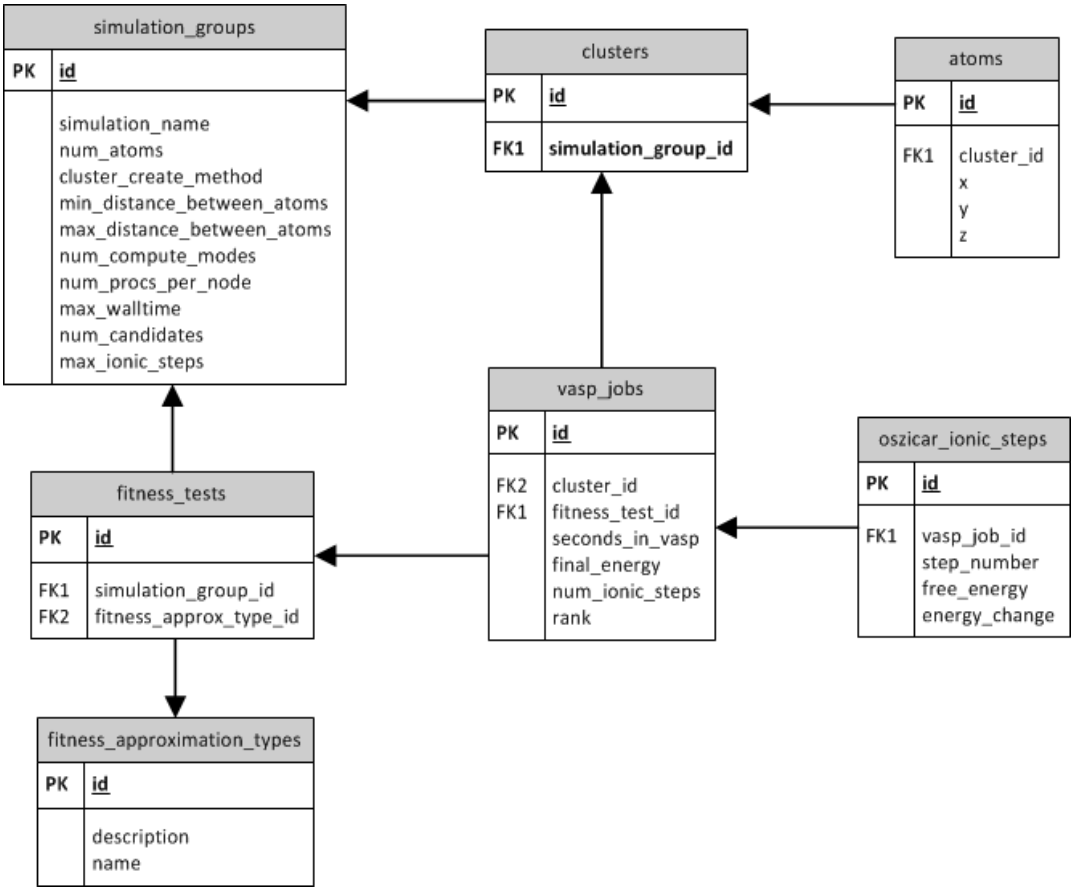
Nanoelectronic materials – nanotechnology, matter in dimensions of approximately 1-100 nanometers, used in electronic components

Density functional theory – a method for modeling the electronic structure and ground states of molecular structure, which is used by VASP

Ground state energy – the lowest energy state of a quantum mechanical system

Potential energy surface – the ground state energies for every geometric configuration of a quantum mechanical system

APPENDIX B
DATABASE SCHEMA



APENDIX C
EXPERIMENTAL RESULTS

	Energy Value Culling	Optimization Rate Culling	Energy-Optimization Score Culling
Simulation Group 1	74%	70%	62%
Simulation Group 2	68%	70%	60%
Simulation Group 3	74%	76%	68%
Simulation Group 4	70%	72%	68%
Simulation Group 5	64%	64%	64%
Simulation Group 6	62%	66%	72%
Simulation Group 7	60%	62%	68%
Simulation Group 8	68%	68%	68%
Simulation Group 9	60%	66%	64%
Simulation Group 10	60%	62%	62%
Simulation Group 11	68%	68%	66%
Simulation Group 12	74%	74%	64%
Simulation Group 13	68%	68%	72%
Simulation Group 14	68%	70%	68%
Simulation Group 15	68%	66%	78%
Simulation Group 16	72%	74%	62%
Simulation Group 17	64%	64%	70%
Simulation Group 18	66%	72%	68%
Simulation Group 19	64%	62%	64%
Average	67%	68%	67%

Table 7. Test configuration 1 - accuracy percentage per heuristic and simulation group

	Control Method	Energy Value Culling	Optimization Rate Culling	Energy-Optimization Score Culling
Simulation Group 1	285308	127969	123018	159428
Simulation Group 2	308683	153595	158495	124104
Simulation Group 3	219638	112726	114697	105682
Simulation Group 4	172366	120750	123315	128759
Simulation Group 5	277325	120683	115044	150045
Simulation Group 6	252513	137406	138578	109460
Simulation Group 7	234555	157434	152444	110280
Simulation Group 8	132308	118732	128494	144791
Simulation Group 9	199205	220472	181063	221063
Simulation Group 10	273803	120978	118271	120453
Simulation Group 11	205020	136526	130601	142590
Simulation Group 12	211024	149985	187296	140664
Simulation Group 13	241788	141482	144303	129005
Simulation Group 14	211567	154432	145989	174699
Simulation Group 15	207614	145101	136742	156583
Simulation Group 16	155814	113685	116449	115729
Simulation Group 17	142836	121122	118613	110037
Simulation Group 18	252545	112157	121813	145568
Simulation Group 19	192887	159467	120047	109118
Total Time	4176799	2624702	2575272	2598058
% of Base Test Time	100%	63%	62%	62%

Table 8. Test configuration 1 - time in VASP (seconds) per simulation group and heuristic

	Energy Value Culling	Optimization Rate Culling	Energy-Optimization Score Culling
Simulation Group 1	45%	43%	56%
Simulation Group 2	50%	51%	40%
Simulation Group 3	51%	52%	48%
Simulation Group 4	70%	72%	75%
Simulation Group 5	44%	41%	54%
Simulation Group 6	54%	55%	43%
Simulation Group 7	67%	65%	47%
Simulation Group 8	90%	97%	109%
Simulation Group 9	111%	91%	111%
Simulation Group 10	44%	43%	44%
Simulation Group 11	67%	64%	70%
Simulation Group 12	71%	89%	67%
Simulation Group 13	59%	60%	53%
Simulation Group 14	73%	69%	83%
Simulation Group 15	70%	66%	75%
Simulation Group 16	73%	75%	74%
Simulation Group 17	85%	83%	77%
Simulation Group 18	44%	48%	58%
Simulation Group 19	83%	62%	57%

Table 9. Test configuration 1 - each heuristic's percentage of time in VASP relative to control method's VASP time

	Energy Value Culling		Optimization Rate Culling		Energy-Optimization Score Culling	
	Accuracy %	**Time %**	**Accuracy %**	**Time %**	**Accuracy %**	**Time %**
Simulation Group 1	74%	45%	70%	43%	62%	56%
Simulation Group 2	68%	50%	70%	51%	60%	40%
Simulation Group 3	74%	51%	76%	52%	68%	48%
Simulation Group 4	70%	70%	72%	72%	68%	75%
Simulation Group 5	64%	44%	64%	41%	64%	54%
Simulation Group 6	62%	54%	66%	55%	72%	43%
Simulation Group 7	60%	67%	62%	65%	68%	47%
Simulation Group 8	68%	90%	68%	97%	68%	109%
Simulation Group 9	60%	111%	66%	91%	64%	111%
Simulation Group 10	60%	44%	62%	43%	62%	44%
Simulation Group 11	68%	67%	68%	64%	66%	70%
Simulation Group 12	74%	71%	74%	89%	64%	67%
Simulation Group 13	68%	59%	68%	60%	72%	53%
Simulation Group 14	68%	73%	70%	69%	68%	83%
Simulation Group 15	68%	70%	66%	66%	78%	75%
Simulation Group 16	72%	73%	74%	75%	62%	74%
Simulation Group 17	64%	85%	64%	83%	70%	77%
Simulation Group 18	66%	44%	72%	48%	68%	58%
Simulation Group 19	64%	83%	62%	62%	64%	57%

Table 10. Test configuration 1 - accuracy percentage and VASP time percentage per heuristic and simulation group

	Energy Value Culling	Optimization Rate Culling	Energy-Optimization Score Culling
Simulation Group 1	50%	50%	55%
Simulation Group 2	65%	70%	65%
Simulation Group 3	80%	75%	50%
Simulation Group 4	65%	75%	55%
Simulation Group 5	60%	65%	65%
Simulation Group 6	60%	60%	60%
Simulation Group 7	50%	65%	70%
Simulation Group 8	85%	65%	65%
Simulation Group 9	70%	50%	55%
Simulation Group 10	75%	65%	50%
Average	66%	64%	59%

Table 11. Test configuration 2 - accuracy percentage per heuristic and simulation group

	Control method	Energy Value Culling	Optimization Rate Culling	Energy-Optimization Score Culling
Simulation Group 1	193995	86659	46707	50891
Simulation Group 2	202918	59065	63928	59378
Simulation Group 3	84922	67113	57397	71654
Simulation Group 4	47781	55043	53936	55867
Simulation Group 5	66401	105991	76010	70811
Simulation Group 6	145570	49928	52747	125203
Simulation Group 7	95980	89733	65849	64451
Simulation Group 8	49857	61655	57141	66097
Simulation Group 9	60103	71379	101231	105508
Simulation Group 10	50629	58173	60849	59501
Total Time	998156	704739	633119	729361
% of Base Test Time	100%	71%	64%	73%

Table 12. Test configuration 2 - time in VASP (seconds) per simulation group and heuristic

	Energy Value Culling	Optimization Rate Culling	Energy-Optimization Score Culling
Simulation Group 1	45%	24%	26%
Simulation Group 2	29%	32%	29%
Simulation Group 3	79%	68%	84%
Simulation Group 4	115%	113%	117%
Simulation Group 5	160%	114%	107%
Simulation Group 6	34%	36%	86%
Simulation Group 7	93%	69%	67%
Simulation Group 8	124%	115%	133%
Simulation Group 9	119%	168%	176%
Simulation Group 10	115%	115%	118%

Table 13. Test configuration 2 - each heuristic's percentage of time in VASP relative to control method's VASP time

	Energy Value Culling		Optimization Rate Culling		Energy-Optimization Score Culling	
	Accuracy %	Time %	Accuracy %	Time %	Accuracy %	Time %
Simulation Group 1	50%	45%	50%	24%	55%	26%
Simulation Group 2	65%	29%	70%	32%	65%	29%
Simulation Group 3	80%	79%	75%	68%	50%	84%
Simulation Group 4	65%	115%	75%	113%	55%	117%
Simulation Group 5	60%	160%	65%	114%	65%	107%
Simulation Group 6	60%	34%	60%	36%	60%	86%
Simulation Group 7	50%	93%	65%	69%	70%	67%
Simulation Group 8	85%	124%	65%	115%	65%	133%
Simulation Group 9	70%	119%	50%	168%	55%	176%
Simulation Group 10	75%	115%	65%	115%	50%	118%

Table 14. Test configuration 2 - accuracy percentage and VASP time percentage per heuristic and simulation group

	Energy Value Culling	Optimization Rate Culling	Energy-Optimization Score Culling
Simulation Group 1	60%	50%	60%
Simulation Group 2	75%	75%	65%
Simulation Group 3	75%	70%	65%
Simulation Group 4	60%	65%	75%
Simulation Group 5	70%	75%	70%
Simulation Group 6	60%	75%	75%
Simulation Group 7	60%	60%	65%
Simulation Group 8	80%	75%	70%
Simulation Group 9	65%	70%	70%
Simulation Group 10	65%	65%	55%
Average	67%	68%	67%

Table 15. Test configuration 3 - accuracy percentage per heuristic and simulation group

	Control Method	Energy Value Culling	Optimization Rate Culling	Energy-Optimization Score Culling
Simulation Group 1	93953	54519	80081	42298
Simulation Group 2	83314	53882	55689	55145
Simulation Group 3	187495	54630	49812	48587
Simulation Group 4	65942	46993	44907	47348
Simulation Group 5	161689	53820	46880	108949
Simulation Group 6	117725	41334	36019	37689
Simulation Group 7	128117	47171	51195	44924
Simulation Group 8	48636	51715	43425	44101
Simulation Group 9	53964	44592	41021	43325
Simulation Group 10	125086	51845	45468	78747
Total Time	1065921	500501	494497	551113
% of Base Test Time	100%	47%	46%	52%

Table 16. Test configuration 3 - time in VASP (seconds) per simulation group and heuristic

	Energy Value Culling	Optimization Rate Culling	Energy-Optimization Score Culling
Simulation Group 1	58%	85%	45%
Simulation Group 2	65%	67%	66%
Simulation Group 3	29%	27%	26%
Simulation Group 4	71%	68%	72%
Simulation Group 5	33%	29%	67%
Simulation Group 6	35%	31%	32%
Simulation Group 7	37%	40%	35%
Simulation Group 8	106%	89%	91%
Simulation Group 9	83%	76%	80%
Simulation Group 10	41%	36%	63%

Table 17. Test configuration 3 - each heuristic's percentage of time in VASP relative to control method's VASP time

	Energy Value Culling		Optimization Rate Culling		Energy-Optimization Score Culling	
	Accuracy %	**Time %**	**Accuracy %**	**Time %**	**Accuracy %**	**Time %**
Simulation Group 1	60%	58%	50%	85%	60%	45%
Simulation Group 2	75%	65%	75%	67%	65%	66%
Simulation Group 3	75%	29%	70%	27%	65%	26%
Simulation Group 4	60%	71%	65%	68%	75%	72%
Simulation Group 5	70%	33%	75%	29%	70%	67%
Simulation Group 6	60%	35%	75%	31%	75%	32%
Simulation Group 7	60%	37%	60%	40%	65%	35%
Simulation Group 8	80%	106%	75%	89%	70%	91%
Simulation Group 9	65%	83%	70%	76%	70%	80%
Simulation Group 10	65%	41%	65%	36%	55%	63%

Table 18. Test configuration 3 - accuracy percentage and VASP time percentage per heuristic and simulation group

APPENDIX D
EXAMPLE INPUT

```
#   Controller Application Setup
#
#   workspace_path: specifies the path where starting files are located
#   sim_name: naming convention for output
#   scalar: diagonal scaling factor from POSCAR
#   num_molecules: number of molecular clusters to generate
#   num_atoms: number of atoms per molecular cluster
#   min_distance: minimum distance between atoms in molecular cluster
#   max_distance: maximum distance between atoms in molecular cluster
#   num_nodes: the number of nodes to use on the computer cluster
#   num_processes: the number of processes to use per node
#   walltime: maximum run time limit
#   max_ionic_steps: maximum number of VASP ionic steps (INCAR NSW)
#   optimization_percentage: percent of structures to optimize

                # <field>:<value>
                workspace_path          : si7
                sim_name                : si7-compact
                scalar                  : 20
                num_molecules           : 10
                num_atoms               : 7
                min_distance            : 1.0
                max_distance            : 2.0
                num_nodes               : 1
                num_processes           : 12
                walltime                : 10:00:00
                max_ionic_steps         : 500
                initial_ionic_steps     : 8
                steps_per_round         : 2
                round_kill_percentage   : 100
                optimization_percentage : 50
```

Figure 14. Example controller application input file

```
Si7
   1.00000000000000
    20.0000000000000000    0.0000000000000000    0.0000000000000000
     0.0000000000000002   20.0000000000000000    0.0000000000000000
     0.0000000000000000    0.0000000000000000   20.0000000000000000
   7
Selective dynamics
Direct
0.0     0.0     0.0      T T T
-0.059572623124852196   0.04995751784664064     -0.00786834777036218     T T T
-0.03380536394511013    -0.056556469975857934   0.013960753948900803     T T T
0.012692412449119622    0.06378425170280404     -0.07487525828966725     T T T
-0.01635915872687561    0.0010421378334538373   0.08364507554828447      T T T
0.010297344068598932    -0.041867024685194945   0.06495782538827385      T T T
-0.06885216739237023    -0.001036334976327195   -0.032223336035243716    T T T

  0.00000000E+00  0.00000000E+00  0.00000000E+00
  0.00000000E+00  0.00000000E+00  0.00000000E+00
  0.00000000E+00  0.00000000E+00  0.00000000E+00
  0.00000000E+00  0.00000000E+00  0.00000000E+00
  0.00000000E+00  0.00000000E+00  0.00000000E+00
  0.00000000E+00  0.00000000E+00  0.00000000E+00
  0.00000000E+00  0.00000000E+00  0.00000000E+00
```

Figure 15. Example VASP input, POSCAR file with random atomic coordinates for Si7 cluster

VITA

JOHN NORMAN MCMEEN, JR.

Education:	Honors Diploma, Elizabethton High School, 2002
	B.S. Computer Science, East Tennessee State University, 2010
	M.S. Computer Science, East Tennessee State University, 2014
Professional Experience:	Computer Lab Manager (2007-2010), Graduate Assistant (2010-2012) - Office of Information Technology East Tennessee State University. Johnson City, TN
	Graduate Assistant (2010), Adjunct Lecturer (2012-2013) - Department of Computing, East Tennessee State University. Johnson City, TN
	Software Developer (2012-2013) - Resort Travel and Xchange. Asheville, NC
	Software Developer (2013-2014) - Advanced Call Center Technologies. Johnson City, TN