

2016

Computational Doping for Fuel Cell Material Design Based on Genetic Algorithms and Genetic Programming

Emrah Atilgan
University of South Carolina

Follow this and additional works at: <https://scholarcommons.sc.edu/etd>

 Part of the [Computer Sciences Commons](#), and the [Engineering Commons](#)

Recommended Citation

Atilgan, E.(2016). *Computational Doping for Fuel Cell Material Design Based on Genetic Algorithms and Genetic Programming*. (Doctoral dissertation). Retrieved from <https://scholarcommons.sc.edu/etd/3576>

This Open Access Dissertation is brought to you by Scholar Commons. It has been accepted for inclusion in Theses and Dissertations by an authorized administrator of Scholar Commons. For more information, please contact dillarda@mailbox.sc.edu.

COMPUTATIONAL DOPING FOR FUEL CELL MATERIAL DESIGN BASED ON
GENETIC ALGORITHMS AND GENETIC PROGRAMMING

by

Emrah Atilgan

Bachelor of Science
Eskisehir Osmangazi University, 2006

Master of Science
University of South Carolina, 2009

Submitted in Partial Fulfillment of the Requirements

For the Degree of Doctor of Philosophy in

Computer Science and Engineering

College of Engineering and Computing

University of South Carolina

2016

Accepted by:

Jianjun Hu, Major Professor

John R. Rose, Committee Member

Jijun Tang, Committee Member

Homayoun Valafar, Committee Member

Frank Chen, Committee Member

Lacy Ford, Senior Vice Provost and Dean of Graduate Studies

© Copyright by Emrah Atilgan, 2016
All Rights Reserved

DEDICATION

To my late father and my wonderful mother...

ACKNOWLEDGEMENTS

My grateful appreciation goes to my advisor, Dr. Jianjun Hu from the Computer Science and Engineering Department at the University of South Carolina. Without his great help, I might have not finished my dissertation. I would like to thank to my committee members, Dr. Rose, Dr.Tang, Dr. Valafar and Dr. Chen. Their valuable suggestions helped me to make a better job. I also would like to thank to Chair of Computer Science and Engineering Department, Dr.Matthews, for being a great mentor during my study.

I would like to thank to my beautiful wife and daughter for being in my life and supporting me during this stressful time.

Thanks to my big family in Turkey for praying for me to get my PhD degree. They never let me feel like I am alone and thousands miles away from my country.

Last, I would like to thank to Turkish government and the members of Turkish Educational Attaché for the financial support. They let me get a higher education in the US and allow me to serve my country.

ABSTRACT

Developing new materials have historically been time-consuming. Computational material discovery can search large design space to identify promising candidates for experimental verification. Recently, Density Functional Theory (DFT) based first principle calculation has been able to calculate many electrical and physical properties of materials, making them suitable for computational doping based material discovery. In material doping, given a base material, one can change its properties by substituting some elements with new ones or adding additional elements. In computational doping, we have a grid of atoms in a supercell, some of which can be substituted with dopant atoms. There are many possible doping positions for the doped elements in the supercell, among which the most stable supercell with the lowest free electronic energy is the one that most likely appears in experiments. So finding the most stable doped supercell configuration is the first step for computational doping, which is usually done exhaustively nowadays. For each such substitution, the Vienna Ab-Initio Simulation Package is usually used to calculate its energy and higher level physicochemical properties. Free energy calculations take about 15-30 hours for a supercell of 75 atoms for substituting two positions out of 15 with a single dopant element, and it may take days to weeks for multiple dopant elements. This is a typical optimization problem with expensive evaluation functions. Here we first developed a genetic algorithm for finding the most stable structure of the doped material with the lowest free electronic energy for a single dopant element. It can reduce the running time for computational doping by up to 75%. We used SrTiO_3

perovskite as the base material and Nb as the substitution element. We also developed another genetic algorithm for multiple dopant elements. Since the search space becomes larger, the genetic algorithm works better and saves up to 85% of calculations for finding the most stable structures. Finally, we developed a genetic programming (GP) algorithm for computational doping which can simultaneously determine multiple dopant elements with different doping ratios. The simultaneous search of dopant elements and their ratios can speed up the search process for large doping spaces.

TABLE OF CONTENTS

| | |
|--|------|
| DEDICATION | iii |
| ACKNOWLEDGEMENTS | iv |
| ABSTRACT..... | v |
| LIST OF TABLES | ix |
| LIST OF FIGURES | x |
| LIST OF ABBREVIATIONS..... | xiii |
| CHAPTER 1 INTRODUCTION | 1 |
| 1.1 COMPUTATIONAL MATERIAL DISCOVERY AND DESIGN | 3 |
| 1.2 SOLID OXIDE FUEL CELLS | 4 |
| 1.3 LIMITATIONS OF CURRENT STUDIES | 8 |
| 1.4 MOTIVATIONS | 13 |
| CHAPTER 2 BACKGROUND | 15 |
| 2.1 DENSITY FUNCTIONAL THEORY | 15 |
| 2.2 GENETIC ALGORITHMS | 17 |
| 2.3 GENETIC PROGRAMMING | 24 |
| CHAPTER 3 GENETIC ALGORITHMS FOR A SINGLE DOPANT ELEMENT | 28 |
| 3.1 GENETIC ALGORITHMS FOR A SINGLE DOPANT ELEMENT | 31 |
| 3.2 RESULTS..... | 40 |
| 3.3 CONCLUSION | 46 |
| CHAPTER 4 GENETIC ALGORITHMS FOR MULTIPLE DOPANT ELEMENTS ... | 47 |
| 4.1 CROSSOVER | 49 |
| 4.2 MUTATION..... | 52 |

| | | |
|--|--|----|
| 4.3 | MATERIAL PREPERATION | 52 |
| 4.4 | RESULTS..... | 53 |
| 4.5 | CONCLUSION | 57 |
| CHAPTER 5 GENETIC PROGRAMMING FOR MULTIPLE ELEMENTS DOPING | | 59 |
| 5.1 | GENERIC VERSION OF GENETIC PROGRAMMING FOR MATERIAL DOPING..... | 59 |
| 5.2 | OUR IMPLEMENTATION OF GENETIC PROGRAMMING FOR MULTIPLE DOPANT ELEMENTS WITH MULTIPLE DOPING RATIOS | 65 |
| 5.3 | CONCLUSIONS..... | 83 |
| CHAPTER 6 CONCLUSION..... | | 84 |
| REFERENCES | | 86 |

LIST OF TABLES

| | |
|---|----|
| Table 3.1: Results of 10 independent runs for algorithms given 40 fitness evaluations .. | 42 |
| Table 3.2: 10 independent runs of three algorithms to check when they find the best solution..... | 43 |
| Table 3.3: Results of 10 independent runs for algorithms given 400 fitness evaluations | 44 |
| Table 3.4: 10 independent runs of three algorithms to check when they find the best solution..... | 45 |
| Table 5.1: 105 different positions that can be chosen for substitutions for two dopant elements | 66 |
| Table 5.2: All possible ordering of 2 elements out of 5 elements | 67 |
| Table 5.3: An example of the configuration file to run Genetic Programming on Open Beagle | 77 |
| Table 5.4: Pseudo-code of our GP implementation's main function..... | 78 |

LIST OF FIGURES

| | |
|---|----|
| Figure 1.1: A Typical Solid Oxide Fuel Cell (SOFC) | 5 |
| Figure 1.2: Two most common SOFC designs: planar and tubular..... | 6 |
| Figure 2.1: A typical genetic algorithm flowchart..... | 18 |
| Figure 2.2: An example of mutation operation using logical operands | 27 |
| Figure 2.3: An example of mutation operation using logical operands..... | 27 |
| Figure 3.1: Computational doping framework with genetic algorithms and genetic programming..... | 29 |
| Figure 3.2: The grid of the Nb-doped SrTiO ₃ . The 4 Nb atoms are respectively in the 3 rd , 8 th , 10 th , 14 th positions..... | 32 |
| Figure 3.3: A representation of an individual that places the 4 Nb atoms at 3 th , 8 th , 10 th , 14 th position of 15 possible positions..... | 33 |
| Figure 3.4: Pseudo-code of our genetic algorithm..... | 33 |
| Figure 3.5: The crossover operator of GA-basic algorithm..... | 36 |
| Figure 3.6: Lattice parameter optimization for SrTiO ₃ primitive cell..... | 39 |
| Figure 3.7: The search space for 13% Nb-doped SrTiO ₃ | 41 |
| Figure 3.8: Comparison of the number of evaluations used to find the optimal solution for three algorithms | 45 |
| Figure 4.1: The grid of Al-In doped SrTiO ₃ . One Al atom substituted with one Ti at 5 th position, and one In atom substituted with one Ti atom at 11 th position | 48 |

| | |
|--|----|
| Figure 4.2: Binary representation of an individual that places Al atom 5 th position and In atom at 11 th position..... | 48 |
| Figure 4.3: One of the successful run of finding the lowest fitness value | 55 |
| Figure 4.4: Another successful run of finding the individual with minimum fitness value starting from far away from the global minimum..... | 56 |
| Figure 4.5: The result of 100 individual runs of the genetic algorithm | 57 |
| Figure 5.1: GP-based computational doping framework..... | 60 |
| Figure 5.2: Representation of the $\text{Sr}_{0.95}\text{Ti}_{0.8}\text{Nb}_{0.2}\text{O}_3$ tree structure | 61 |
| Figure 5.3: GP-Crossover operator for La/Nb-doped SrTiO_3 and Ga/Rh-doped SrTiO_3 . | 63 |
| Figure 5.4: Single dopant configuration of Nb at 4 th position (left). Two-dopant configuration of Nb and Rh at 4 th and 9 th positions (right)..... | 68 |
| Figure 5.5: Crossover of individuals Nb_4 and Rh_9. The created offspring are Nb_9 and Rh_4..... | 70 |
| Figure 5.6: Crossover between Nb_Rh_4_9 and Nb_In_5_8. The created offspring are Nb_In_4_8 and Nb_Rh_5_9 | 71 |
| Figure 5.7: Crossover between Nb_Rh_4_9 and Ga_In_5_8. The created offspring are Nb_Rh_5_9 and Ga_In_4_8 | 72 |
| Figure 5.8: Crossover between a two-dopant configuration individual and single-dopant configuration individual..... | 73 |
| Figure 5.9: Two different mutation operator: a) Mutation was applied on a subtree which is a single-dopant configuration, b) Mutation was applied on Position node (is still a subtree)..... | 74 |
| Figure 5.10: Mutation operator was applied on Nb_Rh_4_9 (a) and the new individual Nb_4 (b) was created | 74 |
| Figure 5.11: Mutation operator mutates the individual Nb_4 and creates the new individual Nb_Ga_4_7..... | 75 |
| Figure 5.12: Different tournament sizes vs best found individuals | 79 |
| Figure 5.13: Best found individuals with different number of population sizes and the number of generations..... | 81 |
| Figure 5.14: The result of 50 independent GP runs. Best found individuals are shown at each run. The top 3 best results were found at 43 runs out of 50 | 81 |

Figure 5.15: The results of 50 runs for different GP operators..... 82

LIST OF ABBREVIATIONS

| | |
|------------|-------------------------------------|
| DFT | Density Functional Theory |
| GA | Genetic Algorithm |
| GP | Genetic Programming |
| SOFC | Solid Oxide Fuel Cell |
| VASP | Vienna Ab-Initio Simulation Package |
| GULP | General Utility Lattice Program |

CHAPTER 1

INTRODUCTION

The clean, efficient, renewable and environmental-friendly energy generation is one of the most challenging works in today's world. In this energy-centric world, the importance of oil as a fuel type is known by everyone. The twentieth century was the witness of the first oil wars in the world, and it looks like these fights will continue to increase. In response, researchers and scientists have focused on substitution sources for oil to generate energy since the oil sources will run out one day. While oil-dependent countries try to create new energy generation sources to remove the high cost of oil, oil-producing countries try to be ready to compete with other countries when their resources are depleted or new energy resources become cheaper and more popular than the oil. Two decades ago, most trains used gasoline as a fuel. Today, trains are mostly run by electricity since its cost is much lower than oil.

Changing the type of the fuel requires changing the engines and equipment where the fuel is used. The type of material of the engine or the equipment using the fuel may need to be changed depending on different fuel types. Interaction between the fuel and the material, the heat when the fuel is used, the exhaust gas are some parameters that need to be considered when an energy generator system is prepared. Thus, the materials used in these systems become very important in the name of cost, security, usage-time etc.

Due to the high cost of oil and the issue of air-pollution, scientists and researchers focused on creating new, clean, environmental-friendly, and cheap power generation systems. Solar panels and wind farms have helped but, their natural limitations makes them useful only in certain conditions.

In 1962, Westinghouse Electric Corporation introduced the “solid electrolyte fuel cell”. Solid oxide fuel cells (SOFCs) are energy conversion devices that convert chemical energy directly into electrical energy. SOFCs are fuel flexible devices and are able to use different gases as fuel including: hydrogen, biogas, natural gas, methane, butane and others. They minimize the emissions, making them environmental-friendly. Another advantage of SOFCs over other power generation systems is that they work quietly and with no vibration. Also they are highly efficient, with pressurized SOFCs reaching 70% efficiency in hybrid/gas turbine power generation systems [1].

The general purpose of this dissertation is to build a framework for new material design and discovery by using genetic algorithms and genetic programming. For this, Solid Oxide Fuel Cells were chosen to be the test bed. Different materials can be used for the components of SOFC. The first topic of this dissertation will be, what *computational material design and discovery* is, and what has been done in this field, in Section 1.1. In Section 1.2, what are SOFCs and what kind of materials were used for the anode, cathode and electrolyte for SOFCs will be explained. The candidate materials proposed by scientists for SOFCs will also be discussed. Then, the kind of methods that were used to discover new materials for SOFCs will be reviewed. The limitations of current studies follows in Section 1.3. Finally, our proposed method will be explained in addition to how

it will help both finding new materials for SOFCs and also other material design/discovery researches in Section 1.4.

1.1 COMPUTATIONAL MATERIAL DISCOVERY AND DESIGN

Since we are living in a competitive world, companies rival with each other to produce a product with a lower cost, better performance, and higher profit. If we take cell phones as an example, costumers are looking for a lighter, thinner, larger screen that also has a longer lasting battery.

In such a competitive world, one needs to make “the better” survive. This may be the thinnest, lightest, the most durable, and strongest, etc. based on the costumers’ demand. While the companies produce the new technology, they need to check multiple things at the same time. For instance, cell phones can be produced more thinly, but they should also cost the same or even lower than the previous version. Similarly, the new cell phones can be produced with a larger screen but the battery cannot die in 2 hours.

Almost all technology in the world consists of a set of materials. In principle, the discovery of novel materials could be accelerated by computational studies. Computational material design minimizes costly and time-consuming experimentation. Since the experimental studies are time-consuming and expensive, computational studies have become an important area for material discovery and design. Based on the calculated properties, the values are provided from the careful synthesis and experimental tests.

In the industry, new materials do not come about easily or quickly. Commercial applications of new materials usually come many years after the initial discovery. The reason for this latency is that many parameters should be optimized to apply a new

material to the product. Some of these parameters are cost, stability, chemical and mechanical properties, durability, and potential for large-scale fabrication.

Sometimes, scientists only look for a small change for the material, attempting to make only one parameter better. In these cases, materials with similar characteristics are determined and tested for that specific parameter. The desired material, for instance, should be as strong as the current one, but needs to be lighter. Using this strategy, a new device, Infrisorb-12 [2], produced in 2011, is used for the screening of porosity of materials. This device identifies high surface area materials out of 12 samples at a time. The process can be called “high-throughput screening” in experimental studies.

For instance, Greeley et al. studied over 700 binary alloys to identify a new electro-catalyst for the hydrogen evolution reaction. They used the Density Functional Theory (DFT) calculations and found that Bismuth and Platinum alloy (BiPt) is the most promising candidate for the Hydrogen Evolution Reaction (HER). It is also verified by experimental tests that BiPt is superior to the Pt, which is a typical (HER) catalyst [3].

Curtarolo et al. studied the stability of 176 crystal structures in 80 binary alloys using more than 14000 DFT calculations [4]. Ortiz et al. combined data mining techniques and DFT to identify new materials for radiation detectors by screening 22000 compounds [5].

1.2 SOLID OXIDE FUEL CELLS

Solid oxide fuel cells (SOFCs) are electrochemical devices that convert chemical energy to electrical energy by combining a fuel and an oxidant. A dense electrolyte is placed between two electrodes, the anode (fuel electrode) and the cathode (air electrode). On the anode side, fuel is fed and passed through the oxidation reaction. At the same

time, the oxidant is fed to the cathode side and, takes the electrons from the electric circuit, and then undergoes a reduction reaction. This electron flow from the anode to the cathode produces the current electricity [6]. The component, called an interconnect, connects the anode of one cell with the cathode of another so that voltage output can be enhanced for practical applications [7]. Interconnect can be a ceramic or metallic layer placed between each individual cell (Figure 1.1).

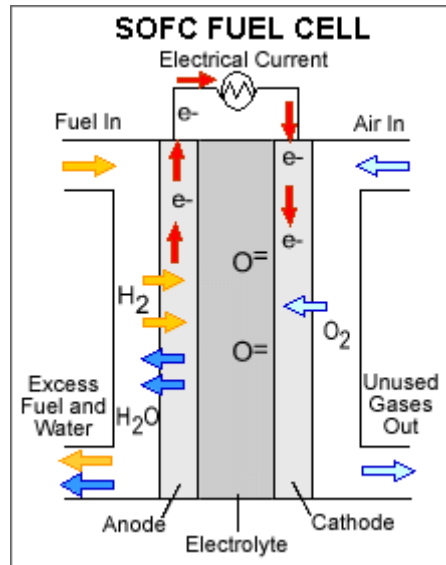


Figure 1.1: A Typical Solid Oxide Fuel Cell (SOFC)

SOFCs are reliable power generating systems in terms of efficiency, fuel flexibility, reliability and environmental friendliness. Since SOFCs are a solid-state construction, they are allowed to choose different cell and stack designs. Different design of solid oxide fuel cells have been investigated so far, such as tubular, planar, bell-and-spigot, banded, and corrugated geometries [8], [9] (Figure 1.2).

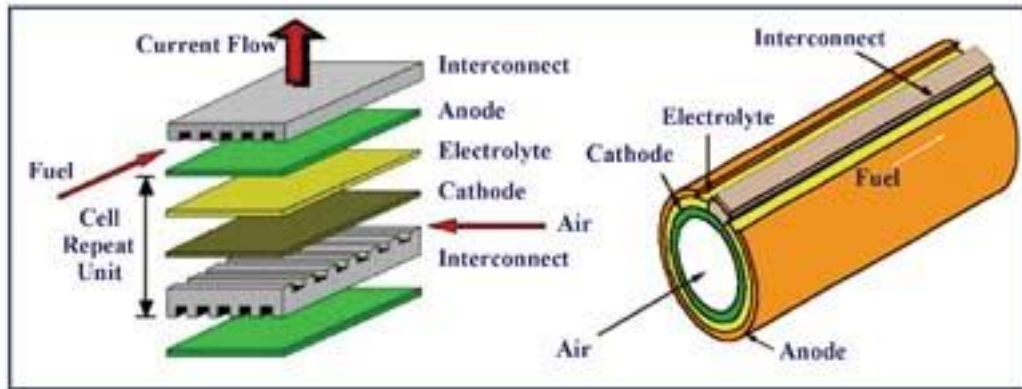


Figure 1.2: Two most common SOFC designs: planar and tubular

Additionally, SFOCs can be used as co-generators with gas turbine power systems to enable full exploitation of electricity and heat. In this system, the efficiency can be increased up to 70% [7]. Comparing to the other fuel cell technologies, SOFCs are fuel flexible and can use H₂, CH₄, coal gas, or other hydrocarbon fuels including biomass. They have been used in many areas including distributed power generation, electric vehicles, portable power for military and consumer electronics such as smart phones. In recent years, studies on SOFC have significantly expanded due to its broad application area [1], [10], [11].

Before the SOFCs can get a significant share of the electrical power market, some important issues, such as choice in fuel and the development of optimal materials for the fuel-cell stack, have to be addressed [12]. Although many materials have been used for the SOFC, some of them are more preferable due to their high ionic conductivity and chemical stability. The most common are oxide ion conducting Yttria-Stabilized Zirconia (YSZ) as electrolyte, strontium-doped lanthanum manganite (LSM) as cathode, and Nickel/YSZ as anode [6].

One of the most important features of SOFC is its high temperature operation ($800-1000^{\circ}\text{C}$). This feature offers many advantages over conventional power-generating systems. The high temperature operation feature allows the natural gas fuel to be reformed within the cell stack, eliminating the use of an expensive external reformer [13]. The high operation temperature of SOFCs produces a high quality heat byproduct, and this can be used for co-generation [14]. Additionally, SOFCs emit almost 65% less carbon dioxide than a conventional coal burning power plant.

Recently, scientists are working on reducing the operation temperature using different kind of materials. Since high temperature operation SOFC's need time to start up and cool down, different materials aid to make this process faster. Also, a lower temperature increases system stability and durability. Moreover, inexpensive metallic interconnections could be used at lower temperatures in place of lanthanum chromite-based ceramic interconnections, which require expensive fabrication costs. However, at lower temperatures, such as 600°C , the electrode kinetics and electrolyte conductivity decrease dramatically. For example, lowering the temperature increases the YSZ electrolyte resistance [15].

SOFC materials usually demand multiple and conflicting requirements [16]. For electrolytes, high ionic conductivity, low electronic conductivity, stability in oxidizing and reducing environment, long-term stability, and low reaction with electrodes are expected. For electrodes, high catalytic activity, high ionic conductivity, and electronic conductivity are expected. While some of these properties can be approximately calculated from DFT, many of them can only be measured reliably from experiments. In doping literature, a common result is a tabular presentation of the material properties of

the doped materials with different doping compositions or temperatures. It is thus useful to develop predictive models to predict such property based on atomic, electronic, and structural features, which may significantly reduce the number of experimental tests as well as search space for genetic programming-based high-throughput screening.

Macro-scale material properties, such as electronic conductivity, are influenced by many material features including crystal structure, local atomic environments, electronic structure, bonding and energy levels of a dopant-host system. Kong et al. developed a predictive model of light yield using Partial Least Squares (PLS) regression [17]. However, the relationship between these low-level features and high-level properties is usually non-linear. One approach can be to develop a predictive model of material properties using **non-linear symbolic regression** [18] based on genetic programming. This approach has been successfully applied to discover scientific laws from experimental data [19].

Due to these conflicting requirements of fuel cell materials, a great number of doping experiments have been conducted to material design to tune the material performance for both anode and cathode sides [20]–[41].

Determining the potential doping elements experimentally is time consuming and expensive work. Despite years of experimental efforts, the ideal set of SOFC materials remain to be discovered. Thus, theoretical data and computational approaches are needed for guiding experimental doping to help the SOFC material discovery process.

1.3 LIMITATIONS OF CURRENT STUDIES

Many experimental studies have been conducted to search for ideal materials for SOFC operation in the past decades [7], [16]. Today, scientists are working on lowering

the operation temperature to 500-600°C, which can significantly reduce the cost and improve the reliability of the SOFC system. However, it is a huge challenge to the development of electrolyte and electrode materials because many requirements have to be met at the same time. The materials used to make cell components for different designs of SOFC can either be the same or very similar. It is important to choose the proper materials for different cell components [13]. Thus, current studies have focused on either some known systems or materials that have already been explored by experimental doping.

In general, a good electrode in solid oxide fuel cells has to meet such requirements; high electronic conductivity, chemical and dimensional stability, and compatibility and minimum reactivity with the electrolyte [13]. Here, the thickness for such electrode-supported designs can be as low as 5-20 µm which decreases ohmic resistance and makes them suitable for operation at lower temperatures (~800°C).

Similarly, the requirements for electrolytes are high ionic conductivity, low electric conductivity, chemical stability, and good mechanical properties for long-term stability. Mainly, three electrolyte systems have been used in recent studies; Yttria stabilized zirconia (YSZ), strontium-doped lanthanum manganite (LSM), and gadolinium or samarium-doped ceria (CGO/CSO) [16]. The thickness of the electrolyte, typically YSZ, is 50-150 µm, which keeps its ohmic resistance high. These cells are suitable only for operation at ~1000 °C [10]. Many dopant electrolyte materials have been tested for these systems, including $\text{Ce}_{0.85}\text{Gd}_{0.85}\text{Mg}_{0.05}\text{O}_{1.9}$, $\text{Ce}_{1-a}\text{Gd}_{a-y}\text{Sm}_y\text{O}_{2-0.5a}$, $\text{Ce}_{1-x-y}\text{Gd}_x\text{PrO}_{2-z}$, $\text{Ce}_{0.8}\text{Sm}_{0.2-x}\text{Y}_x\text{O}_{1.9}$, $\text{Ce}_{0.8}\text{Gd}_{0.2-x}\text{Y}_x$, etc [42]. A lot of performance comparisons have been

reported for these materials in the literature [43]. However, it is difficult to obtain reliable design rules to guide the search process.

Cathode (electrode) material systems are usually in the form of ABO_3 perovskite oxides, K_2NiF_4 structures, or ordered double perovskites. The main approach of identifying new cathode materials is also doping materials such as $Ba_{0.9}Co_{0.7}Fe_{0.2}Nb_{0.1}O_{3-\delta}$, $Ba_{0.5}Sr_{0.5}Co_{0.8}Fe_{0.2}O_{3-x}$, $La_{2-x}Sr_xNiO_{4+x}$, $LaBaCuFeO_{5+x}$, $LaBaCuCoO_{5+x}$, $La_{0.6}Sr_{0.4}Co_{0.2}Fe_{0.8}O_{3-\delta}$ [22], [41], [44]–[48]. However, the number of doping possibilities is extremely large, and an exhaustive search is not feasible. Thus, some computational screening is necessary to guide the experimental studies. The anode material systems include mainly YSZ and Ni/YSZ Cermets and perovskite such as Titanates and Chromites. An anode material is preferred to be a stable microstructure with high ionic and electronic conductivity. Doping strategies have been applied to increase the ionic conductivity of $SrTiO_3$, or electronic conductivity by Niobium [23], [24], [35], [49]–[51].

In the last few years, Chen et al. have explored many different SOFC materials by experimental doping, and they are: electrolyte materials $La_{0.8}Sr_{0.2}Ga_{0.87}Mg_{0.13}O_3$, $BaCe_{0.7}In_{0.2}Yb_{0.1}O_{3-\delta}$ [52], anode materials $Sr_2Fe_{1.5}Mo_{0.5}O_{6-\delta}$ [27], $Sm_{0.2}(Ce_{1-x}Ti_x)_{0.8}O_{1.9}$ [23], $Sr_2Fe_{4/3}Mo_{2/3}O_6$ [25], $Sr_{0.9}Ti_{0.8-x}Ga_xNb_{0.2}O_3$ [20], cathode materials $Ba_{0.9}Co_{0.5}Fe_{0.4}Nb_{0.1}O_{3-\delta}$ [34], $BaCo_{0.7}Fe_{0.2}Nb_{0.1}O_{3-\delta}$ [22], $La_{0.6}Sr_{0.4}Co_{0.2}Fe_{0.8}O_{3-\delta}$ [41]. However, a more extensive exploration of the doping space is needed to find more suitable SOFC materials. Since the SOFC material search is a multi-objective optimization problem due to multiple property requirements, it is useful to explore the design space using first principle density functional theory (DFT) calculations.

First principle density functional theory (DFT) calculations have been applied to doped materials since 1978 [53]–[69], and the number of such studies is increasing. The studies about the first principle calculations have been reviewed by Hautier et al. [70]. The most common approach is to build a supercell from unit cells and then replace the elements at some positions with the possible dopant atoms. The exchange and correlation potential is usually treated with the Generalized Gradient Approximation (GGA) by Perdew and Wang (PW91), and the interaction between ions and electrons is described by the projector-augmented wave (PAW) method. For example, Han et al. [Hu 103] determined the magnetic properties of Li, Na and K doped AlN using VASP (Vienna Ab Initio Simulation Package) based first-principle calculations. They used a supercell built from 3x3x2 Wurtzite unit cells, containing 72 atoms in total. By calculating the density of states (DOS) and spin density distribution, it was shown that the origin of ferromagnetic coupling can be attributed to a p-p hybridization interaction involving holes. Chen et al. [71] studied the electronic band structure of SrTiO_3 with different Nb-doped concentrations by using first principle calculations. Zhang et al. [61] studied the electronic structure and optical properties in heavy metal doped ZnO using DFT-based structural and band structure calculations. The supercells with the 32, 64, 72 and 108 atoms were used in these calculations. They found that Ag- and Au-doped ZnO have little lattice mismatch, while Pt-doped ZnO has a large lattice mismatch. Additionally, the structural, electronic and magnetic properties of Ca-doped $\alpha\text{-Cr}_2\text{O}_3$ (Chromium oxide) crystal have been investigated using the DFT calculation [72]. It was found that the electronic band structure and the band-gap width are in close agreement with the experimental results. Oxygen vacancy formation and migration in N-doped Cu_2O [73]

and Sr- and Mg-doped LaGaO_3 [74] are investigated using DFT principles. They calculated the energy based on different modifications of the band structures and oxygen vacancy formations.

First principle studies have also been applied to fuel cell material research [53], [75]–[80]. Zhou et.al [81] studied oxygen reduction reaction (ORR) on a cathode material LaSrCoO_4 by using the periodic density functional theory (DFT + U). They investigated that the Co site is a more preferred site for the absorption of oxygen. Shishkin [75] et al. also performed a DFT + U study of the electronic structure and chemical properties of the Ni/CeO_2 and $\text{Ni/CeO}_2/\text{YSZ}$ systems and compared the change in the electronic charge localization. Chen et al. [82] studied spin-polarized DFT calculations to investigate ORR on Sr-doped LaMnO_3 cathode material. Ahmad et al. [83] used the CRYSTAL09 software package and studied the thermodynamic phase stability of LaMnO_3 and its competing oxides. An et al. [84] applied DFT to determine catalytic activity of bimetallic nickel alloys for solid-oxide fuel cell anode reactions.

Most of the current DFT studies on fuel cell materials have been used for the explanation of experimental results rather than high-throughput screening to find new materials. Currently, a lot of information can be extracted by using first-principle calculation based DFT, including the type of the band gap, the width of valence and conduction bands, the effective mass of electron and hole, charge densities, total and partial density of states. These features of materials can be used to predict high-level performance such as ion and electronic conductivity. In this study, DFT studies were applied on SrTiO_3 perovskite for SOFC anode material. This dissertation will help to

develop a computational doping framework and pipeline for large-scale screening of new fuel cell materials.

1.4 MOTIVATIONS

This dissertation is motivated by the recent success of combinational data mining algorithms and material informatics [17], [85]–[90], and DFT based high-throughput screening [91], for material discovery. Recently, research groups have studied the screening of new materials for lithium ion battery materials [92], [93], alloys [94], photocatalysts [95], and nanowires [96]. Ramprasad et al. brought the review papers of different application areas of first principles computation studies together and presented an excellent source for researchers in 2012 [97].

Curtarolo et al. proposed an automatic optimization framework AFLOW [98] which has automatically calculated the band structures of 150,000 structures from the Inorganic Crystal Structure Database (ICSD) using VASP. Also, a repository for materials was built, AFLOW.LIB.ORG [99] that has phase diagrams, free energies, stable and meta-stable structures of alloys, electronic structures and magnetic properties. AFLOW framework has useful utility scripts that can further extract various types of electronic properties from the band structures, such as band gap, charge density, etc. They claimed that their hybrid GGA/PBE(+U) calculation takes much less time than the default VASP calculations that usually take about 5 hours. The details of high throughput electronic band structure calculations and several other challenges of DFT was explained in [100].

Another high-throughput infrastructure for density functional theory calculations has been proposed by Jain et al. [91]. They implement crystal structure prediction coded-

in Java for data selection and then use Java back-end to create batches of DFT jobs. These jobs are wrapped by AFLOW, which optimizes each structure. The jobs are submitted to the Sun Grid Engine queue system, and the results are entered into a PostgreSQL database via Java Database Connectivity (JDBC). This functionality is accompanied by a graphical front-end, allowing for data exploration and analysis.

These projects have established a good model of the use of DFT and will guide this study to develop our computational doping framework. Starting from one candidate element for one material, we will be able to test multiple candidate elements for that material at the same time. Using the power of genetic algorithms and genetic programming we will be able to make “high-throughput screening” for new material design and discovery studies.

CHAPTER 2

BACKGROUND

2.1 DENSITY FUNCTIONAL THEORY

Density functional theory (DFT) is one of the most successful approaches to investigating the electronic structure of atoms, molecules, solids; in general, many-body systems. The ground state properties of a system can be calculated by using some functions of electron density with this theory. Today, DFT can be applied in many research areas such as physics, chemistry and material science. For instance, in computational chemistry, one can easily predict molecular structures, atom and ion energies, and electric and magnetic properties. In most cases, DFT calculations are quite satisfactory when compared to traditional methods.

DFT tries to obtain approximate solutions to the Schrodinger equation (1) at a reasonable computational time while keeping the balance between the accuracy and the scalability. This equation was formulated by Erwin Schrodinger in 1925, and it indicates how the quantum state of a physical system changes with time.

$$i\hbar \frac{\partial}{\partial t} \psi = \hat{H}\psi \quad (1)$$

Equation 1 is the general form of the time-dependent Schrodinger equation; i is the imaginary unit, t , is time, \hbar is the Planck's constant divided by 2π , ψ is the wave function of the system, and \hat{H} is the Hamiltonian operator (total energy of any given wave function).

The non-relativistic time-dependent Schrodinger equation (2) calculates the movement of a single particle in an electric field. Here, m is the particle's mass, V is its potential energy, ∇^2 is the Laplacian, and ψ is the wave function.

$$i\hbar \frac{\partial}{\partial t} \psi(r, t) = \left[\frac{-\hbar^2}{2m} \nabla^2 + V(r, t) \right] \psi(r, t) \quad (2)$$

Soon after the description of the Schrodinger, Hartree and Fock created a theory for determining the wave function and energy of a quantum many-body system in a stationary state. The Hartree-Fock method simplifies the Schrodinger equation by calculating the one-electron wave functions that are approximated by a linear combination of atomic orbitals (wikipedia).

Hohenberg and Kohn established the fundamentals of today's DFT by stating their theorems. First the ground state properties of a many-electron system can be uniquely determined by its electron density. With this theorem, the many-body problem of N electrons with $3N$ coordinates can be reduced to 3 spatial coordinates by using the some functional of the electron density. The second theorem states that the exact ground state charge density minimizes the total energy [101].

Finally, Kohn and Sham moved the idea one step forward and turned the DFT into a more practical tool by expressing the total energy as a set of equations for non-interacting electrons. Finding a fictitious system of non-interacting electrons will lead to finding the same density as the one with the interacting electrons.

$$\left[-\frac{\hbar^2}{2m} \nabla^2 + v_{KS}(r) \right] \psi_i(r) = \varepsilon_i \psi_i(r) \quad (3)$$

In the equation (3), Kohn and Shan defined v_{KS} as the external potential in which non-interacting electrons move. Wave functions are represented by $\psi_i(r)$, and ε_i is the orbital energy of the corresponding Kohn-Shan orbital. $v_{KS}(r)$ can be expressed as the

sum of three terms. $V(r)$ represents the Coulomb interaction between the nuclei and electron, $V_H(r)$ is the Hartree potential, and $V_{xc}(r)$ is the exchange-correlation potential. Here, $V_{xc}(r)$ is the only unknown term.

So far, a systematic way to determine this exchange-correlation potential, V_{xc} , has not been found. Most DFT researchers use two major approximations to define this function. The first is Local Density Approximation (LDA), which accepts the exchange-correlation potential at each location is equivalent to that of a homogeneous electron gas with the same electron density. The second is the Generalized Gradient (GGA), which uses the gradient of the electron density.

2.2 GENETIC ALGORITHMS

Genetic algorithms were first developed and used by John Holland in the 1970s [102]. As a subclass of evolutionary algorithms, genetic algorithms solve the optimization problems using the idea of natural evolution. Individuals are born and, try to survive in a competitive environment. The winners proceed with their lives and pass their genetics onto the next generation, while the others die. This procedure is usually referred to as the “survival of the fittest”. Today, genetic algorithms are used for search and optimization problems. Many different disciplines such as bioinformatics, computational science, mathematics, physics, manufacturing, use genetic algorithms to find global optimum solutions.

As shown in Figure 2.1, the procedure starts with the creation a population with the individuals in it. The population is created randomly in most cases, but if the programmer has some prior knowledge of the solution, the system may be started with some known individuals. It may help to converge quickly because the system has already

started at a good position. The individuals, also called chromosomes, are actually “candidate solutions” for the optimum solution. The representation of individual may vary based on the problem, however the most commonly used representations are binary strings and real values. The individuals consist of “genes”, which represent a one-bit string in a binary string representation, or a real value in real value representation. Each individual has one fitness value that is calculated by the fitness function, describes its score (or rank) compared to other individuals in the population. Based on this fitness value, the individuals are ranked in the population in accordance to which ones are adequate and which ones are not. The adequate individuals pass on to the next generation while the remaining are removed from the population.

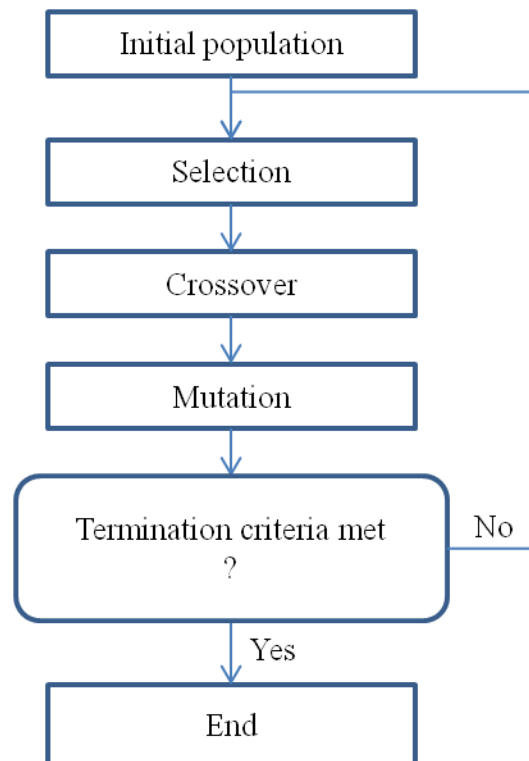


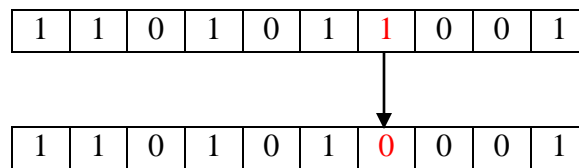
Figure 2.1: A typical genetic algorithm flowchart

Although programmers have described some new operators for genetic algorithms recently, the main operators are mutation, crossover and selection. The new operators are actually some modified or combined versions of these operators.

A mutation operator takes an individual and modifies one or more genes of an individual. It maintains the genetic diversity of the population from one generation to the next. A user-defined mutation probability is chosen to decide what percentage of the population will be mutated. If the mutation operator creates an individual that is superior to the current one, the new individual is put into population for the next generation. Thus, mutation in genetic algorithms usually helps to create better individuals. The mutation probability is usually chosen to be low, because if it is too high, the population will alter all acceptable individuals, not allowing the population to create better solutions.

Different types of mutations can be used in genetic algorithms; several are detailed below:

- Flip Bit Mutation: The mutation operator takes an individual, chooses a gene (randomly) and changes it (0 replaced with 1, and 1 replaced with 0). This operator is only used for binary genes.



- Boundary mutation: This type of mutation operator is usually used for real valued individuals. The mutation operator selects a gene and replaces it with either a lower or upper bound of search space. For example, if the search space is $[-500, 500]$,

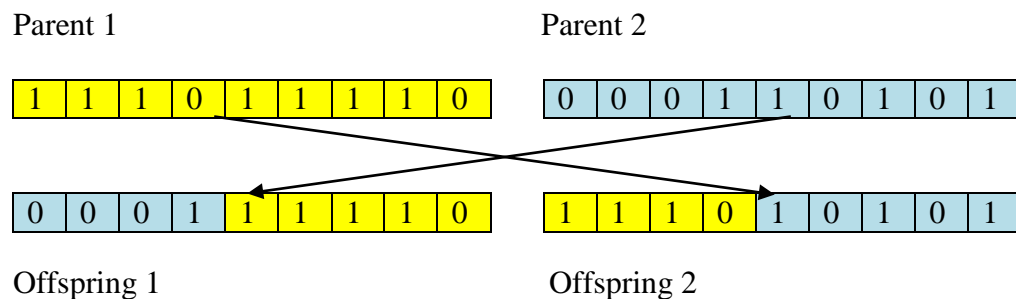
| | | | | | | | | | |
|----|----|----|-----|----|-------|----|-----|-----|------|
| 34 | 72 | 56 | -20 | 10 | -12.5 | 22 | -82 | 402 | -311 |
| | | | | | ↓ | | | | |
| 34 | 72 | 56 | -20 | 10 | 500 | 22 | -82 | 402 | -311 |

- Non-uniform Mutation: This operator increases the probability that the amount of mutation will be close to 0 with the next generations. Hence, this mutation operator keeps the population from stagnating in the early generations and allows tuning solutions in later generations.
- Uniform Mutation: The operator replaces the value of the selected gene with the uniform value chosen between the lower and the upper bounds for this gene.
- Gaussian Mutation: This mutation operator adds a unit Gaussian distributed value to the selected gene. If the new value of a gene falls outside the boundaries, it is clipped.

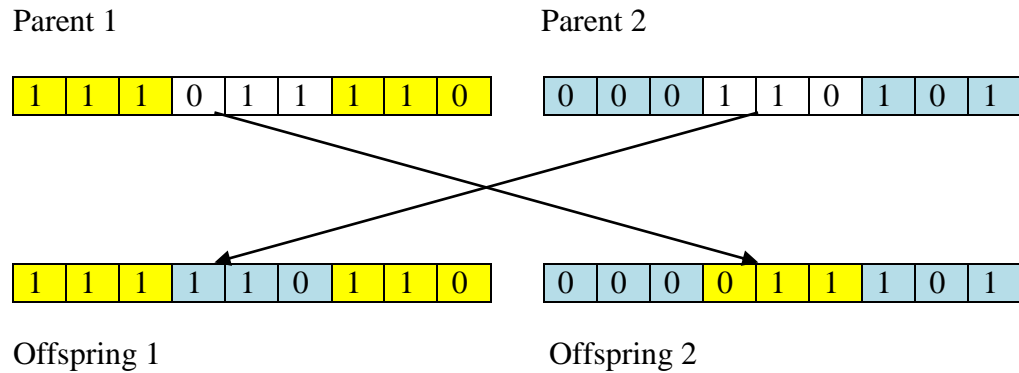
Crossover is another important operator for genetic algorithms. The idea is to get good genes from each parent and create a better offspring. Crossover also occurs according to user-defined probability in the evolution process.

Different types of crossover can be chosen based on the user's preference:

- One-point Crossover: This operator randomly chooses a crossover point, and exchanges the parts from each parent to create two offspring.



- Two-point Crossover: This operator selects two crossover points and exchanges the parts between these two points from each parent.



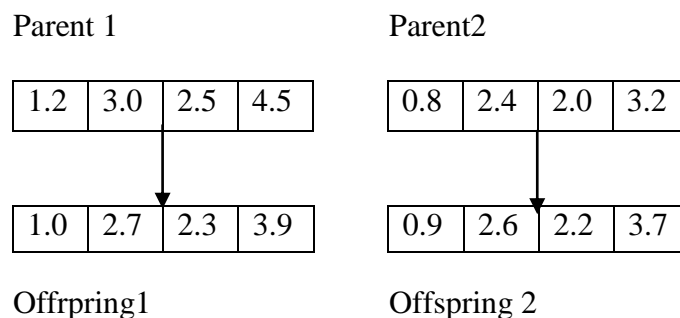
One disadvantage of one-point crossover and two-point crossover is destruction of the building blocks. In other words, if 5 consecutive genes have good characteristics, and if these genes are separated while choosing the points, the offspring will not be as adequate as their parents.

- Arithmetic Crossover: A weighting factor ($x \in [0,1]$) and a linear function are used to create offspring in the following way:

Offspring 1: $x * (\text{Parent1}) + (1-x) * (\text{Parent2})$

Offspring 2: $(1-x) * (\text{Parent1}) + x * (\text{Parent2})$

For example: $x = 0.6$



This arithmetic crossover is clearly for real-valued individuals, not for binary strings. For binary representation, similar operators can be defined, such as two parents can be added by using the logical operator ‘AND’, and they can create one offspring.

- **Uniform Crossover:** This operator uses a fixed mixing ratio to decide which parent will contribute each of the gene values in the offspring individual. If the mixing ratio is 0.5, then the offspring will consist of half of each parent. Note that, unlike one-point and two-point crossovers, a uniform crossover works at the gene level, not the segment level. It is decided randomly which genes are selected from each parent.

Parent 1

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|

Parent 2

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 |
|---|---|---|---|---|---|---|---|---|

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 |
|---|---|---|---|---|---|---|---|---|

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
|---|---|---|---|---|---|---|---|---|

Offspring 1

Offspring 2

Selection operator is the mechanism used to choose which individuals will be selected to be parents for the crossover operator. Selection uses fitness value to evaluate the individuals and sort them in descending order based on their fitness value. The idea is to choose the best individuals and then create better offspring. To do that, different selection methods have been tested in the literature. The most commonly used methods will be shown here.

- **Roulette-Wheel Selection:** This operator calculates the sum of the fitness values in the population (S). A random value (r) is chosen from the interval $(0-S)$. Then, selection starts a loop to sum fitness values through 0 to temporary sum (s). If s is greater than r , the operator stops and returns the

individual to its origin. For this operator, the individuals with greater fitness values will be more likely to be chosen. This may cause a problem if the difference among the fitness values is large. The individuals with lower fitness have very few chances to be selected. To avoid that, individuals can be ordered with a weight value, which is determined after they ordered according to their fitness. For example, if a population has N individuals, the best individual's weight value will be N , the second best $N-1$, and the worst individual 1. Thus, the operator will select the better individuals anyway; however, the individuals with lower fitness will have a chance to be selected.

- **Tournament Selection:** Although there are many different versions of this operator, the main idea is to make two randomly chosen individuals combat based on their fitness value. The winner will be the first parent from the fight. Then another combat occurs between two randomly chosen individuals, and the winner will be the second parent. Another version of this method chooses two individuals again and as well as a random value between 0 and 1. If this number is greater or equal than the user-defined tournament parameter, the selection operator chooses the first individual as parent. Then it repeats the same procedure again to select the second parent.

Elitism is another operator that can be used in genetic algorithms. The main idea is to keep the good individuals and not allow them to be modified. At each generation, before mutation and crossover, elitism copies a user-defined number of best individuals to the next population. This procedure actually increases the performance of genetic algorithm since the best found individuals are protected.

One known problem of genetic algorithm is “premature convergence”. Premature convergence is used if a population for an optimization problem converges too early and is stuck in the local optimal solution. If sub-optimal individuals dominate the population, selection operator tends to keep it around, and the population only creates individuals that are very similar to their parents. As the number of generation increases, the population loses its variety.

Two key parameters are used in genetic algorithms. One of them is population size, which indicates the number of individuals in the population. If the chosen population size is too small, there will not be enough diversity in the population. If it is too large, however, then it will take a long time to compute even for one generation. The second important parameter is number of generations. The number of generations indicates how many generations the population will breed. If the chosen number of generations is too small, the population may not be able to find the optimal solution.

The complete procedure of genetic algorithm is concluded with a termination criterion. This termination criterion is usually set to the maximum number of evaluation reached. The number of evaluation is the multiple of the number of generation and the population size. Another termination criterion is to set a threshold that checks the solution found by the genetic algorithm. If the system starts creating similar individuals and can no longer find better solutions, the search process can be stopped.

2.3 GENETIC PROGRAMMING

Genetic programming (GP) is also an evolutionary-based method inspired by biological evolution. GP consists of a set of instructions and a fitness value to determine how a computer program performs a job. While the individuals in genetic algorithms are

usually raw data, the individuals in genetic programming are computer programs. These individuals (computer programs) are traditionally represented as tree structures. Every internal node in the tree has an operator function, and every terminal node holds an operand.

Although the first studies on genetic programming were made in the early 1950s, today's modern genetic programming was developed by John R. Koza in the 1990s [103].

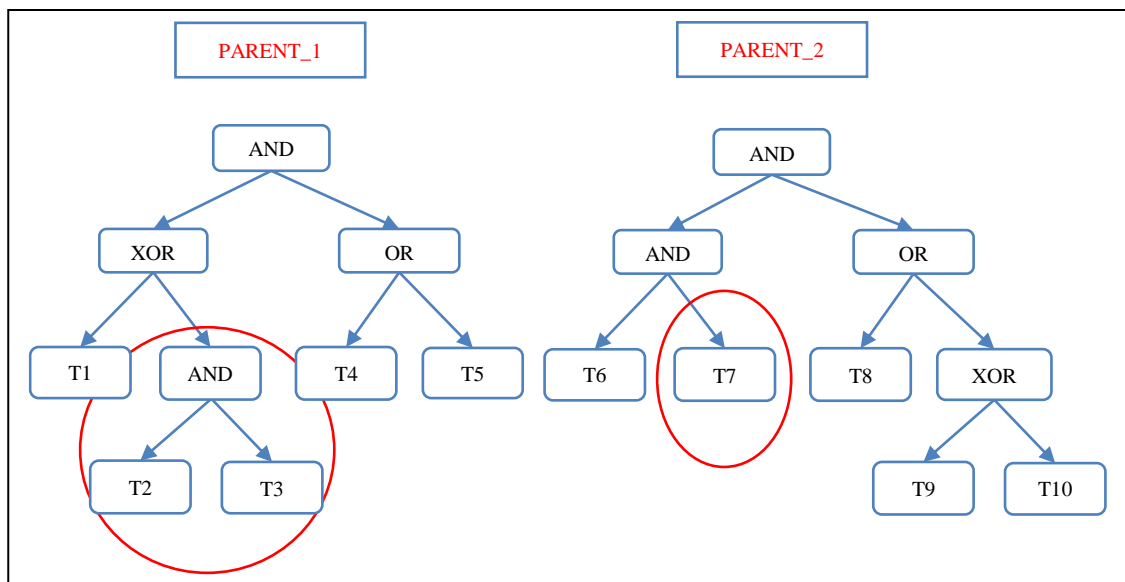
GP is a probabilistic algorithm that searches the space of combinations of predefined functions and terminals [104]. Some of the important parameters are population size, maximum generation, maximum initial program length (initial tree depth), maximum program length (tree depth), termination criteria, and mutation/crossover duplication rates. GP starts with randomly created computer programs, which are called individuals. First, the root function is selected, and then each leaf is filled until the depth of tree is reached. This procedure is repeated until all the members of population are created. When the initial population is ready, the fitness values of the individuals are calculated by a user-defined fitness function. The calculated fitness values are stored to be later utilized by the selection operator. The selection operator ranks the individuals in population with respect to their fitness values, and the better (the fitter) individuals are selected for reproducing. In other words, the pairs of computer programs (individuals) from the population, based on their observed fitness, are used to create new subprograms (new offspring).

The population of the programs evolves recursively over a series of generations. Like other Evolutionary Algorithms (EA), GP also uses the Darwinian principle of natural selection (survival of the fittest). GP also uses EA operators such as selection,

crossover (Figure 2.2) and mutation (Figure 2.3). The fitness function in GP determines whether the programs (individuals) are successful or not. In other words, fitness function ranks the individuals based on their value, the useful ones are chosen for the next generation, and the ones with the lower fitness are kicked out of the population. In most cases, fitness values are real numbers, and the optimization problem turns out to be a minimization or maximization problem.

In genetic algorithms, the individuals can be represented as binary or real values. The length of the individuals should be fixed through the generations and cannot be changed. However, in genetic programming, the individuals can be different sizes, and their length can change as the number of generation increases.

Koza et.al. [104] defines 6 steps to prepare a genetic programming: determining the architecture of the program trees, identifying the terminals, identifying the functions, choosing the fitness measure, choosing control parameters for process execution and identifying the termination criterion and output option. In chapter 3, the steps will be explained with an illustrative example.



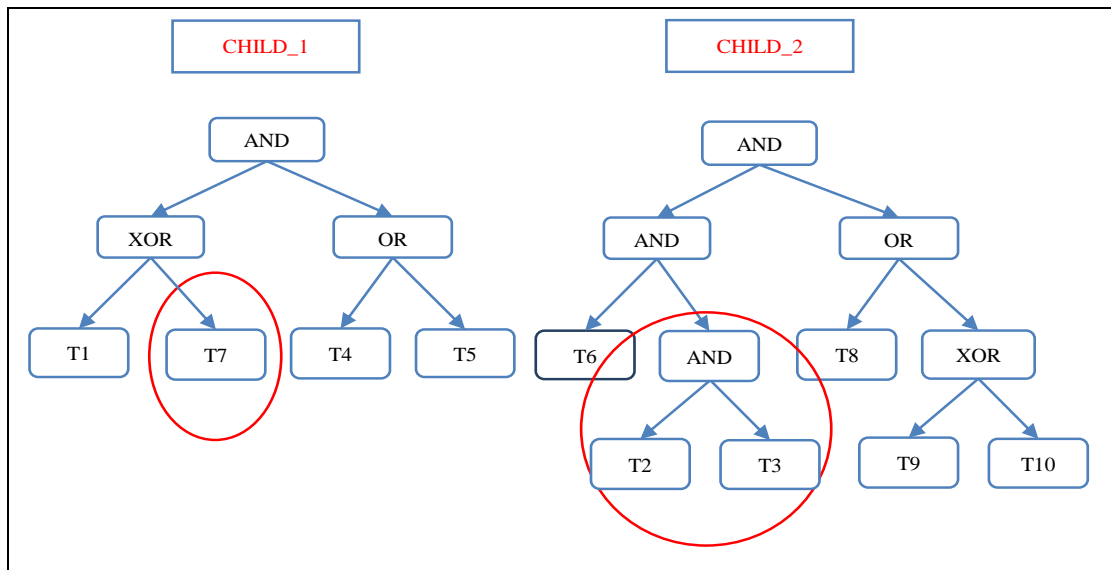


Figure 2.2: An example of mutation operation using logical operands

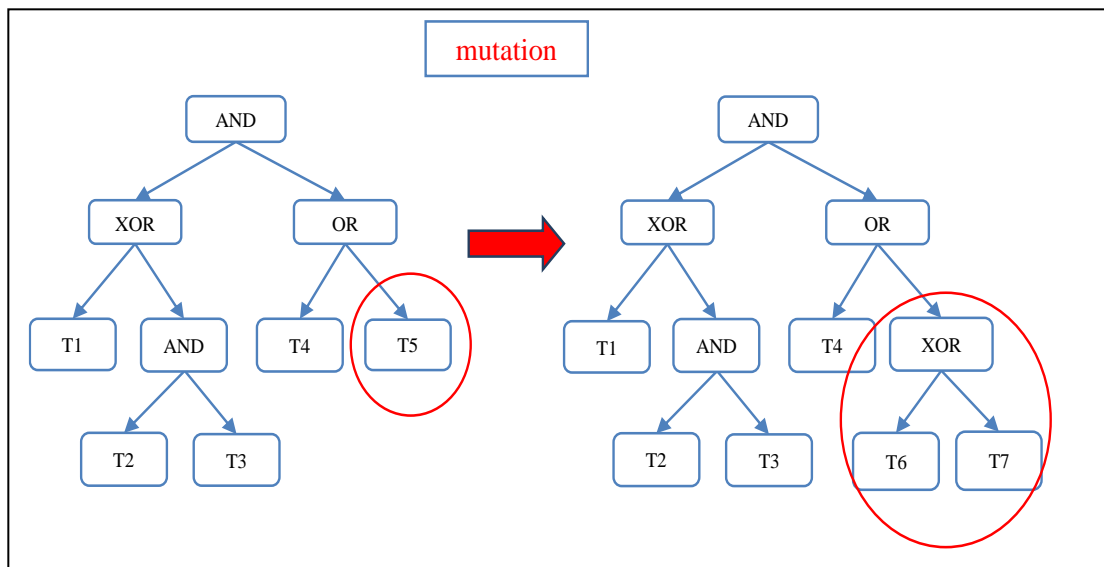


Figure 2.3: An example of mutation operation using logical operands.

CHAPTER 3

GENETIC ALGORITHMS FOR A SINGLE DOPANT ELEMENT

The material doping problem can be described as determining what species/elements and in what compositions should be added to the base material to achieve desired material properties. It is also called inverse design problem [105]. The most common approach in computational-doping based material design is to substitute specific elements in a material with new ones. This substitution is done in a small percentage to keep the balance of the original material's characteristics with the new element's effects. Too many or too few dopant elements will be unable to achieve desired results. To find the best ratios of base material and the substitution elements via computational doping, a supercell is first created from one or more primitive cells of the base material, of which several of the atoms will be replaced by the dopant elements. The larger the created supercell, the higher the simulation accuracy of the doped material. However, larger supercells make the running time of DFT calculations to be as long as days or weeks for each doped configuration. Given a base material such as SrTiO_3 , there are dozens of possible dopant elements for substituting Sr/Ti/O elements, and for each dopant elements, there are many possible positions for the substitution, which leads to a complex doping space. It is practically infeasible to exhaustively search and evaluate all possible doping configurations even using DFT calculation since each DFT calculation of a configuration may need hours or even days. To reduce the number of such expensive optimization simulations, we developed a combinatorial genetic algorithm to search the

dopant element configuration space on the supercell to find the most stable configuration with the lowest free electronic energy. The following sections will describe the representations of individuals and the methods for mutation and crossover. It will also detail the usage of the elitism, the type of selection method and the objective function for our computation doping experiments.

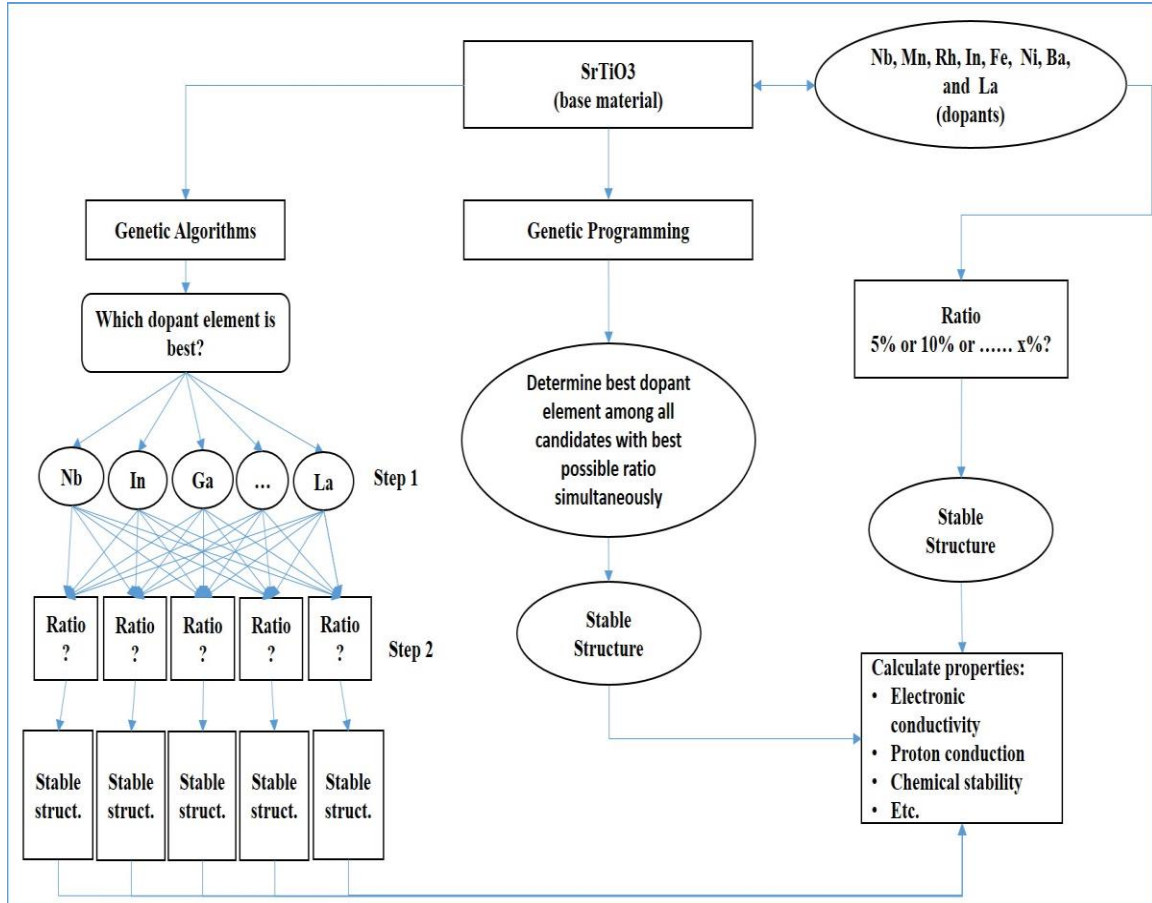


Figure 3.1: Computational doping framework with genetic algorithms and genetic programming

The computational doping procedure is summarized in Figure 3.1. We now explain this procedure by framing the following questions: 1) What is the purpose of doping elements into another material? 2) Why do we need to do computational doping? 3) Can doping not be done experimentally? The purpose of doping elements into a

material is to change the characteristics of the base material. If, for example, we have a base material and we want to increase its electronic conductivity, this can be done by introducing another element into the material or substituting some elements of the base material with the new element. Similarly, the heat resistance of a material can also be increased by doping new elements. Many electronic, chemical and even physical properties of a material can be changed by doping new elements [106]. In addition, new materials can be discovered by doping [107].

As shown in Figure 3.1, there are many possible dopant elements given a base material. To calculate the chemical or electronic properties of the doped material, such as electronic conductivity, one needs to get the most stable structure for the material. However, there are many different configurations for doping an element into the base material. To get the most stable structure, the energy of all configurations needs to be calculated exhaustively. Since DFT calculations may take hours to days depending on the material structure, supercell size and/or dopant elements, it is infeasible to calculate the energies of all the configurations exhaustively. In addition, the ratios for doping also needs to be determined to find the most stable structure. For instance, if Nb was chosen for doping SrTiO_3 consisting of 75 atoms (15 Sr, 15 Ti, and 45 O) and will be used to substitute the Ti atoms. For 1 Nb-Ti substitution, there are 15 different configurations, and DFT calculations need to be run for each of them. For 2 Nb-Ti substitution, there are 105 (15-choose-2) configurations. For 4 Nb-Ti substitution there are 1365 configurations. Each of these 1365 DFT calculations takes approximately 15-30 hours on a 12-Core 2.46Ghz CPU with 24 GB ram. Nevertheless, all different ratios and the corresponding configurations needs to compute intensive DFT calculations to find the most stable

structure for Nb-doped SrTiO_3 . It is almost impossible to prepare this many different configurations of any doped material in laboratory, and that's why we need computational doping.

In our work, by using genetic algorithms, we reduced the number of DFT calculations for finding the most stable structure of the doped material for 13% Nb-doped SrTiO_3 (2 Nb-Ti substitutions), to 27%Nb-doped SrTiO_3 (4 Nb-Ti substitutions), and saved up to 70% of time. Details about implementation of our genetic algorithms will be explained in the following section.

Even if reasonable amount of time can be saved by using genetic algorithms, determining the most stable doped structure among different dopant elements is still sequential. It is necessary to run all different configurations depending on the ratios for each individual dopant element. To address this limitation, we proposed the genetic programming based computational doping, which can work with different dopant elements with different ratios simultaneously for parallel search of the most stable doped materials. The procedure will be explained in Chapter 5.

3.1 GENETIC ALGORITHMS FOR A SINGLE DOPANT ELEMENT

Although the general idea of all genetic algorithms (GA) is similar, each GA implementation is specific to its application. The representations of individuals, the methods for mutation and crossover, the usage of elitism, the type of selection method, and finally the objective function make all genetic algorithms unique. These important components of genetic algorithm must be chosen carefully to achieve its application goals.

Since we are looking for the best configuration of atoms in the supercell, our goal is to find the best doping positions among all possible configurations. Consider an example of a supercell consisting of 75 atoms of SrTiO_3 material with Nb as the dopant element. If 4 Nb atoms are substituted with 4 Ti atoms, it makes approximately 27% Nb-doped SrTiO_3 . Since there are 15 Ti atoms in the supercell, 4 of them will need to be chosen to substitute with the Nbs. It turns the problem into a combinatorial optimization problem. Mathematically, 15-choose-4 is equal to 1365, meaning there are exactly 1365 different options for the placement of Nb atoms in the supercell for the Nb-Ti substitution (Figure 3.2). The energy of each configuration of dopants will be evaluated by running VASP [108] simulation, which is the state-of-the-art DFT calculation software. This many VASP calculations are simply impractical because each of them takes approximately 13-16 hours to complete. The goal of our genetic algorithm based approach is to reduce this number of calculations.

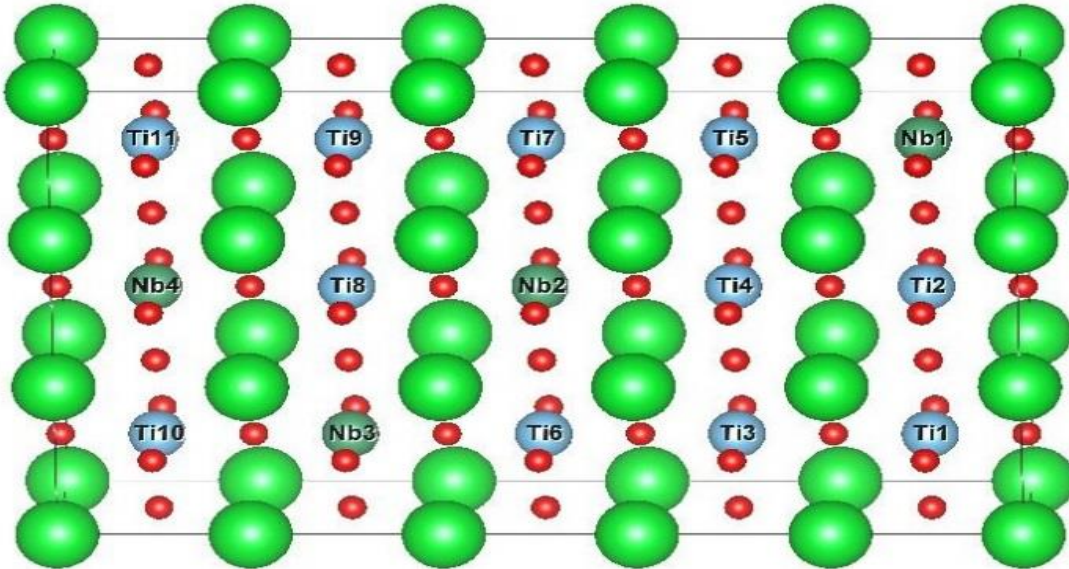


Figure 3.2: The grid of the Nb-doped SrTiO_3 . The 4 Nb atoms are respectively in the 3rd, 8th, 10th, 14th positions

The genetic algorithm used here is a generation-based genetic algorithm with fixed numbers of generations. A binary representation is used for individuals but with some constraints compared to traditional binary representation. Each individual's length is equal to the possible doping positions. The bits/positions that are occupied by doping candidate elements are set to 1, and the remaining are 0 (Figure 3.3).

| | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

Figure 3.3: A representation of an individual that places the 4 Nb atoms at 3th, 8th, 10th, 14th position of 15 possible positions

| |
|---|
| <i>Set parameters</i> |
| <i>Initial Population</i> |
| <i>Fitness Calculation</i> |
| <i>Generation Loop Start</i> |
| <i>Find elite individuals</i> |
| <i>Selection</i> |
| <i>Crossover</i> |
| <i>Mutation</i> |
| <i>Fitness Calculation</i> |
| <i>Combine elite and new population</i> |
| <i>End Loop</i> |

Figure 3.4: Pseudo-code of our genetic algorithm

To handle the issue of expensive fitness calculation, we develop a mechanism to avoid duplicate fitness evaluation (Figure 3.4). Traditional genetic algorithms can generate identical individuals that were evaluated in previous generations or even in the same generation through the crossover or mutation operator. Here we don't prefer to use the duplicate individuals in the next generations. Thus, we implemented our genetic algorithm in three different cases depending on how we avoid evaluating the same individual multiple times and what we should do when a created individual has already been evaluated before:

1) GA-basic algorithm: it checks a new individual created by crossover and mutation operators. If it has been evaluated before, it randomly creates a similar but different individual.

2) GA-SS algorithm: it checks the individual after it is created by crossover or mutation, and if the individual has been created and evaluated before, GA_SS creates another similar individual based on individuals with previously calculated fitness values.

3) GA-SC, it uses similar statistical idea with GA-SS. However, it applies this statistics during the crossover process. The details about these three different versions will be explained in forthcoming, sub-sections and the results will be explained under Results sections.

After creating the initial population, the algorithm calculates the fitness values and is ready to start generation loop. We used a fixed number of generations to keep the number of evaluations fixed. In the loop, first we find the *elite* individuals in the population based on user-defined elite percentage. These elite individuals can be used for crossover and mutation, as with any others. But eventually they will survive to the next generation. *Tournament selection* is used as the selection operator. Our selection method finds $N/2$ parents (N = number of individuals in the population – elite individuals) by using the winner of tournament idea. The selected parents are used in our uniform *crossover* operator. Since the 1's in the individuals represent the actual position of the individual dopant elements, there will always be the same number of 1's. Thus, we divide each individual to get an equal number of 1's at the same side. If we have an even number of 1's, say M , in an individual, both parts of each parent will have $M/2$ 1's. If it's odd, however, the first part of each parent will have $(M+1)/2$, and the second part will

have $(M-1)/2$ 1's. Then, we exchange the first part of the first parent with the second part of the second parent to create the first child. The second child is created with other parts. Since there should be a fixed number of 1s in the representation of an individual, uniform crossover may create a child with more or less 1s. If 4 positions are sought to place the substitution element, the crossover operator may create one child with two 1s, and another child with six 1s. This configuration is unacceptable. Another problem with uniform crossover is that if parents have some common positions. We will explain our crossover operator with an example (Figure 3.5).

After crossover, some individuals are chosen for *mutation* with user-defined mutation probability. The central idea behind the mutation is simply flipping bits. One of the 1's is chosen randomly and substituted with a randomly chosen 0. The *fitness* function is called for new individuals after crossover and mutation. Finally, these new individuals and elite individuals are added together to create a new population for the next generation. For testing purposes, our algorithm keeps the best-so-far individuals and the current population for each generation.

As mentioned above, three genetic algorithms for finding stable doped materials based on VASP DFT calculation have been implemented and evaluated.

3.1.1 GA-basic

GA-basic is a traditional generational genetic algorithm for solving the doping problem. It includes a special representation type. Since our main goal is to reduce the number of fitness calculations while still finding the optimal solutions, our genetic algorithm doesn't evaluate the individuals that have already been evaluated before.

i) If there are no common positions

Let the parents be :

| | | | |
|---|---|---|----|
| 3 | 5 | 7 | 13 |
|---|---|---|----|

 and

| | | | |
|---|---|---|---|
| 2 | 4 | 6 | 8 |
|---|---|---|---|

Then, after the uniform crossover is applied, the offspring will be:

| | | | |
|---|---|---|---|
| 3 | 5 | 6 | 8 |
|---|---|---|---|

 and

| | | | |
|---|---|---|----|
| 2 | 4 | 7 | 13 |
|---|---|---|----|

ii) If there are common positions:

- *If there is only one different position for each parent.*

In this case, if we apply uniform crossover, then the offspring will be the same with parents. Ex:

| | | | |
|---|---|---|----|
| 4 | 6 | 7 | 12 |
|---|---|---|----|

 and

| | | | |
|---|---|---|----|
| 2 | 6 | 7 | 12 |
|---|---|---|----|

| | | | |
|---|---|---|----|
| 2 | 6 | 7 | 12 |
|---|---|---|----|

 and

| | | | |
|---|---|---|----|
| 4 | 6 | 7 | 12 |
|---|---|---|----|

Instead, we search for other available positions from the *Pool* that include the common positions. Let's say 1st, 3rd, 5th, 8th, 9th, 10th, 11th, 13th, 14th, 15th positions was not used with 6th, 7th, and 12th positions together so that to build the individuals, such as [1, 6, 7, 12], [3, 6, 7, 12], or so on. Then we randomly choose one of them to create a similar offspring:

| | | | |
|---|---|---|----|
| 6 | 7 | 8 | 12 |
|---|---|---|----|

 and

| | | | |
|---|---|---|----|
| 6 | 7 | 8 | 12 |
|---|---|---|----|

- *If there is 2 or more different positions for the parents:*

The idea is to crossover the different positions of the parents and keep the common positions unchanged. Ex: Let the parents be:

| | | | |
|---|---|---|----|
| 4 | 6 | 7 | 12 |
|---|---|---|----|

 and

| | | | |
|---|---|---|----|
| 2 | 5 | 7 | 12 |
|---|---|---|----|

Then, the offspring will be :

| | | | |
|---|---|---|----|
| 4 | 5 | 7 | 12 |
|---|---|---|----|

 and

| | | | |
|---|---|---|----|
| 2 | 6 | 7 | 12 |
|---|---|---|----|

Figure 3.5: The crossover operator of GA-basic algorithm

If the crossover or mutation operators create an individual that has already been evaluated before, it needs to be replaced with a new one. In this case, GA-basic randomly chooses one single position of the individual, let's say 8th position of [3, 5, 8, 11], and then look for individuals consisting of the rest positions, 3th, 5th, and 11th. Finally, the algorithm chooses one of the available individuals from the *Pool* and creates a new individual. If

there is no individual in the *Pool* consisting of these 3th, 5th, and 11th positions, then the algorithm chooses another positions to change, let's say 5th, and look for the individuals consisting 3th, 8th, and 11th positions. If there is no individuals left in the *Pool* for 3-tuples, then the algorithm randomly chooses a pair, let's say 3th, 5th positions, and look for the individuals consisting of 8th and 11th positions in the *Pool*. This procedure is repeated for 3-tuples to change and one position to keep, if no individuals found in the *Pool*.

3.1.2 GA-SS

The second version of our implementation is GA-SS, genetic algorithm with statistical similarity. Here a statistical procedure is used to pick better parents for mutation and crossover operators to generate better candidates for fitness evaluation. In GA-basic, if an individual is created which has already been evaluated before; the algorithm will choose a random but similar positions for individuals from the *Pool*. Instead, GA-SS runs a statistical process. Instead of choosing only similar positions, GA-SS checks the previously evaluated individuals containing each position separately. For example, let the individual be [3, 8, 10, 14] which placed the Nb atoms in 3th, 8th, 10th, 14th positions, respectively. Considering we randomly choose 8th position and we will keep 3th, 10th and 14th positions occupied. Then, we look for the candidates from *Pool* which they haven't been used before. We determine the positions consisting of [3, 10, 14] and keep the fourth position as a candidate for substitution for 8th. In this instance the 1st, 2nd, 5th, 7th, and 12th positions are available and can be used. Then, we calculate the average fitness values of the individuals which have already been calculated and choose the best position which has the maximum average fitness (lowest energy). Let the 5th

position be the one which has the maximum, then we create the individual as [3, 5, 10, 14]. This procedure is repeated if there is no individual left in *Pool*, which consists of [3, 10, 14]. If so, we choose another single element from [3, 8, 10, 14] except what we already chose before, 8th. If all the possibilities is over for the single elements, the procedure is repeated for the pairs, such as [8, 10], to substitute and to keep the rest.

3.1.3 GA-SC

The last implementation of our genetic algorithm is GA-SC, a genetic algorithm with statistical crossover. The method is similar to GA-SS. However, we implement this statistical sampling process at different levels of the algorithm. While GA-SS uses statistical sampling **after** crossover or mutation, GA-SC uses statistical sampling **during** the crossover process. The statistical sub-combinatorial approach (explained in GA-SS) is applied to parents who have the common positions. GA-SS still allows running uniform crossover in part i) on the other hand, in part ii of Figure 3.5, GA-SC uses statistical sampling to choose “not-common” positions rather than choosing randomly.

3.1.4 Material Preparation

We tested our algorithms on 2 different systems of Nb-doped SrTiO₃ material: 13% Nb-doped SrTiO₃ (2 Nb atoms in 15 possible positions) and 27% Nb-doped SrTiO₃ (4 Nb atoms in 15 possible positions).

Since the material preparation is similar for both systems, only the 27% Nb-doped SrTiO₃ (4 Nb atoms in 15 possible positions) will be explained in detail. The crystal structure for the primitive cell of SrTiO₃ material was obtained from materialproject.org [109]. We initially optimized the lattice parameter of the SrTiO₃ unit cell. Figure 3.6 shows the energy values of different lattice parameters in the search process. Our optimal

lattice parameter with the lowest energy (3.946) is consistent with materialproject.org (3.945).

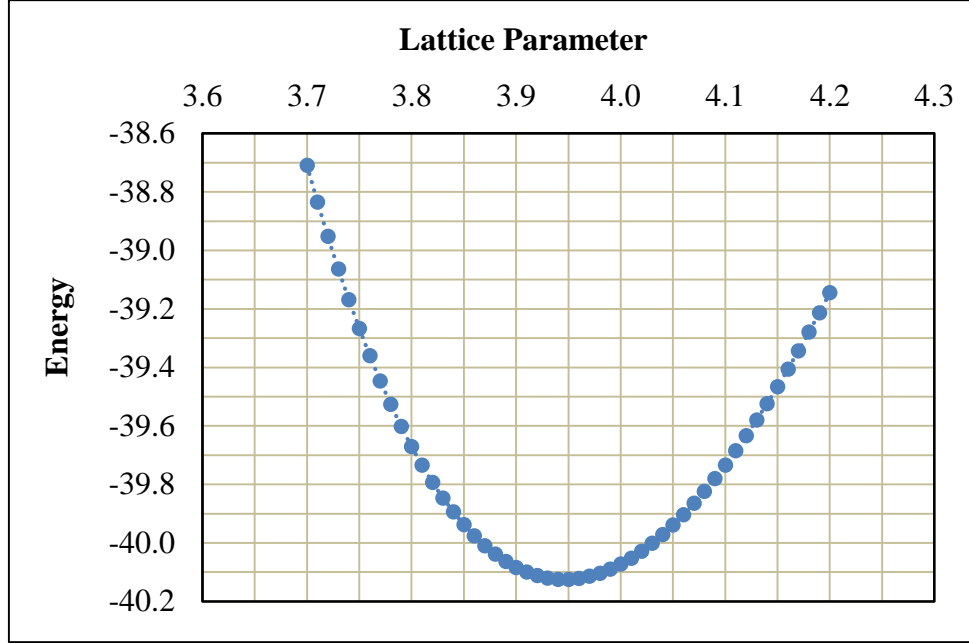


Figure 3.6: Lattice parameter optimization for SrTiO₃ primitive cell

After lattice optimization, a supercell of 75 atoms containing 15 unit cells (5x3x1) was created from fully relaxed structure. A template file is created by substituting 4 Nb atoms with 4 Ti atoms from the POSCAR file of SrTiO₃ supercell. When the genetic algorithm creates an individual representing the positions in which the Nb atoms will be placed, then the actual POSCAR file for this individual is created from the template file. For this system of Nb-doped SrTiO₃, there are 1365 possible configurations. Exhaustively evaluating all these configurations would take almost 3 months on a 10-node Linux cluster.

In our calculations, we used the free energy as the fitness function of our genetic algorithm based on plane wave density functional theory implementation of the Vienna Ab initio Simulation Package (VASP 5.3) [108]. The exchange and correlation potential

was treated within the Generalized Gradient Approximation (GGA) with the Perdew-Burke-Ernzerhof (PBE) functional [110]. The interaction between ions and electrons was described by the projector-augmented wave (PAW) method [111]. The cutoff for the kinetic energy was set to 520eV for all calculations with respect to ENMAX values of corresponding elements in POTCAR file.

3.2 RESULTS

We implemented our genetic algorithms in Matlab, and tested their performances using the fitness values obtained for all the 1365 possible configurations using exhaustive search. We compared our different implementations of the genetic algorithm with each other and with exhaustive search to see how much our algorithms can speed up the whole optimization process.

3.2.1 13% Nb-doped SrTiO₃

In this system, the supercell is created from 15 unit cells (5x3x1). There are 15 Sr, 15 Ti and 45 O atoms in the supercell, and the goal is to find the doping positions with the lowest energy for 2 Nb atoms out of 15 Ti positions. There are 105 possible doping configurations in total for these experiments if running exhaustively. Calculating the free electronic energy of each configuration using VASP DFT package needs 15-30 hours on our Linux cluster computing node with 12-Core 2.46Ghz CPU with 24 GB ram. Although this search space is small for a genetic algorithm to expose its ability, we did this experiment to see how our genetic algorithm's behavior changes from small search spaces to larger ones. The fitness landscape space is shown in Figure 3.7. We duplicated the values for symmetry view in 3D. It means we only have individuals as, for example [3, 5] but not [5, 3], since they occupy the same positions for dopant elements. Also, we

set the values for couples like [3, 3] to little lower than the maximum value of all results. The search space showed that there exists gradient information that genetic algorithms can exploit to quickly find the optimal or near-optimal results.

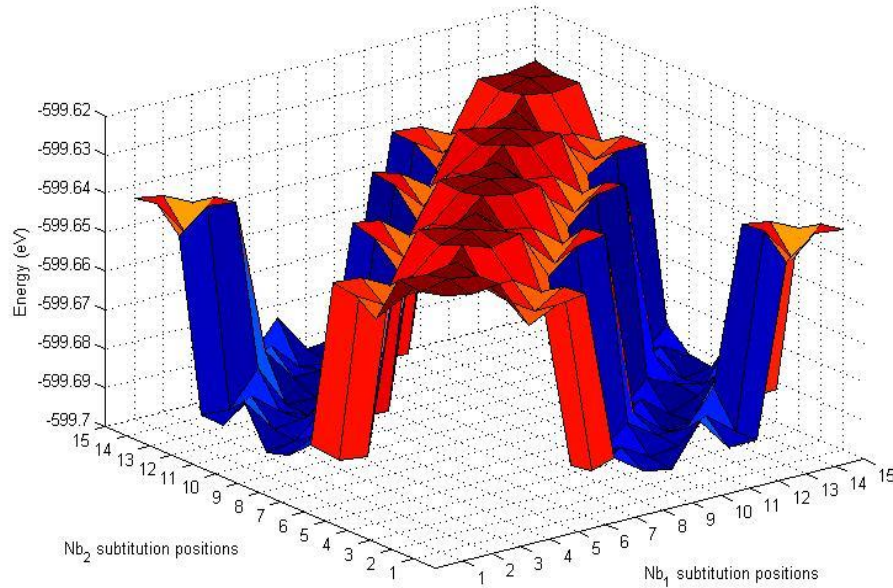


Figure 3.7: The search space for 13% Nb-doped SrTiO₃

The first experiment tested the ability of different GA implementations for finding the optimal doping configurations given a fixed numbers of evaluations. Here only 40 fitness evaluations out of 105 are allowed, and compared the best found results of three GAs compared with the exhaustive search results. Additionally, 10 independent runs were executed for each algorithm to check robustness of the algorithms (Table 3.1). As a result, with only 38% of total evaluations, even the basic GA can get one of the top 3 results in most of the 10 runs. The GA-SS algorithm had similar or slightly better performance. In most cases, it found better results than the basic, but sometime it became stuck in local optima and takes longer to get good results. The GA-SC with statistical crossover works best as indicated by Table 3.1. The numbers in columns show the ranks

of the best results found by each algorithm. If it is 2, then it means 2nd best solution is found for that run. For most of the runs, it successfully located the best solution and it finds one of the top 3 results with only 40 out of 105 evaluations. That means a saving of $65 \times 15 = 975$ hours of computing resources on a 12-core high-end Computer.

Table 3.1: Results of 10 independent runs for algorithms given 40 fitness evaluations

| runs | GA-basic | GA-SS | GA-SC |
|------|----------|-------|-------|
| 1 | 4 | 2 | 2 |
| 2 | 2 | 1 | 1 |
| 3 | 7 | 2 | 3 |
| 4 | 4 | 3 | 1 |
| 5 | 1 | 6 | 1 |
| 6 | 3 | 1 | 1 |
| 7 | 2 | 1 | 1 |
| 8 | 3 | 10 | 1 |
| 9 | 1 | 2 | 3 |
| 10 | 2 | 4 | 3 |

The second experiment tested how fast our algorithms find the lowest energy doping positions. The algorithms run as long as possible and are then checked when they converged and in what generations they found the lowest energy (worst-case scenario). Table 3.2 shows the final results. The numbers in columns show that the number of individuals evaluated to find the best solution out of 105 evaluations. It shows that the GA-SC not only find one of the top 3 best solutions out of the 40 fitness run experiments as shown in Table 3.1, it also found such solutions much faster than the other two algorithms. On average it founds the best solution when about 46 evaluations have been used, which is much better than the other two algorithms.

Table 3.2: 10 independent runs of three algorithms to check when they find the best solution

| runs | GA-basic | GA-SS | GA-SC |
|---------|----------|-------|-------|
| 1 | 58 | 36 | 20 |
| 2 | 34 | 92 | 28 |
| 3 | 42 | 76 | 28 |
| 4 | 42 | 28 | 68 |
| 5 | 74 | 84 | 76 |
| 6 | 50 | 68 | 68 |
| 7 | 26 | 52 | 60 |
| 8 | 82 | 28 | 44 |
| 9 | 50 | 28 | 44 |
| 10 | 66 | 84 | 20 |
| average | 52.4 | 57.6 | 45.6 |

3.2.2 27% Nb-doped SrTiO₃

In this system, the supercell is created from 15 unit cells (5x3x1). There are 15 Sr, 15 Ti and 45 O atoms in the supercell. The goal is to find the best positions for 4 Nb atoms to substitute with 4 Ti atoms. In this real-world optimization problem, 1365 DFT calculations are needed to find the best dopant positions if done exhaustively and each takes approximately 15 hours on a 12-core 2.4GHz CPU with 24 GB ram Linux computing node.

Similar to 13% Nb-doped SrTiO₃ experiments, we tested the ability of our different GA implementations for finding the optimal doping configurations given a fixed numbers of evaluations. Here only 400 fitness evaluations are allowed out of 1365, and compared the best found results of three GAs with the exhaustive search results. Additionally, 10 independent runs were executed for each algorithm to check the robustness of the algorithms (Table 3.3). With only 32% of total evaluations, the basic GA can get one of the optimal results in half of the 10 runs. The GA-SS algorithm had similar performance. In some cases, it found better results than the basic GA. The GA-SC

with statistical crossover works best as indicated by Table 3.3. The numbers in columns show the ranks of the best results found by each algorithm. If it is 2, then it means 2nd best solution is found for that run. For all of the 10 runs, it successfully located the best solution and it finds one of the top 3 results with only 440 out of 1365 evaluations. That means a saving of $915 \times 15 = 13725$ hours of computing resources on a 12-core high-end Computer.

Table 3.3: Results of 10 independent runs for algorithms given 400 fitness evaluations

| runs | GA-basic | GA-SS | GA-SC |
|------|----------|-------|-------|
| 1 | 4 | 1 | 3 |
| 2 | 1 | 2 | 1 |
| 3 | 4 | 1 | 1 |
| 4 | 1 | 3 | 2 |
| 5 | 1 | 1 | 1 |
| 6 | 6 | 1 | 3 |
| 7 | 5 | 3 | 3 |
| 8 | 1 | 5 | 1 |
| 9 | 4 | 3 | 2 |
| 10 | 1 | 4 | 1 |

Similarly, the second experiment tested how fast our algorithms find the lowest energy doping positions for this real-world computational doping experiments. Thus, the algorithms run as long as possible until they find the optimal doping position. We then checked when they converged and in what generation (with how many fitness evaluations) they found the lowest energy (worst-case scenario) configuration. Figure 3.8 shows the final results. It shows that the GA-SC not only find one of the top 3 best solutions out of the 440 fitness run experiments as shown in Table 3.4, it also found such solutions much faster than the other two algorithms. The numbers in columns show that the number of individuals evaluated to find the best solution out of 1365 evaluations. On

average it finds the best solution when on average about 350 evaluations have been used, which is much better than the other two algorithms. Compared to exhaustive search, it uses only about ¼ evaluations to find the optimal solution, which is a great saving in terms of computational resource and speeding up computational doping experiments.

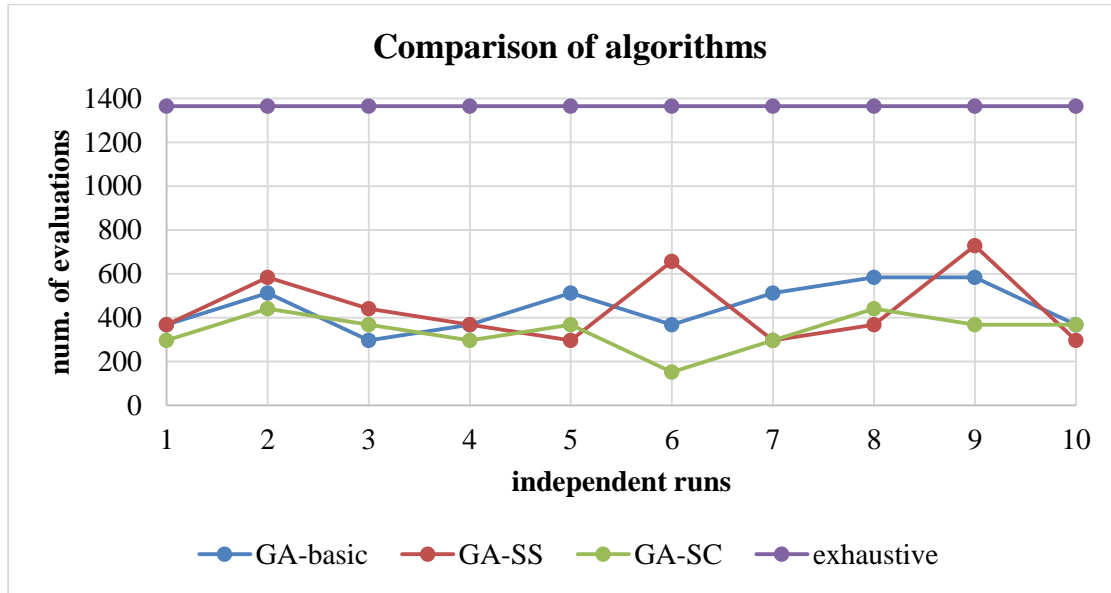


Figure 3.8: Comparison of the number of evaluations used to find the optimal solution for three algorithms

Table 3.4: 10 independent runs of three algorithms to check when they find the best solution

| runs | GA-basic | GA-SS | GA-SC |
|---------|----------|-------|-------|
| 1 | 368 | 368 | 296 |
| 2 | 512 | 584 | 440 |
| 3 | 296 | 440 | 368 |
| 4 | 368 | 368 | 296 |
| 5 | 512 | 296 | 368 |
| 6 | 368 | 656 | 152 |
| 7 | 512 | 296 | 296 |
| 8 | 584 | 368 | 440 |
| 9 | 584 | 728 | 368 |
| 10 | 368 | 296 | 368 |
| average | 447.2 | 440 | 339.2 |

3.3 CONCLUSION

In this chapter, we developed genetic algorithm based approach for identifying optimal doping position assignment for single element dopant substitutions. For the base material SrTiO_3 , Niobium (Nb) was chosen to substitute with Titanium (Ti) atoms. A supercell of 75 atoms was created for SrTiO_3 and 4 of the 15 Ti atoms were substituted with Nb atoms. The whole search space has 1365 different possible configurations. Our GA finds the top 3 best configurations with the lowest fitness by only running 25% of total evaluations.

CHAPTER 4

GENETIC ALGORITHMS FOR MULTIPLE DOPANT ELEMENTS

This chapter describes the technique and evaluation of finding optimal doping configurations with multiple dopant elements using genetic algorithms. Rather than looking for the best configuration for a single type of dopant element and testing another dopant element and finally comparing them, this algorithm can search and evaluate multiple dopant elements at the same time and obtain the best (most stable) configuration in terms of the electronic free energy. The developed algorithm can thus be used for search mixed dopant materials.

In this chapter, five candidate elements were chosen; Niobium (Nb), Rhodium (Rh), Gallium (Ga), Aluminum (Al) and Indium (In) to substitute with Titanium (Ti) elements on Strontium Titanate (SrTiO_3) material. For the base material SrTiO_3 , the supercell model now consists of 75 atoms. At this time, 2 Ti atoms are to be substituted with any of the 5 candidate elements. Both substitution positions can be filled with atoms of the same or different candidate elements (Figure 4.1).

The binary representation of a GA individual is very similar to the GA for single element doping as discussed in Chapter 3, except that it represents the positions with different integers for atoms of different elements. In this representation, the occupied positions will be represented by the integers which shows the *id* of the elements. If, again

there are 5 dopant candidates, the *id* for these elements will be 1 to 5. The bits showing with 0 still represent not-substituted Titanium positions (Figure 4.2).

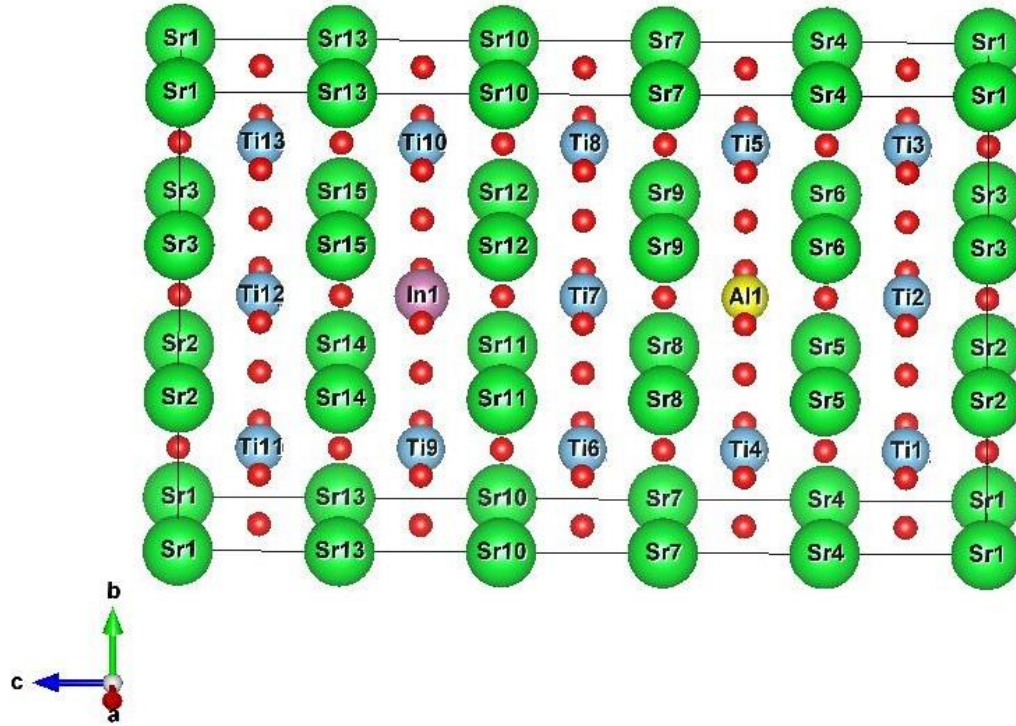


Figure 4.1: The grid of Al-In doped SrTiO₃. One Al atom substituted with one Ti at 5th position, and one In atom substituted with one Ti atom at 11th position

| | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 4 | 0 | 0 | 0 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

Figure 4.2: Binary representation of an individual that places Al atom 5th position and In atom at 11th position

Different from the single-dopant-element GA, statistical crossover was not used here. Recalling GA-SS, the statistics for each position is kept to determine how favorable a position is. If the individuals with that position set as 1 get high fitness value, then this position will get higher chance for selection. However, there are multiple dopant elements and these elements may prefer different positions for different reasons. For

example, while Niobium atoms prefer to be doped into SrTiO_3 around the center, Indium atom may prefer the corners or sides. Thus, a different crossover method is implemented different from the single-dopant-element GA.

4.1 CROSSOVER

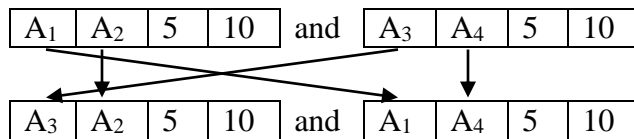
Since we have multiple dopant candidates this time, three more crossover options were added to the existing genetic algorithm to handle different cases. There is one main crossover function and these sub-options are chosen in this function based on different cases. These crossover options behave differently, thus users can choose whichever works for their cases. The following sub-sections will explain how the crossover method is implemented

4.1.1 Crossover only elements

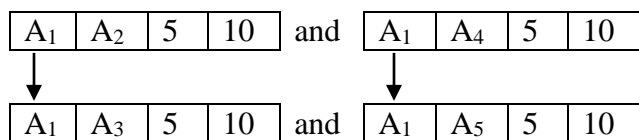
Here we implemented the crossover option when both parents occupy the same positions. We assume these positions are highly favorable and do not want to lose them. Instead, we exchange the elements at the positions using the crossover operator.

Let the parents be $A_1_A_2_5_10$ and $A_3_A_4_5_10$. In this case, there are still 3 options.

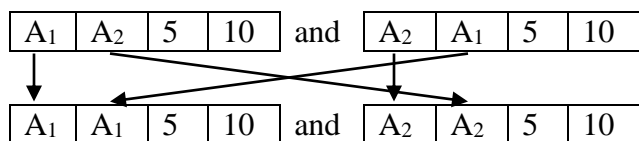
If atoms, A_1 , A_2 , A_3 , A_4 , are all different, then a single point crossover over elements is implemented and the offspring will be:



If one of the atoms are the same from A_1 , A_2 , A_3 , A_4 , then the same atom is kept and 2 different atoms are chosen randomly from our atom list. Let the parents be $A_1_A_2_5_10$ and $A_1_A_4_5_10$. New offspring will be created in the following way:



Finally, if both elements and positions are the same for both parents but in different order, the elements are swapped to create new offspring. Let the parents be $A_1_A_2_5_10$ and $A_2_A_1_5_10$. New offspring will be created in the following way:

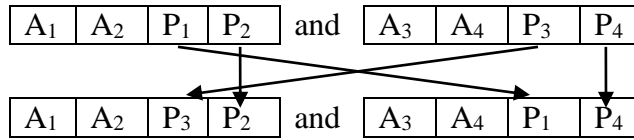


Recalling the previous question: What if both elements and positions of the parents are the same? The answer is, the checking procedure does not allow the same parent to stay in the population. As soon as one individual is created, the algorithm checks if this individual has been created before or not. If it has been created before, immediately another is created. In this way, the population never has matching individuals at the same time.

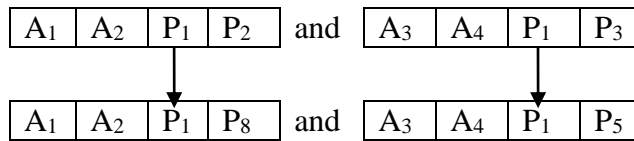
4.1.2 Crossover only positions

This crossover option is implemented when we want to crossover only dopant positions. If there is one atom superior to others or one pair of atoms superior to the other pairs, then this will dominate the population in later generations. Thus, it is a good idea to keep these atoms unchanged and try to find better positions for this configuration. In this case, again there are 3 different cases to be considered.

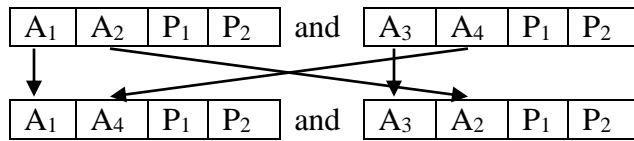
Let the parents be $A_1_A_2_P_1_P_2$ and $A_1_A_2_P_3_P_4$. In this case, there are still 3 options. If positions P_1, P_2, P_3, P_4 , are all different, then implementing a single point crossover over these positions, the offspring will be:



If one of the positions is the same from P_1, P_2, P_3, P_4 , then the same position is kept and two different positions are chosen randomly from the not-occupied positions. Let the parents be $A_1_A_2_P_1_P_2$ and $A_3_A_4_P_1_P_3$. New offspring will be created in the following way:

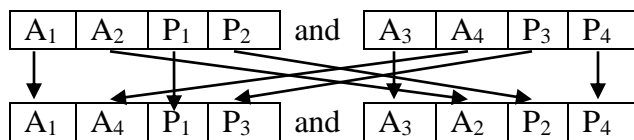


Finally, if the doping positions are the same for both parents, then crossover the dopant elements (by calling *xover_elems_only*) to create new offspring. Let the parents be $A_1_A_2_P_1_P_2$ and $A_3_A_4_P_1_P_2$. New offspring will be created in the following way:



4.1.3 Crossover elements and positions

This crossover option is implemented to keep the diversity in the population. The main crossover function will first call this option if there is no common position of the parents. Basically, the crossover of the atoms and the positions is done at the same time. If the parents are $A_1_A_2_P_1_P_2$ and $A_3_A_4_P_3_P_4$, then, the new offspring will be created in the following way:



Although these cases are chosen by default, users can change the behavior of the crossover operator. For instance, if both parents have the same positions it is preferable to keep those positions and crossover the atoms. Instead, users may change the positions and keep the atoms, or even may crossover both positions and atoms.

4.2 MUTATION

The mutation operator here is very similar to the single dopant element GA mutation, except that it can mutate the atoms too. Although one mutation option was set as default, it can be implemented in different ways. The default operator chooses one substituted position at random and swaps that position with non-substituted position.

Let Al_Rh_2_8 be an individual to be mutated. The mutation operator chooses 2 or 8 randomly and replaces it with one of [1,3,4,5,6,7,9,10,11,12,13,14,15] and the new offspring will be Al_Rh_2_10.

In the same way, the positions can be kept unchanged and the atoms mutated. In the mutation operator, a coin toss operation is made to decide if either atom mutation or position mutation will be implemented.

Another idea for mutation operator would be to apply mutation to both atoms and positions. These options were written in the code and left to the user's preference.

4.3 MATERIAL PREPERATION

The material preparation for evaluation experiments is similar to previous chapter. In addition to Niobium atom, four more types of atoms were prepared, Aluminum, Indium, Rhodium, and Gallium. Related files, such as INCAR, POSCAR, POTCAR and KPOINTS were also prepared to run VASP experiments. Initially the lattice parameters were optimized for the POSCAR file and a supercell was created consisting of 75 atoms,

which has 15 Strontium atoms, 15 Titanium atoms, and 45 Oxygen atoms. The POSCAR file was used as a template to create actual POSCAR files which has different doped configurations.

For fitness calculations, the VASP parameters were carefully chosen. Monkhorst-Pack is used to create the grid with 5x3x1 dimensions grid for KPOINT file. In the INCAR file, the cutoff for the kinetic energy was set to 520eV for all calculations with respect to ENMAX values of the corresponding elements in the POTCAR file. The interaction between ions and electrons was described by the projector-augmented wave (PAW) method. The exchange and correlation potential was treated with the Generalized Gradient Approximation (GGA) with the Perdew-Burke-Ernzerhof (PBE) functional. The maximum number of ionic steps (NSW) was to 200 to allow as much ionic calculation as possible. EDIFF, which is used to global break condition for electronic calculation, was set to 10^{-4} .

After the electronic structure and total energy are calculated, several high level material properties can be calculated. For instance, the electronic conductivity and ionic conductivity can be calculated for different dopant elements and the effect of dopant elements can be compared [112].

4.4 RESULTS

This sub-section describes the evaluation of the proposed genetic algorithm on 13% XY-doped Strontium Titanate (SrTiO_3) system where X and Y can be any of 5 candidate dopants, Niobium (Nb), Aluminum (Al), Indium (In), Rhodium (Rh) and Gallium (Ga). These 5 dopant elements will be substituted with any two of 5 candidate dopant elements to find the lowest free energy of VASP calculations.

Since we have 15 positions and among them 2 positions will be substituted from any of 5 dopant elements, totally 2625 different combinations are possible. Different from single dopant element experiments, these multiple dopant experiments take much more time to calculate the free energy via VASP. The configurations with different dopant elements make the VASP calculations much more complex. For example while the configuration Nb_Nb_3_5 takes 15 to 30 hours to calculate the free energy, the configuration Nb_Rh_4_9 may take 175 hours to finish calculation, since one additional different element is in the system.

In this test, the population size was set to 40 and 10 generations were executed. The elite probability was set to 0.1. As a selection method, tournament selection was used and tournament size was set to 2. Mutation operator was used as described in section 4.2 and mutation probability was set to 0.1.

During our experiments, we first calculated the free energies for all the possible doping configurations to get the ground truth via exhaustive search and then the genetic algorithm was tested against all known fitness values. In this experiment, the known fitness values vary between -571.796 and -599.695. Since genetic algorithms are heuristic search methods, the initial population is chosen by random. If the individuals in the initial population are close to global optimum, then the algorithm can easily converge in a couple generations. Figure 4.3 shows how the population is converged if the initial population is close to global optimum. The population finds the best individual at 6th generation and converged.

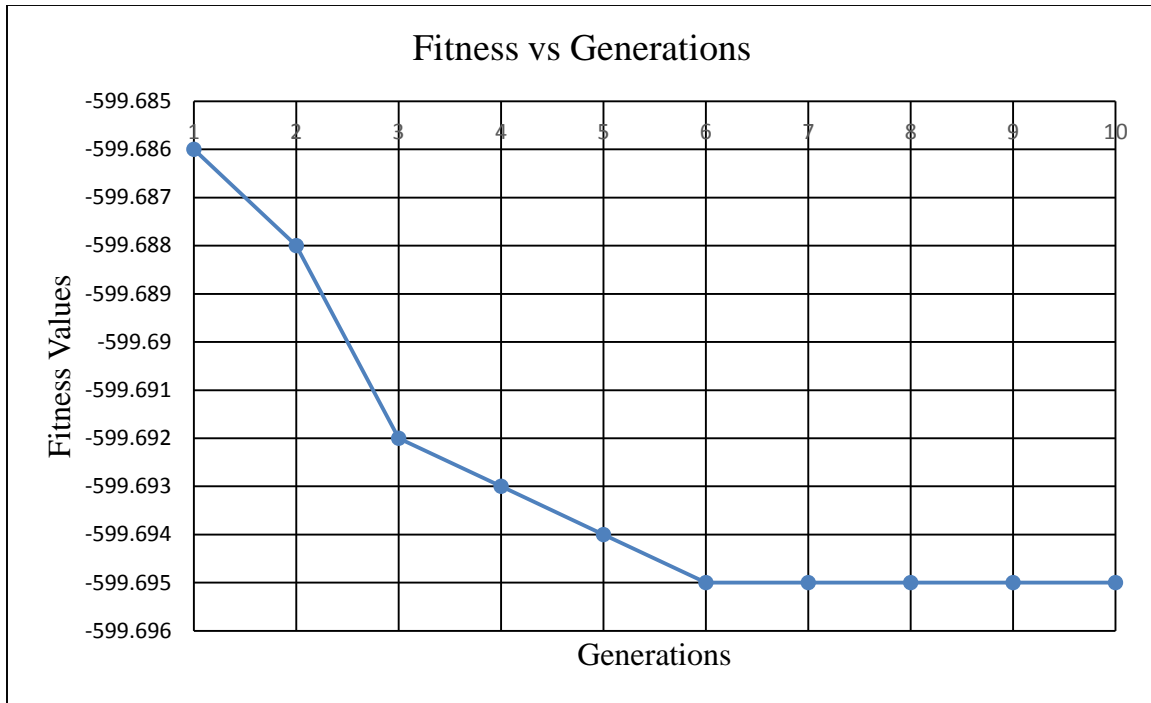


Figure 4.3: One of the successful run of finding the lowest fitness value

If the individuals in initial population are far away from the global optimum, it may take more generations to converge if at all in a given number of generations. In this case, with the power of crossover and mutation, the population should be able to find better individuals as early as possible. Figure 4.4 shows fitness values over generations if the initial population is not close to global optimum. The population makes a big jump at search space in early generations and then converges in 7th generation.

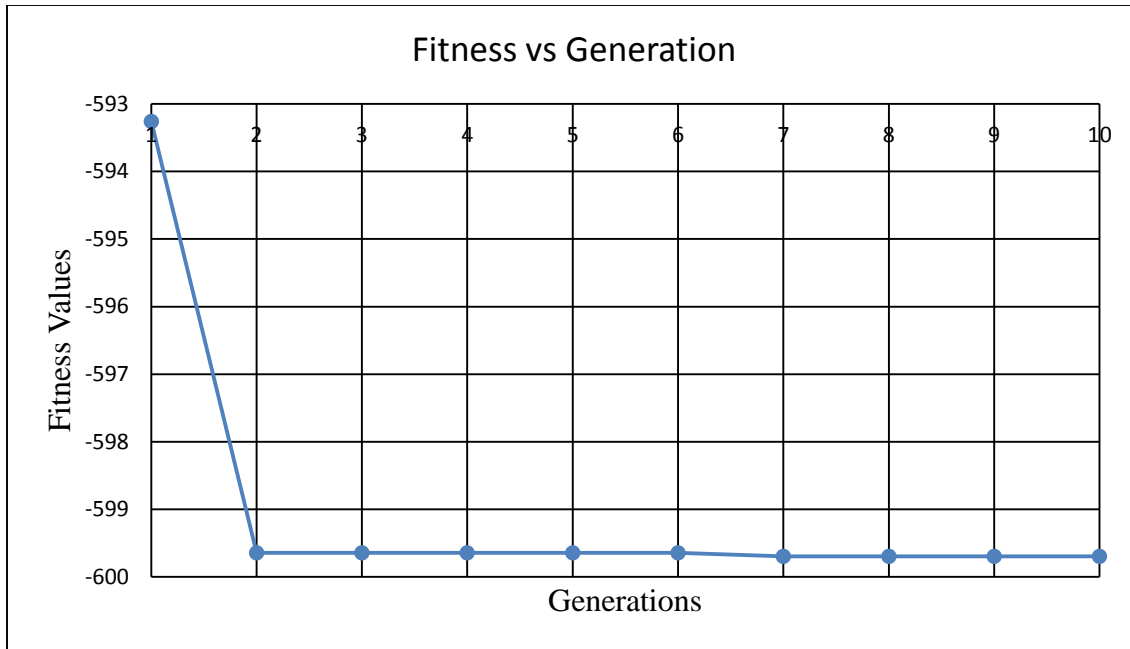


Figure 4.4: Another successful run of finding the individual with minimum fitness value starting from far away from the global minimum

To see how many evaluations can be saved with the genetic algorithm compared to exhaustive search to find the optimal doping configurations, only a small number of evaluations was allowed in our GA experiments. This tests the ability of the genetic algorithm for finding the optimal doping configuration. For this purpose, only 400 evaluations were run out of 2625. Then those 400 best found individuals were checked against the best known individuals. It means only 15% of all configurations were ran and 85% of configuration were saved. At the same time, the robustness was tested by executing the genetic algorithm 100 times. Figure 4.5 shows the results of 100 runs and out of 72 of them the genetic algorithm found the best configuration. Out of 22 runs, the genetic algorithm finds the second best configurations, out of 3 runs the third-best configuration was found. Among all 100 runs, only two times the genetic algorithm failed to find the one the top 3 configurations.

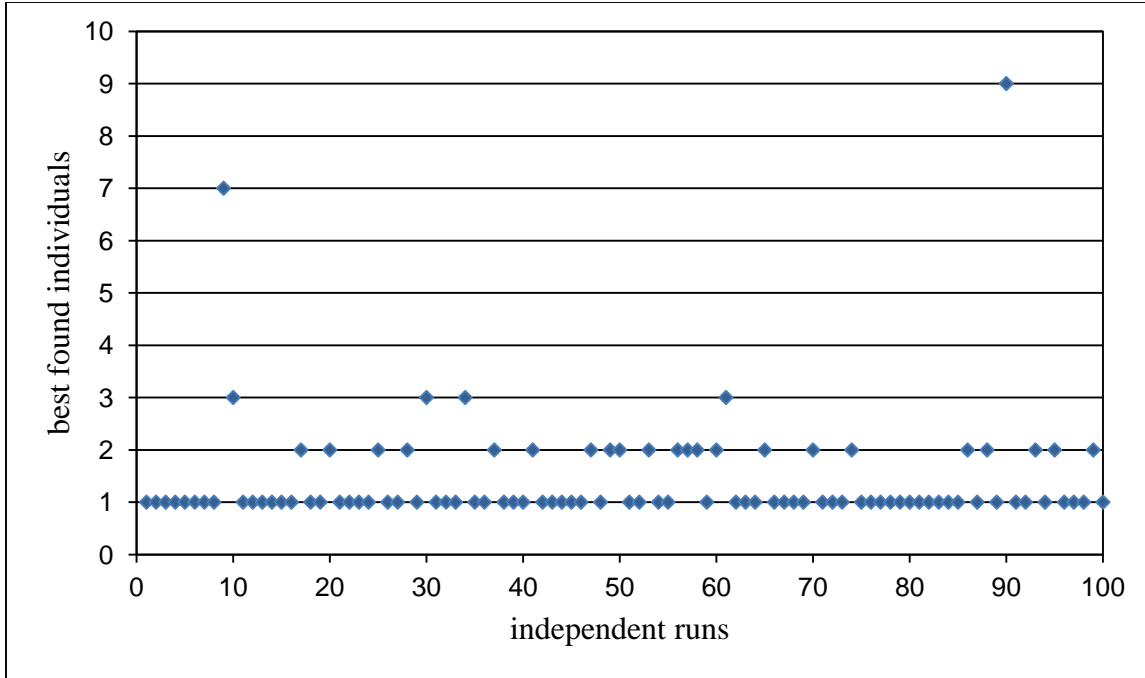


Figure 4.5: The result of 100 individual runs of the genetic algorithm

These results showed that, by running only 15% of the VASP simulations in exhaustive search, our genetic algorithm found the best results in 72 out of 100 runs. In addition, the genetic algorithm found one of the top 3 configurations in 98 out of 100 runs. This means by running our genetic algorithm with only 15% of exhaustive search for computation doping experiments, lots of experiments and calculation time can be saved.

4.5 CONCLUSION

In this chapter, we developed genetic algorithm based approach for identifying optimal doping position assignment for multiple element dopant substitutions. For the base material SrTiO_3 , Niobium (Nb), Aluminum (Al), Gallium (Ga), Rhodium (Rh) and Indium (In) were chosen to substitute with Titanium (Ti) atoms. A supercell of 75 atoms was created for SrTiO_3 and 2 of the 15 Ti atoms were substituted with any combination of these five atoms. The whole search space has 2625 different possible configurations.

Our GA finds the best configuration with the lowest fitness value at 72 cases out of 100 and top 3 best configuration at 92 out of 100 by only running 15% of total evaluations.

CHAPTER 5

GENETIC PROGRAMMING FOR MULTIPLE ELEMENTS

DOPING

Genetic programming is a genre of genetic algorithms where the typical individual representations are syntax trees composed of internal functional nodes and end terminals [113]. Starting with an initial population of tree-structured individuals (materials), a fitness evaluation process will be applied to all individuals. Then, the individuals with higher fitness values will be selected for crossover and mutation according to the survival of the fittest principle. A new population will then be generated, and the process loops until the termination criteria is met. The whole framework is shown in Figure 5.1 using SrTiO_3 as the example. Below, we define the components of the framework, which will be implemented using Open-Beagle, a C++ framework for genetic programming [114], [115].

This chapter will first describe the generic version of genetic programming for material doping, then we will describe how we implemented in our project.

5.1 GENERIC VERSION OF GENETIC PROGRAMMING FOR MATERIAL DOPING

The doping space of a given material system such as ABO_3 perovskite consists of the allowable doping sites, dopant elements of each site, and amount of doping elements.

While it is possible to apply doping on A and B site at the same time, it is also possible to

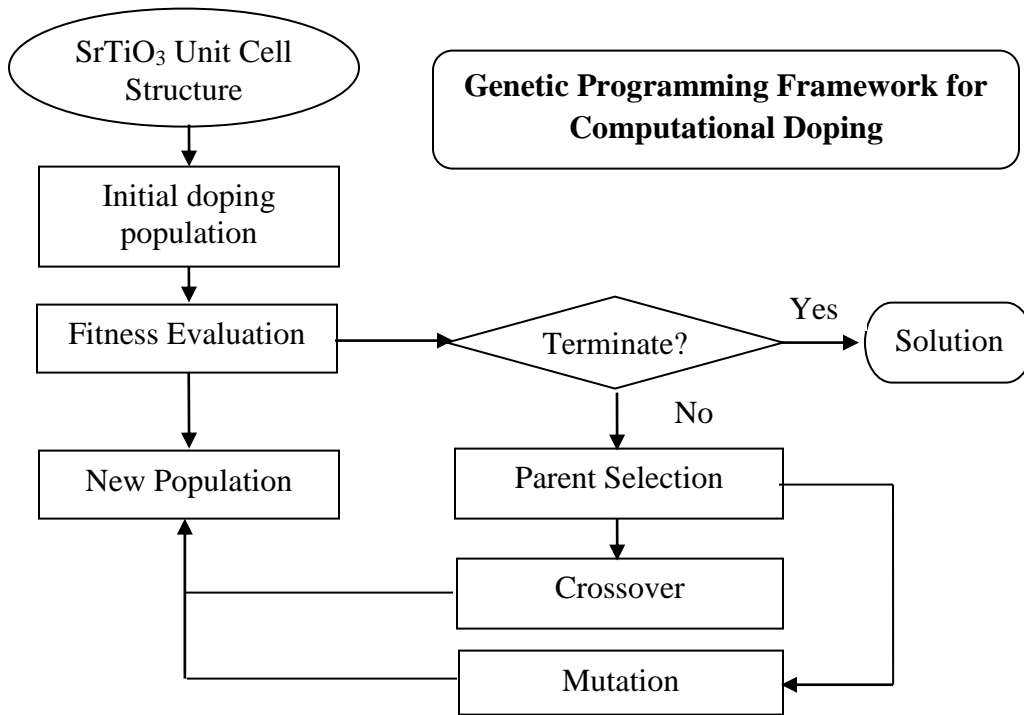


Figure 5.1: GP-based computational doping framework

apply vacancy on either side. The doping process can be mapped as a sequential process of adding dopant elements to the base material, which can be modeled by a GP syntax tree.

A variable-length GP tree structure representing a doped material is shown in Figure 5.2. It represents the Nb-doped SrTiO_3 material, $\text{Sr}_{0.95}\text{Ti}_{0.8}\text{Nb}_{0.2}\text{O}_3$ with 0.05 vacancy on Strontium side. The second tree level represents the allowed doping sites. Empt means vacancy; Nb means doping with Niobium; EndP means no more allowing doping on that site; 0.05 and 0.2 are dopant amounts as percentages. With this mapping, finding the optimal doped material is equivalent to finding the optimal GP tree. To search in this variable tree structure doping space, we used Genetic Programming (GP) [104], [116]–[118] as the efficient sampling technique combined with DFT calculation for computational doping. Genetic programming is a class of evolutionary algorithms

inspired by biological evolution, and it has been used in a wide variety of fields, including bioinformatics, quantum computing, mathematical algorithms, and mechanical systems. One of the major benefits of a genetic search is that complementary features of two individuals can be combined to generate better offspring solutions. This reduces the complexity of search space and eliminates the most infeasible doping configurations.

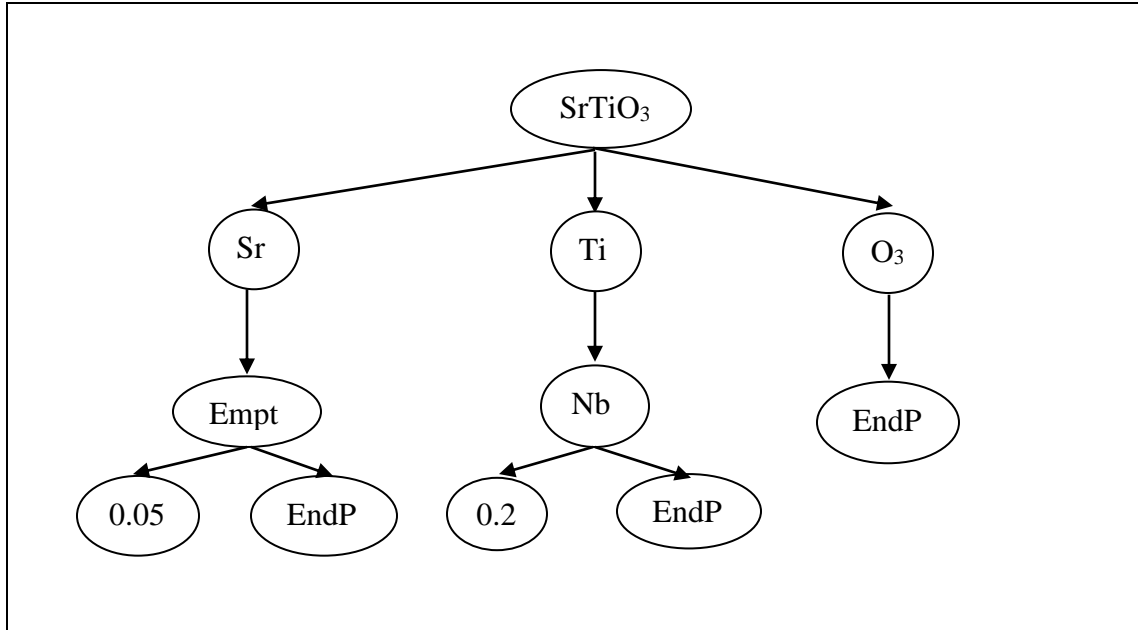


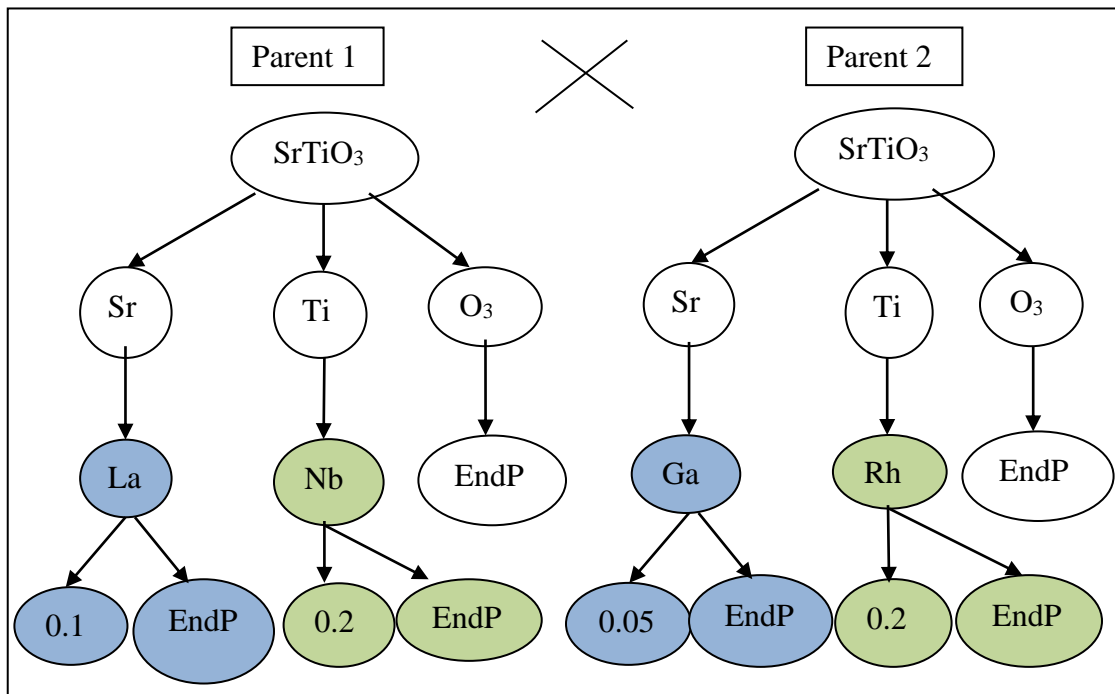
Figure 5.2: Representation of the $\text{Sr}_{0.95}\text{Ti}_{0.8}\text{Nb}_{0.2}\text{O}_3$ tree structure

5.1.1 GP Tree Representation

A GP syntax tree is composed of two types of nodes. First type is the GP functions which have branches. The second type is called GP terminals, which are ending nodes of the GP tree. GP algorithms can be further classified into non-typed GP and strongly typed GP, in the latter case, all GP nodes have a return type and only GP nodes with identical return type can be exchanged or replaced. In this work, we used strongly typed GP for computational doping.

5.1.2 Genetic Crossover and Mutation in GP

Crossover and mutation are the two major approach to introduce variation during evolutionary search of GP. These two operators can be conveniently implemented by the constrained strongly typed genetic programming (STGP) approach [119]. STGP is an extended version of genetic programming that enforces data type constraints, which provides a great reduction in representation and search space. This significantly decreases the search time and/or improves the generalization performance of the solutions. Figure 5.3 shows the crossover operator between two individuals. Only the sub-trees of the same doping site can be exchanged. The tree structure can be mutated by changing the dopant element at a given node, modifying the dopant composition parameter, or replacing a subtree with a randomly generated subtree.



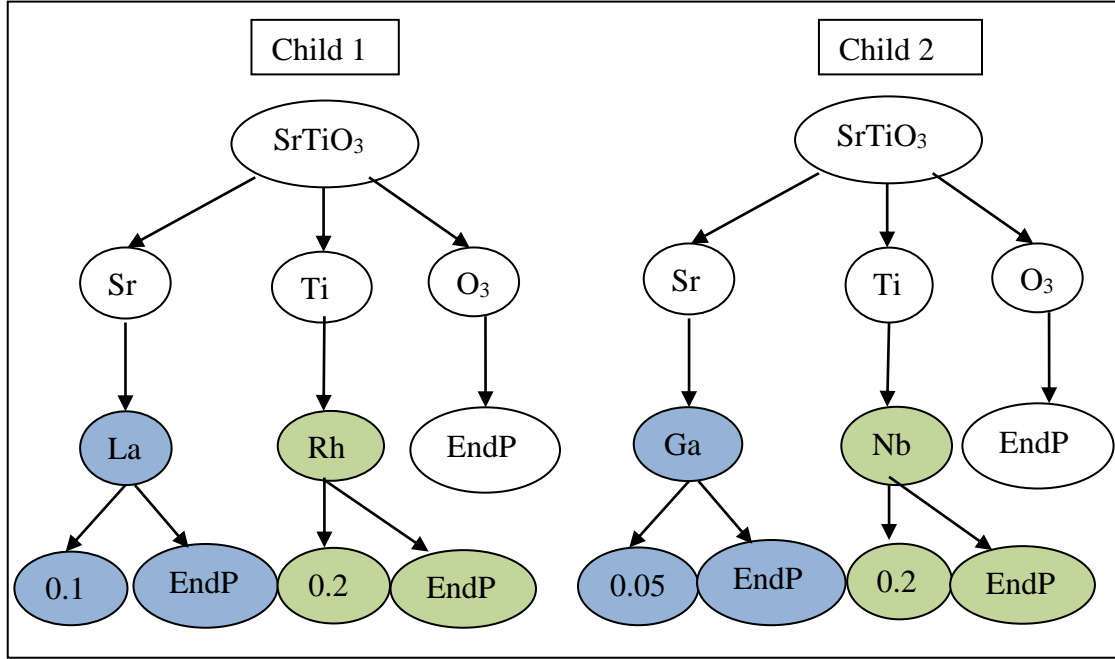


Figure 5.3: GP-Crossover operator for La/Nb-doped SrTiO₃ and Ga/Rh-doped SrTiO₃

5.1.3 Initial Population Seeding

Individuals of initial population can be generated according to the strongly typed GP rule. While new arbitrary individuals can be randomly generated, prior experimentally verified doped materials reported in the literature can also be used to seed the starting population.

5.1.4 Fitness Evaluation

This is the most challenging and time-consuming part of the computational doping process. Since multiple conflicting properties usually need to be optimized while seeking ideal doped materials, the genetic search is defined as a multi-objective optimization problem, for which the evolutionary algorithms are among the best methods [120]–[123]. Since the objectives are usually conflicting, it is not easy to find a single optimal solution that respects all objectives.

The material property calculation is based on the supercell approach for DFT studies on doped materials [76]. DFT simulation will be calculated by the Vienna Ab initio Simulation Package (VASP), with some improved features. There are four steps in the DFT based property/fitness calculation: 1) Supercell generation; 2) doping/vacancy position assignment; 3) DFT calculation; 4) property calculation. One of the major obstacles here is in Step 2, in which a large number of possible configurations of the doped elements or vacancy can exist in the lattice structure. In this case, we need to identify the configuration with the lowest energy. In the previous study [76], Suthirakun et al. used an exhaustive search to achieve that, which is too computationally costly for the proposed computational doping.

VASP-based DFT calculations are extremely time-consuming. For example, in our experiments a supercell of 75 atoms with 2 dopant elements takes 15-30 hours to run on a 12 core CPU Linux node. When increase the number of dopant elements, the running time increases days to weeks. Several approaches can be used to speed up the fitness evaluation step: 1) Multi-resolution first principle calculations that mix VASP with faster lower-resolution DFT calculation codes such as GULP [124], [125] can be used for calculating the electronic structure. 2) The statistical material prediction models developed below can be utilized, or performance comparison results in the literature will be used. This is possible because in a genetic search with tournament selection, only relative performance among individuals is needed for parent selection.

5.2 OUR IMPLEMENTATION OF GENETIC PROGRAMMING FOR MULTIPLE DOPANT ELEMENTS WITH MULTIPLE DOPING RATIOS

In previous section, we proposed the general version of GP based computational doping. However the doping material simulation is done via VASP simulation of supercells, of which partial positions of the atoms are substituted by the doped elements. To avoid the redundancy of the representation, we have developed the following GP approach for computational doping. This section will describe how the genetic programming is implemented to find best doping configurations when multiple dopant elements are used to substitute Ti-site of material SrTiO_3 . The SrTiO_3 supercell consists of 75 atoms (15 Sr, 15 Ti, and 45 O) and is created from 15 unit cells (5 x 3 x 1). The candidate materials for substitution with Titanium are Niobium (Nb), Rhodium (Rh), Indium (In), Gallium (Ga) and Aluminum (Al). The allowed doping ratios are chosen as 15:1 or 15:2, which means either one Ti atom will be substituted with one of the candidate atoms or 2 Ti atoms will be substituted with any of 2 candidate atoms. Total possible configurations can be calculated in the following way: since there is 15 Ti atoms and 5 candidate atoms, there are $15 \times 5 = 75$ possible configurations for single dopant substitution (15:1 doping ratio). Similarly, for 15 Ti positions any 2 of 5 candidate atoms will be chosen leading to $C(15,2) * (C(5,2) + 5) = 2625$ different ways. $C(15,2)$ means 15 choose 2, and it calculates how many different ways can be chosen 2 spaces from 15 possible spaces ($C(15,2) = 105$) (Table 5.1). Notice that the positions cannot be the same because one position can only be occupied by one atom, and the first position will always be smaller than the second position. The purpose for this is to avoid generating duplicate configurations. For example, the individuals Nb_Al_3_5 and Al_Nb_5_3 are the same

configurations, where Nb will be substituted at 3rd position and Al will be substituted at 5th position for both configurations.

Table 5.1: 105 different positions that can be chosen for substitutions for two dopant elements

| | |
|------|------|
| 1 | 2 |
| 1 | 3 |
| 1 | 4 |
| 1 | 5 |
| 1 | 6 |
| 1 | 7 |
| 1 | 8 |
| 1 | 9 |
| 1 | 10 |
| 1 | 11 |
| 1 | 12 |
| 1 | 13 |
| 1 | 14 |
| 1 | 15 |
| 2 | 3 |
| 2 | 4 |
| 2 | 5 |
| 2 | 6 |
| | |
| | |
| | |
| | |
| | |
| 11 | 15 |
| 12 | 13 |
| 12 | 14 |
| 12 | 15 |
| 13 | 14 |
| 13 | 15 |
| 14 | 15 |

C(5,2) means 5 chooses 2 and it calculates how many different ways a pair of dopants can be chosen out of 5, for example {Nb,Rh} is one of them. However it doesn't choose duplicate elements as a pair. Thus the pairs with the same atoms should be added in this set and that 5 in the equation corresponds the pairs with identical atoms, such as

{Nb,Nb}. Thus, the right side of the equation $(C(5,2)+5) = 25$, and this means the pairs for 15:2 doping rate can be chosen in 25 different ways (Table 5.2). This makes the total number of 15:2 doping rate substitutions to be $105*25 = 2625$.

Table 5.2: All possible ordering of 2 elements out of 5 elements

| | |
|----|----|
| Nb | Nb |
| Nb | Al |
| Nb | Rh |
| Nb | Ga |
| Nb | In |
| Al | Al |
| Al | Nb |
| Al | Rh |
| Al | Ga |
| Al | In |
| Rh | Rh |
| Rh | Ga |
| Rh | In |
| Rh | Nb |
| Rh | Al |
| Ga | Ga |
| Ga | In |
| Ga | Nb |
| Ga | Al |
| Ga | Rh |
| In | In |
| In | Nb |
| In | Al |
| In | Rh |
| In | Ga |

All together for the single and multiple dopant elements, there are $75 + 2625 = 2700$ total possible configurations. Although doping ratios are chosen as up to 15:2, it can be chosen to substitute more Ti atoms such as 15:3 or 15:4, however, it will increase the computational complexity exponentially.

5.2.1 Tree Structure – Representation of Individuals

In genetic programming, all individuals are actually syntax trees. The general tree structure was explained in previous sections. In this work, the tree structure will be simplified to make it more understandable. Since we will only apply doping on the Ti side, the aim is to choose the positions and the atoms which give the best fitness values. Thus, there is no need to include Sr, Ti or O₃ atoms in the tree structure. Our tree structure will represent the substitution positions and substitution elements with Ti.

First, we define the GP functions and the GP terminals. The GP functions are ADD functions for all different atoms. Since there is 5 candidate dopants there are 5 functions AddNb, AddIn, AddAl, AddGa, AddRh, which means whenever the algorithm chooses these functions, the corresponding atom will be substituted with one Titanium atom on one specific position. The position is a terminal of these functions and represents at which position will substitute Titanium atom with corresponding atom. There is one more terminal, named EndP. EndP means there is no more doping on this side. Figure 5.4 shows how does a single dopant element configuration and two-dopant elements configuration are represented.

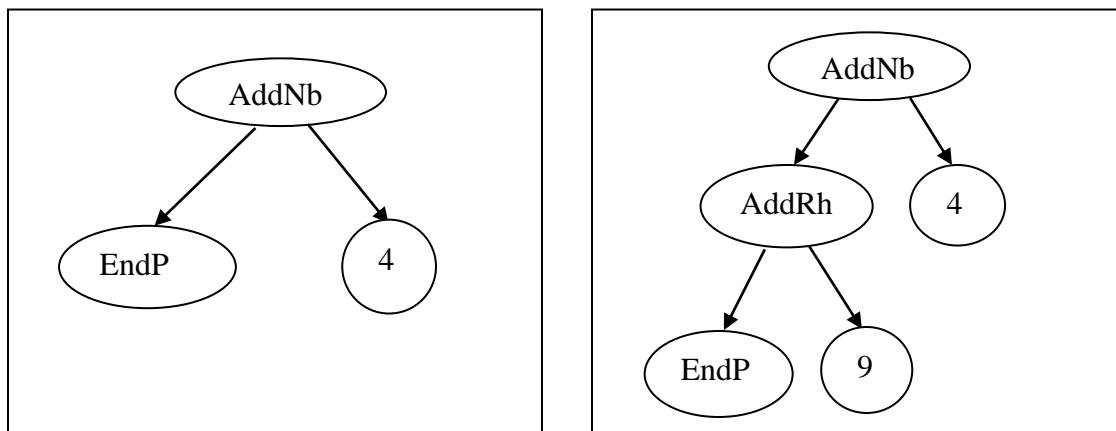
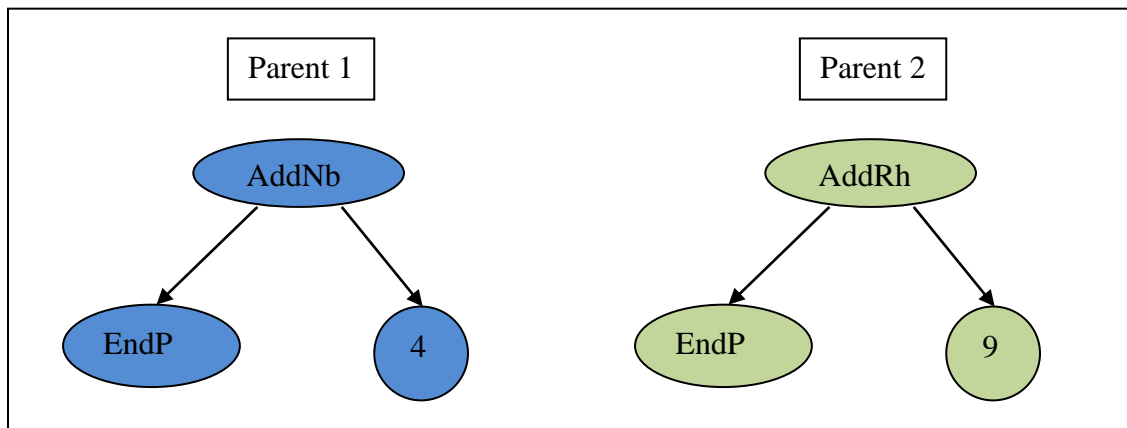


Figure 5.4: Single dopant configuration of Nb at 4th position (left). Two-dopant configuration of Nb and Rh at 4th and 9th positions (right)

In Figure 5.4, the left tree is the individual for substitution Ti with Nb at 4th position. Similarly, the right tree is the individual for substitution of two Titanium atoms with Nb at 4th position and Rh at 9th position. As seen in Figure 5.4 EndP is a terminal point which stops the tree growing, that's how the program can generate single dopant element configurations. EndP is chosen randomly as soon as the other functions such as AddNb, AddRh, etc. Since we are only testing 15:1 and 15:2 doping ratios, the types of trees on the Figure 5.4 are the only valid trees. The OPEN BEAGLE parameter max_tree_depth is set to 2, and this parameter is used to stop growing the tree through more levels. In other words, if a user wants to apply 15:3 doping ratio, this max_tree_depth should be set to 3 to generate configurations which has 3 substitution positions and corresponding atoms.

5.2.2 Crossover

The crossover operator exchanges two compatible subtrees with the same return type of the selected parents and creates two offspring. If the selected parents are only single dopant elements, then crossover exchanges their positions and creates the offspring (Figure 5.5).



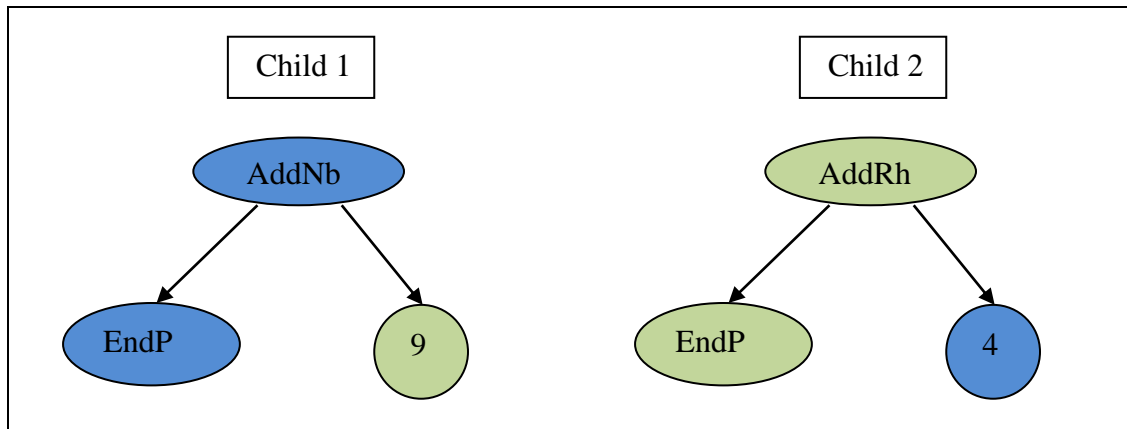
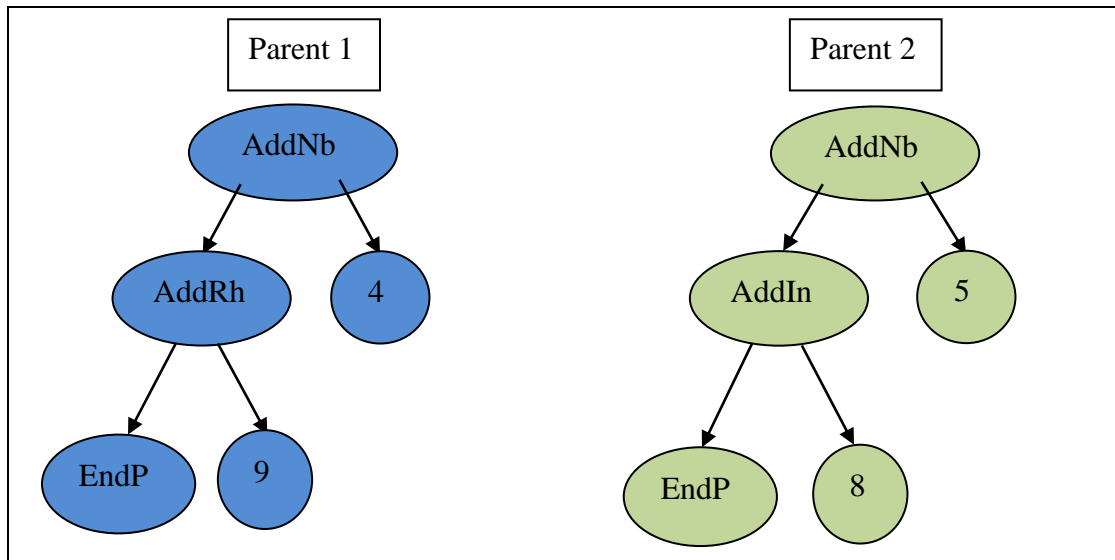


Figure 5.5: Crossover of individuals Nb_4 and Rh_9. The created offspring are Nb_9 and Rh_4

If the crossover is between two-dopant elements individuals, then one subtree is chosen from each parent and exchanges each other (Figure 5.6). If the root is selected as a subtree then the created individual will not be a valid individual, since the number of dopant elements will exceed the maximum number of substitutions for one child. In this case, this individual will be ignored and another subtree will be chosen.



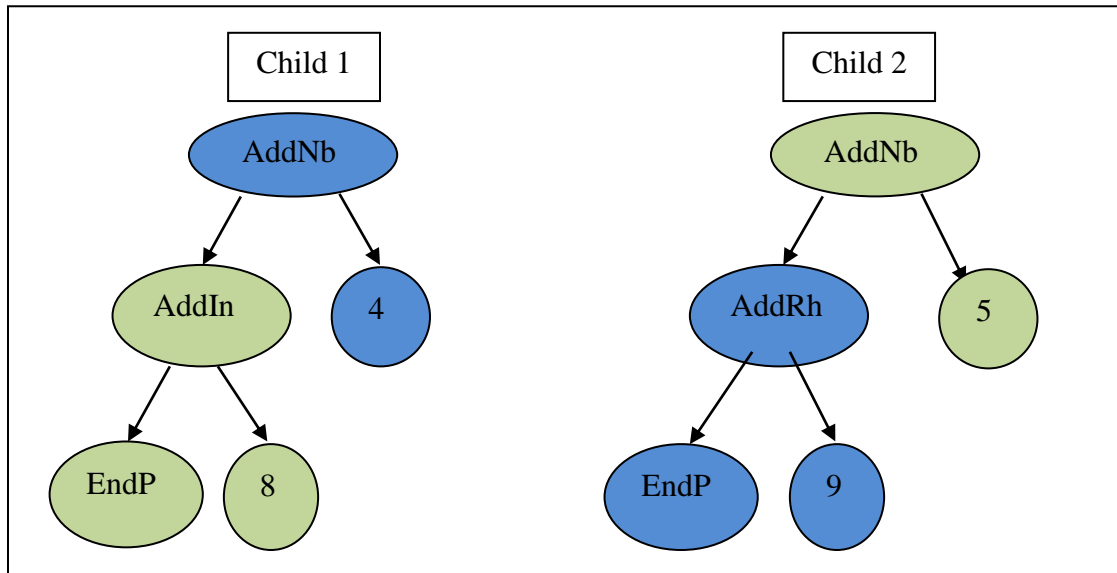
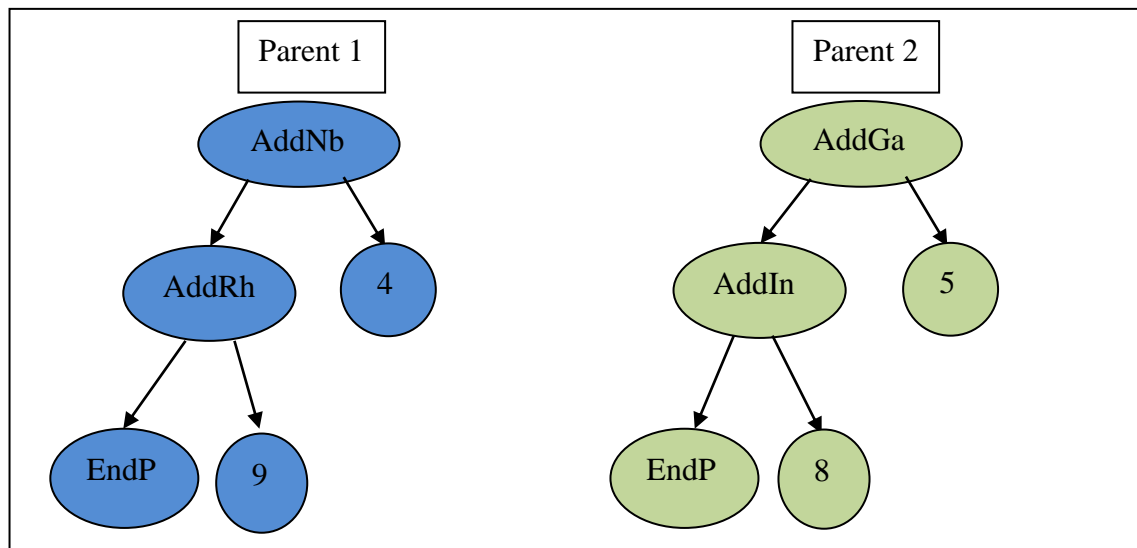


Figure 5.6: Crossover between Nb_Rh_4_9 and Nb_In_5_8. The created offspring are Nb_In_4_8 and Nb_Rh_5_9

In addition to the crossover operator seen in Figure 5.6, another crossover possibility to exchange only positions was explained in Figure 5.7. Since *Position* node is still a valid node to crossover, the offspring can be created by exchanging only positions too.



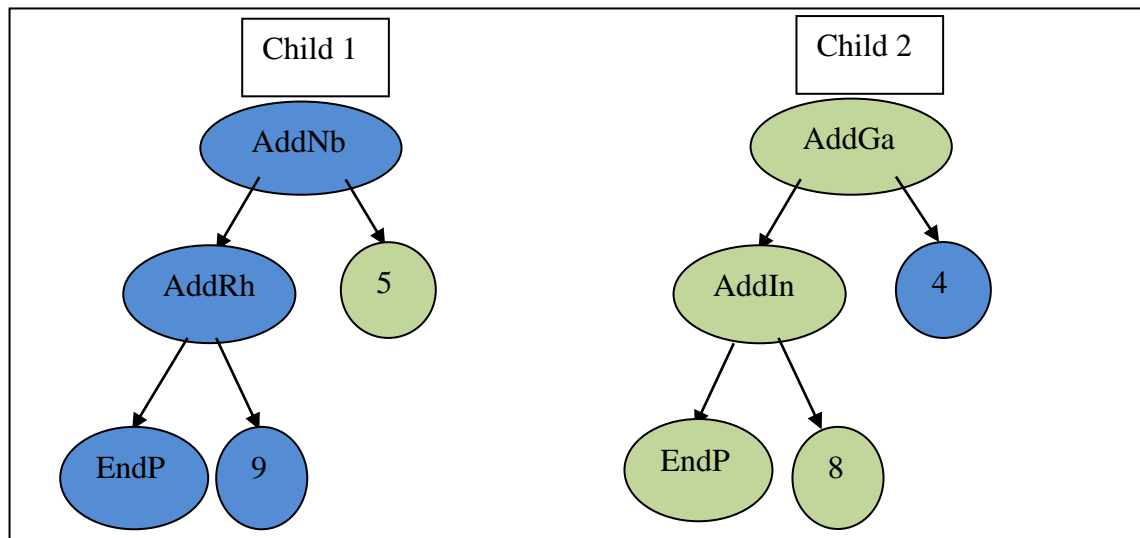
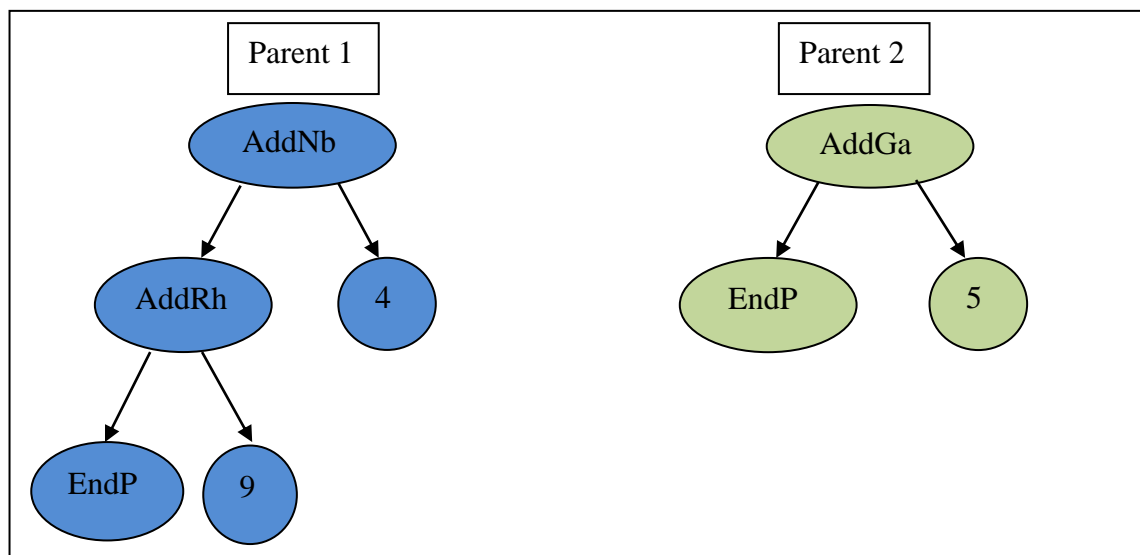


Figure 5.7: Crossover between Nb_Rh_4_9 and Ga_In_5_8. The created offspring are Nb_Rh_5_9 and Ga_In_4_8

If the selection operator selects an individual from single-dopant configuration and the other one from two-dopant configuration, the crossover operator chooses one sub-tree from two-dopant configuration and exchanges it with other parent (Figure 5.8).



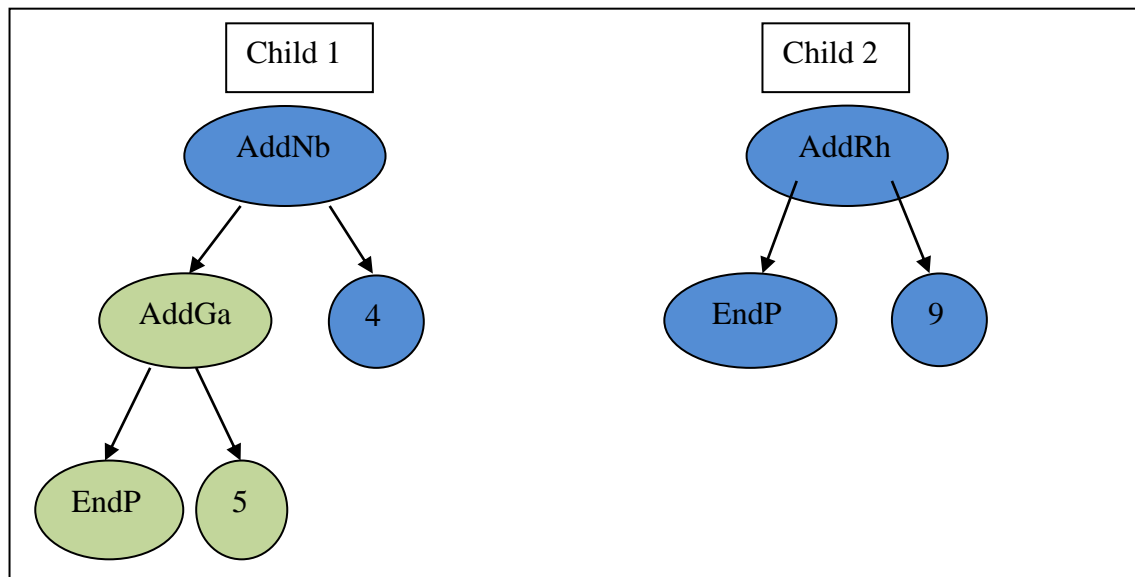
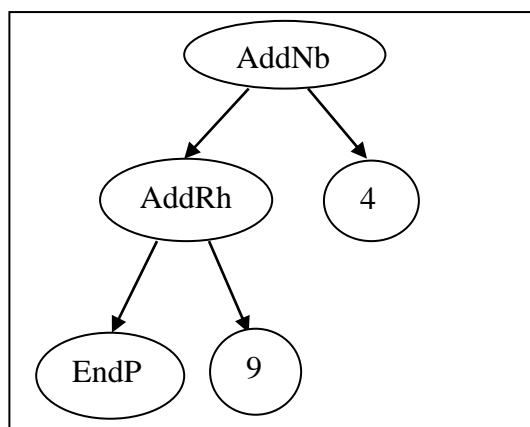


Figure 5.8: Crossover between a two-dopant configuration individual and single-dopant configuration individual

5.2.3 Mutation

Similar to the crossover operator, mutation can also be done in different ways. The idea is basically modifying an individual to get better individuals. This modification can be adding or removing a subtree or replacing the position node with another one.

Let the individual to mutate be Nb_Rh_4_9. Figure 5.9 shows how to replace a subtree with another one. This subtree can be a single-dopant configuration or can be a position only.



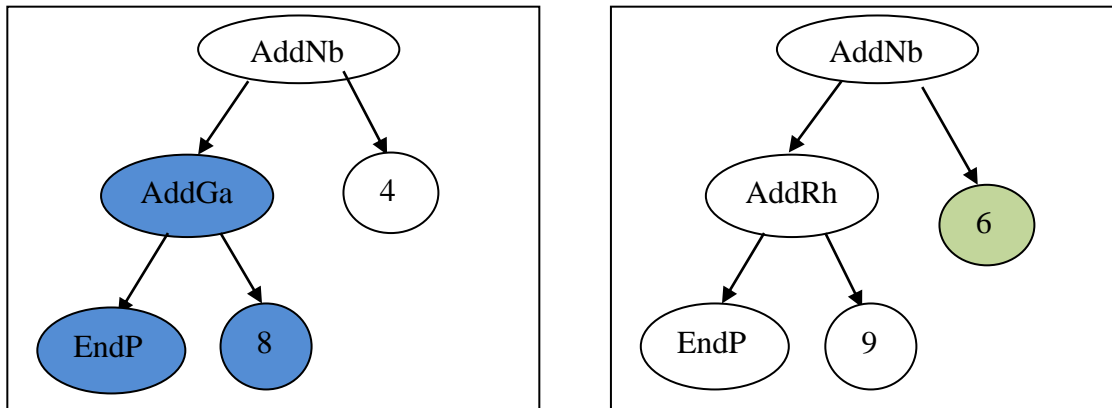


Figure 5.9: Two different mutation operator: a) Mutation was applied on a subtree which is a single-dopant configuration, b) Mutation was applied on Position node (is still a subtree).

Another possible mutation is to remove a subtree completely and replace it with an EndP node (Figure 5.10). Once again, Position node cannot be selected to replace with EndP since it is always a right subtree.

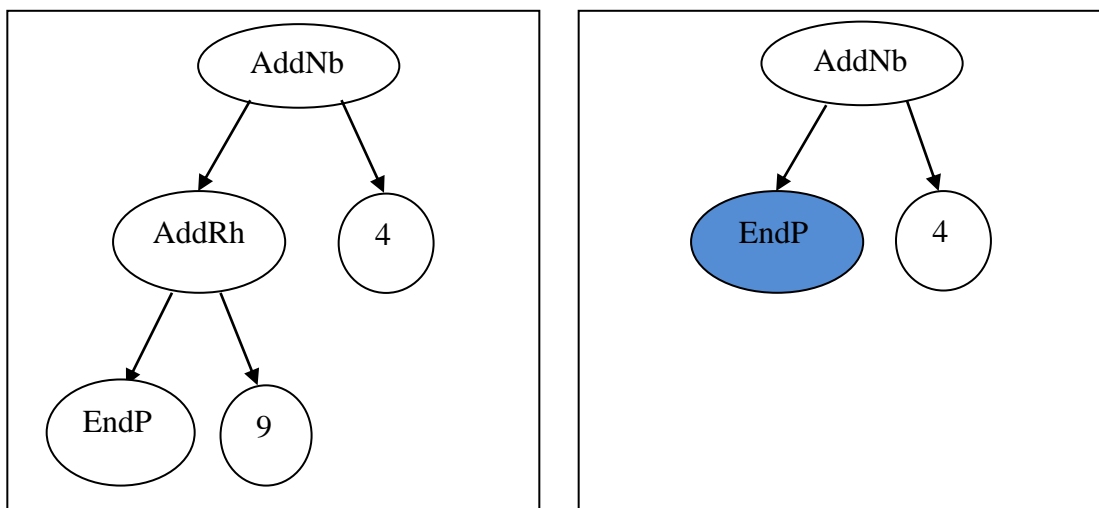


Figure 5.10: Mutation operator was applied on Nb_Rh_4_9 (a) and the new individual Nb_4 (b) was created

Similarly, mutation operator can add another subtree by replacing EndP with a valid subtree and can create a two-dopant configuration from a single-dopant configuration individual (Figure 5.11).

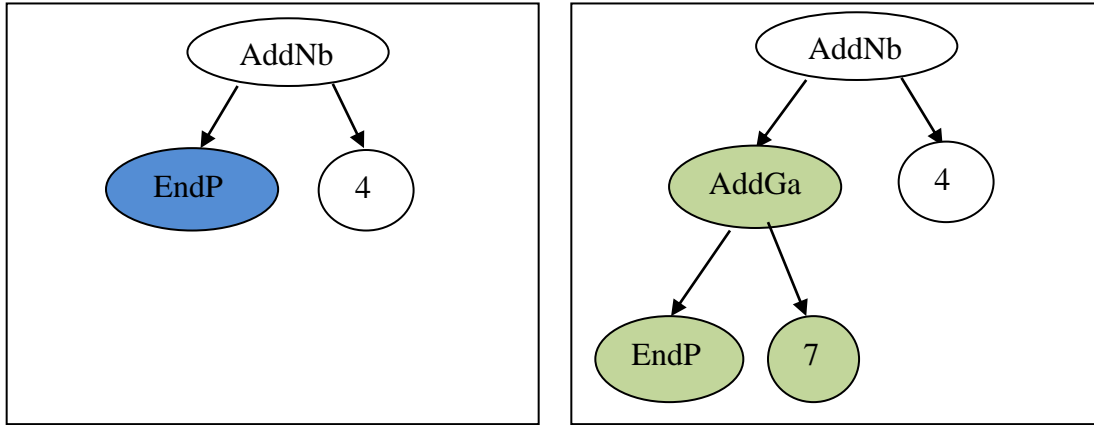


Figure 5.11: Mutation operator mutates the individual Nb_4 and creates the new individual Nb_Ga_4_7

5.2.3 Parameters

Open Beagle uses a configuration file XML format to define the parameters (Table 5.3). These parameters and more can be reached by running the program with extension `-OBUsage` or `-OBhelp` and set in *beagle.conf* file in desired way. The results can be both printed out the on screen and also be written in *beagle.log* file with more details. If a user wants to change the default parameters, this can be done by writing `<Entry key="ec.pop.size">40</Entry>` as shown in Table 5.3. This line overwrites the parameter for population size and set its new value to 40. The parameter “ec.repro.prob” is the reproduction probability of an individual and is used to reproduce an individual. The parameter “ec.hof.demesize” is the number of individuals kept in each deme’s hall-of-fame (best individuals so far). The termination criteria can be set in two ways: One is the, number of maximum generations and the parameter “ec.term.maxgen” is used to

determine it. The other one is the maximum fitness value with the parameter “ec.term.maxfitness”. If both of them are used, then the program stops when any of them is reached first. If the user’s optimization problem is minimization of the fitness value instead of maximization fitness, then this parameter can still be used when we convert the fitness values by multiplied the objective value by -1. The parameter “gp.cx.indpb” sets the crossover probability of an individual at each generation. The parameter “gp.mutshrink.indpb” is the shrink mutation probability describing how likely to replace a branch with one of its child node. In this case the selected node and its other children are removed. The parameter “gp.mutstd.indpb” is the standard mutation probability for an individual. A standard mutation replaces a sub tree of the individual with a randomly generated one. The parameter “gp.init.mindepth” is the minimum depth for newly created trees. In this implementation this parameter is set to 2 because each valid individual should have at least one atom and its position. The parameter “gp.init.maxdepth” is the number of maximum depth of initial trees and it is also set to 2 in our GP. The parameter “gp.tree.maxdepth” is the number of maximum depth of trees and it is set to 3 in our implementation. This is an important parameter because it is also used to stop the trees growing. Since in the doping example described above, there are only two substitution positions and corresponding elements. If the tree grows more than depth of 3, then the program will create individuals with more positions and elements, indicating that invalid individuals will be created. If a user wants to implement this program on 3 or more substitution positions, then this parameter needs to be set as the number of substitution positions.

Table 5.3: An example of the configuration file to run Genetic Programming on Open Beagle

```

<Beagle>
  <Evolver>
    <BootstrapSet>
      <GP-InitHalfConstrainedOp/>
      <SpambaseEvalOp/>
      <GP-StatsCalcFitnessSimpleOp/>
      <TermMaxGenOp/>
      <TermMaxFitnessOp fitness="1000.0"/>
      <MilestoneWriteOp/>
    </BootstrapSet>
    <MainLoopSet>
      <SelectTournamentOp/>
      <GP-CrossoverConstrainedOp/>
      <GP-MutationStandardConstrainedOp/>
      <GP-MutationShrinkConstrainedOp/>
      <GP-MutationSwapConstrainedOp/>
      <GP-MutationSwapSubtreeConstrainedOp/>
      <SpambaseEvalOp/>
      <GP-StatsCalcFitnessSimpleOp/>
      <TermMaxGenOp/>
      <TermMaxFitnessOp fitness="1000.0"/>
      <MilestoneWriteOp/>
    </MainLoopSet>
  </Evolver>
  <System>
    <Register>
      <Entry key="ec.pop.size">40</Entry>
      <Entry key="ec.repro.prob">0.0</Entry>
      <Entry key="ec.hof.demesize">5</Entry>
      <Entry key="ec.term.maxgen">20</Entry>
      <Entry key="gp.cx.indpb">0.8</Entry>
      <Entry key="gp.mutshrink.indpb">0.2</Entry>
      <Entry key="gp.mutstd.indpb">0.2</Entry>
      <Entry key="gp.init.mindepth">2</Entry>
      <Entry key="gp.init.maxdepth">2</Entry>
      <Entry key="gp.tree.maxdepth">3</Entry>
      <Entry key="lg.file.level">7</Entry>
    </Register>
  </System>
</Beagle>

```

5.2.4 Results

The main routine of the program consists of these 5 steps (Table 5.4): 1) Building primitives, 2) Building an evolutionary system, 3) Building evaluation, 4) Building an evolver and 5) Building a vivarium. In the primitive set, the only constrain is the Add_ functions and EndNode can only be placed in the left subtree and the node Position will always be in the right sub-tree.

Table 5.4: Pseudo-code of our GP implementation's main function

```
int main(int argc, char *argv[]) {
try {
// 1: Build primitive set
GP::PrimitiveSet::Handle lSet = new GP::PrimitiveSet;
lSet->insert(new AddNb);
lSet->insert(new AddRh);
lSet->insert(new AddIn);
lSet->insert(new AddAl);
lSet->insert(new AddGa);
lSet->insert(new EndNode);
lSet->insert(new Position);
// 2: Build a system
System::Handle lSystem = new System();
lSystem->addPackage(new GP::PackageConstrained(lSet));
// 3: Build evaluation operator
lSystem->setEvaluationOp("yourEvalOp" new "YourEvalOp"::Alloc;
// 4: Build evolver
Evolver::Handle lEvolver = new Evolver();
// 5: Build vivarium
Vivarium::Handle lVivarium = new Vivarium;
// 6: Initialize and evolve the vivarium
lEvolver->initialize(lSystem,argc,argv);
lEvolver->evolve(lVivarium, lSystem);
}
catch(Exception& inException) {
inException.terminate();
}
return 0;
}
```

Before the GP tested, some important parameters need to be optimized. Tournament selection size which determines how many tournaments will be occurred to select the parents for crossover is one of them. Different sizes have been tested and then we decided to set to 7. Figure 5.12 shows the different tournament sizes against best found individuals. For each size of tournament, the GP run 10 times and the ranks of the best found individuals were recorded and then compared with the known results. For example, on the first run the second best individual is found. On the second run the third best individual is found, on the third run the best individual is found, and so forth. Then the average values and standard deviation was calculated for each tournament size. Finally, it was determined to use tournament size 7, as it has the best average value and the lowest standard deviation found in this experiment. For instance, the average values of 7 and 5 are close to each other, however for tournament size 5, the GP may find the ninth or tenth best individual, and is why the standard deviation of tournament size 5 is higher than the tournament size 7. On the other hand, with tournament size 7, the GP finds one of the top 3 best results on all 10 independent runs.

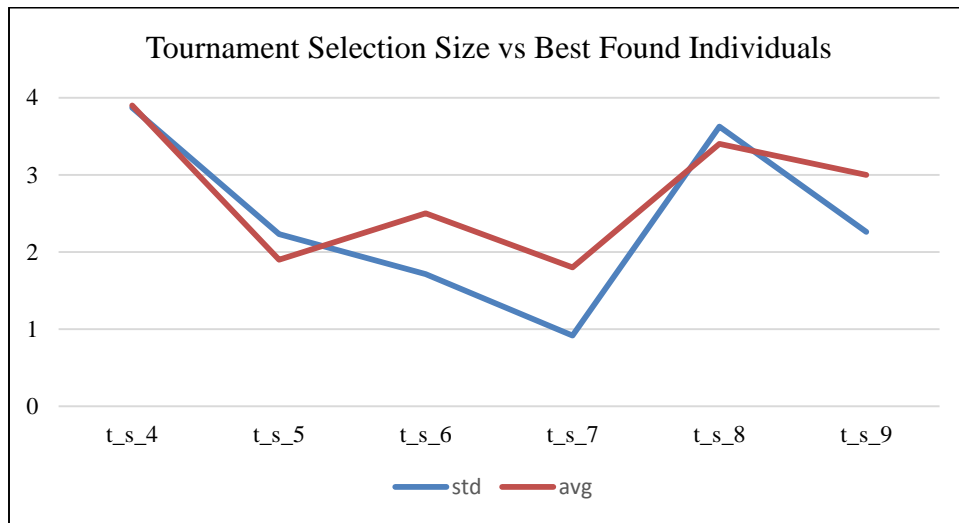


Figure 5.12: Different tournament sizes vs best found individuals

Population size and the number of generations are two critical parameters for all kind of genetic searches. In this work, since the search space is relatively small, there are not many options to vary these parameters. Also the total number of evaluations was set to around 400 to be comparable to Genetic Algorithms run in previous chapters. Generally, the total number of evaluations is equal to multiplication of population size and number of generations. However, in this work, GP may create invalid individuals. This can be an empty tree, or invalid number of nodes in the tree of which the fitness evaluation cannot be evaluated. Thus, the number of total evaluations is not exactly equal to the multiplication of population size and the number of generations. Instead, a counter is implemented inside the code to keep track of the total number of evaluations. In addition, since each different run may produce different number of total evaluations, ± 10 evaluations are ignored. Thus, if the total number of evaluations are set to 400, and if the program produces any number in the range [390-410], it was accepted as 400. Figure 5.13 shows that best found individuals with different number of population sizes and the number of generations. For each test case, GP run 10 times and best found individuals are compared to known results. For the case of population size 45 and the number of generations is 11, the GP finds one of top 3 results for each different runs.

Finally, the GP was tested with optimized parameters. The number total evaluations was set to around 410 (15% of the whole search space) to be comparable to Genetic Algorithms implemented in previous chapters. To check the robustness of the algorithm, 50 independent runs were executed. The best individuals were recorded and compared to the best known results. The top 3 best results were found for 43 runs out of

50. Figure 5.14 shows the results of 50 independent runs and which best individual was found at each run.

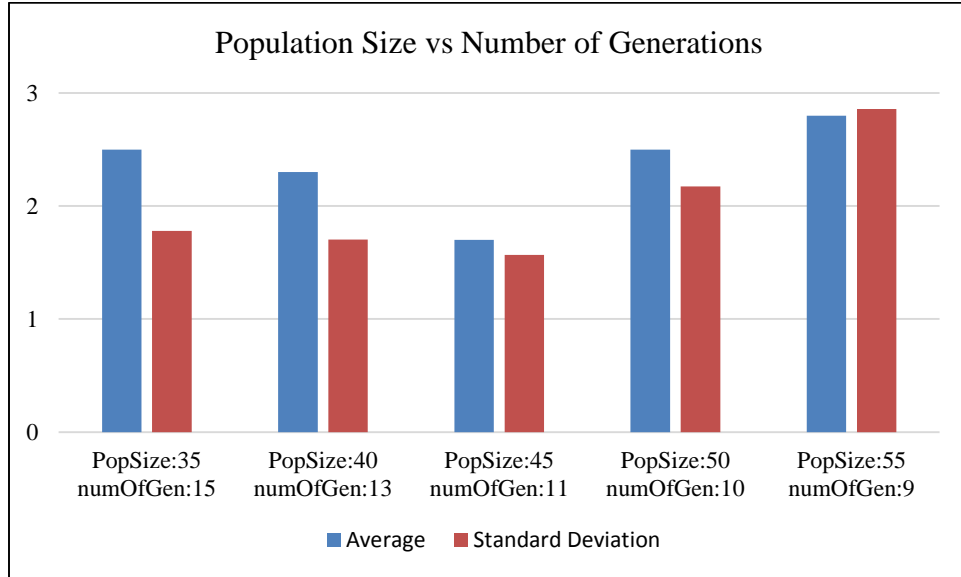


Figure 5.13: Best found individuals with different number of population sizes and the number of generations

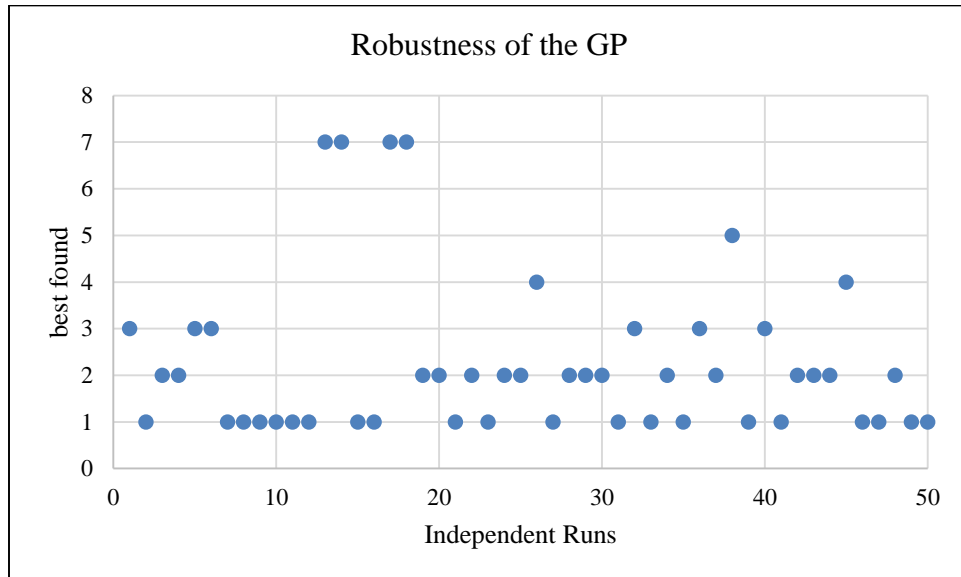


Figure 5.14: The result of 50 independent GP runs. Best found individuals are shown at each run. The top 3 best results were found at 43 runs out of 50

After testing GP, the crossover and mutation operators were also tested to see how they affect the generic search. First, the crossover operator was shut down and only mutation was allowed. The population size and the number of generations was set to proper numbers to get the similar total number of evaluations with GP for comparison. Then, the crossover operator was turned on and the mutation operator was shut down. Again, proper number of population size and the number of generations were chosen for fair comparison. Figure 5.15 shows the results of 50 runs of GP, GP with no crossover and GP with no mutation. Surprisingly, the results showed that mutation operator is more effective on genetic search than the crossover operator. It is generally not true. In this work, mutation looks more effective than crossover, because the crossover operator needs to be more carefully implemented and the GP tree depth limit put strong limitation on the search space capability of GP crossover operator. For larger search space, the contribution of crossover may be different, which is left for our future work.

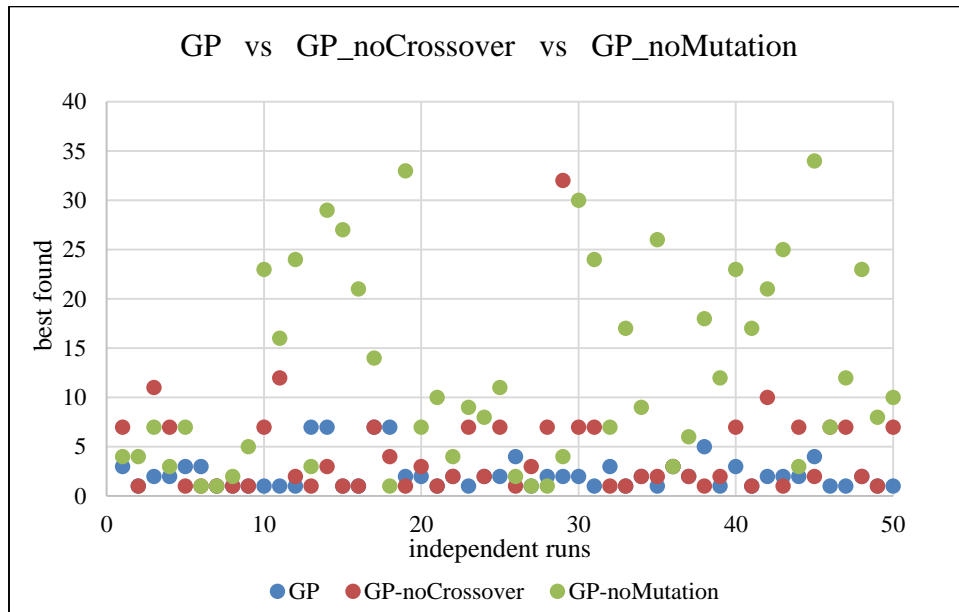


Figure 5.15: The results of 50 runs for different GP operators

5.3 CONCLUSIONS

We have successfully applied genetic programming for multi-element computational doping. Our experiments showed that GP is an effective algorithm for searching optimal doped materials in terms of free energy. It can save up to 80% of VASP simulations when doing VASP based simulation. Compared to genetic algorithm, the GP algorithm has the advantage of open-ended search space and may find better solutions for large doping problems.

CHAPTER 6

CONCLUSION

In this dissertation, an evolutionary framework for high-throughput computational doping were developed using genetic algorithms, genetic programming and Density Functional Theory (DFT) calculations. Although many DFT calculations have been applied to material research [53], [76], [126]–[134], a majority of them are used for measuring or verifying material performance [135]–[140]. Only some of them were applied for high-throughput material screening [95], [141], [142]. The ultimate goal of this dissertation is transforming current heuristic research practice in material research. More specifically,

- **We developed a genetic algorithm-based approach for identifying optimal doping position assignments for single and multiple dopant elements.**
- **We developed a genetic programming based search framework for computational doping that can screen materials with different doping elements and/or different substitution ratios.**

Evolutionary algorithms based computational doping framework will reduce the time for both experimental studies and computational studies. Fast computational screening will help to expand the material design space dramatically by evaluating novel

dopant elements with high speed. By working on the material of Solid Oxide Fuel Cell materials, this work may also help to speed up the SOFC material exploration processes. The genetic algorithms we implemented in Chapter 3 will help to find best doping configurations when only a single dopant element is doped in a base material. Our experiments showed that it can save up to 75% of VASP calculations by running the genetic algorithm to find the most stable structure compared to exhaustive search. In Chapter 4, we implemented another genetic algorithm which can work on multiple dopant elements. Since search space is larger in this case, the genetic algorithm brings even better benefits. With only 15% of total calculations, GA can get one of top 3 configurations out of 2625 different configurations. Finally, we implemented a Genetic Programming algorithm, which can work with different dopant elements and also different substitution rates. In this case, the search complexity increases dramatically when the number of elements increases and the allowed substitution rates increase. We tested our Genetic Programming with 5 dopant elements {Nb, Rh, Ga, Al, In} and 2 different substitution rates (15:1 and 15:2) on our base material SrTiO_3 . The results show that, by running only 15% of total calculations, one of the top 3 best configurations can be found by using GP.

Overall, this work will dramatically speed up the computational doping process, which is especially critical for large-scale computational screening.

REFERENCES

- [1] K. Kendall, S. C. Singhal, and K. Kendall, *High-temperature solid oxide fuel cells: fundamentals, design, and applications*. Elsevier Advanced Technology, 2003.
- [2] P. Wollmann, M. Leistner, U. Stoeck, R. Grönker, K. Gedrich, N. Klein, O. Throl, W. Grähler, I. Senkovska, F. Dreisbach, and S. Kaskel, "High-throughput screening: speeding up porous materials discovery.," *Chem. Commun. (Camb)*., vol. 47, no. 18, pp. 5151–3, May 2011.
- [3] J. Greeley, T. F. Jaramillo, J. Bonde, I. Chorkendorff, and J. K. Nørskov, "Computational high-throughput screening of electrocatalytic materials for hydrogen evolution.," *Nat. Mater.*, vol. 5, no. 11, pp. 909–13, Nov. 2006.
- [4] S. Curtarolo, D. Morgan, and G. Ceder, "Accuracy of ab initio methods in predicting the crystal structures of metals: A review of 80 binary alloys," *Calphad*, vol. 29, no. 3, pp. 163–211, Sep. 2005.
- [5] C. Ortiz, O. Eriksson, and M. Klintenberg, "Data mining and accelerated electronic structure theory as a tool in the search for new functional materials," *Comput. Mater. Sci.*, vol. 44, no. 4, pp. 1042–1049, Feb. 2009.
- [6] N. Q. Minh, "Solid oxide fuel cell technology--features and applications," *Solid State Ionics*, vol. 174, pp. 271–277, 2004.
- [7] W. . Zhu and S. . Deevi, "A review on the status of anode materials for solid oxide fuel cells," *Mater. Sci. Eng. A*, vol. 362, no. 1–2, pp. 228–239, Dec. 2003.
- [8] N. Q. Minh, "Ceramic Fuel Cells," *J. Am. Ceram. Soc.*, vol. 76, pp. 563–588, 1993.
- [9] N. Q. Minh and T. Takahashi, *Science and technology of ceramic fuel cells*. Elsevier, 1995.
- [10] S. C. Singhal, "Solid oxide fuel cells for stationary, mobile, and military applications," *Solid State Ionics*, vol. 152–153, pp. 405–410, Dec. 2002.
- [11] M. C. Williams, J. P. Strakey, W. A. Surdoyal, and L. C. Wilson, "Solid oxide fuel cell technology development in the U.S," *Solid State Ionics*, vol. 177, no. 19–25, pp. 2039–2044, Oct. 2006.
- [12] B. C. Steele and A. Heinzl, "Materials for fuel-cell technologies," *Nature*, vol. 414, no. 6861, pp. 345–52, Nov. 2001.
- [13] S. C. Singhal, "Advances in solid oxide fuel cell technology," *Solid State Ionics*, vol. 135, no. 1–4, pp. 305–313, Nov. 2000.
- [14] A. S. Nesaraj, "Recent developments in solid oxide fuel cell technology – a review," *Ind. Res.*, vol. 13, pp. 117–131, 2010.

- [15] J. Will, A. Mitterdorfer, C. Kleinlogel, D. Perednis, and L. J. Gauckler, "Fabrication of thin electrolytes for second-generation solid oxide fuel cells," *Solid State Ionics*, vol. 131, pp. 79–96, 2000.
- [16] A. J. Jacobson, "Materials for Solid Oxide Fuel Cells," *Chem. Mater.*, vol. 22, no. 3, pp. 660–674, Feb. 2010.
- [17] C. S. Kong and K. Rajan, "Rational design of binary halide scintillators via data mining," *Nucl. Instruments Methods Phys. Res. Sect. A Accel. Spectrometers, Detect. Assoc. Equip.*, vol. 680, pp. 145–154, Jul. 2012.
- [18] J. W. J. Davidson, D. A. Savic, and G. A. G. Walters, "Symbolic and numerical regression: Experiments and applications," *Inf. Sci. (Ny)*, vol. 150, no. 1, pp. 95–117, 2003.
- [19] M. Schmidt and H. Lipson, "Distilling free-form natural laws from experimental data.," *Science*, vol. 324, no. 5923, pp. 81–5, Apr. 2009.
- [20] G. Xiao, X. Dong, K. Huang, and F. Chen, "Synthesis and characterizations of A-site deficient perovskite $\text{Sr}_{0.9}\text{Ti}_{0.8-x}\text{GaxNb}_{0.2}\text{O}_3$," *Mater. Res. Bull.*, vol. 46, no. 1, pp. 57–61, Jan. 2011.
- [21] G. Xiao and F. Chen, "Ni modified ceramic anodes for direct-methane solid oxide fuel cells," *Electrochem. commun.*, vol. 13, no. 1, pp. 57–59, Jan. 2011.
- [22] Z. Yang, C. Yang, B. Xiong, M. Han, F. Chen, C. Jin, M. Han, and F. Chen, " $\text{Ba}_{0.9}\text{Co}_{0.7}\text{Fe}_{0.2}\text{Nb}_{0.1}\text{O}_{3-\delta}$ as cathode material for intermediate temperature solid oxide fuel cells," *Electrochem. commun.*, vol. 13, no. 22, pp. 882–885, Nov. 2011.
- [23] Y. Chen, F. Chen, W. Wang, D. Ding, and J. Gao, " $\text{Sm}_{0.2}(\text{Ce}_{1-x}\text{Ti}_x)_{0.8}\text{O}_{1.9}$ modified Ni–yttria-stabilized zirconia anode for direct methane fuel cell," *J. Power Sources*, vol. 196, no. 11, pp. 4987–4991, Jun. 2011.
- [24] X. Dong, S. Ma, K. Huang, and F. Chen, " $\text{La}_{0.9-x}\text{CaxCe}_{0.1}\text{CrO}_{3-\delta}$ as potential anode materials for solid oxide fuel cells," *Int. J. Hydrogen Energy*, vol. 37, no. 14, pp. 10866–10873, Jul. 2012.
- [25] G. Xiao, Q. Liu, X. Dong, K. Huang, and F. Chen, " $\text{Sr}_2\text{Fe}_{4/3}\text{Mo}_{2/3}\text{O}_6$ as anodes for solid oxide fuel cells," *J. Power Sources*, vol. 195, no. 24, pp. 8071–8074, Dec. 2010.
- [26] L. Zhang, Y. Liu, Y. Zhang, G. Xiao, F. Chen, and C. Xia, "Enhancement in surface exchange coefficient and electrochemical performance of $\text{Sr}_2\text{Fe}_{1.5}\text{Mo}_{0.5}\text{O}_6$ electrodes by $\text{Ce}_{0.8}\text{Sm}_{0.2}\text{O}_{1.9}$ nanoparticles," *Electrochem. commun.*, vol. 13, no. 7, pp. 711–713, Jul. 2011.
- [27] Q. Liu, D. E. Bugaris, G. Xiao, M. Chmara, S. Ma, H.-C. zur Loye, M. D. Amiridis, and F. Chen, " $\text{Sr}_2\text{Fe}_{1.5}\text{Mo}_{0.5}\text{O}_{6-\delta}$ as a regenerative anode for solid oxide fuel cells," *J. Power Sources*, vol. 196, no. 22, pp. 9148–9153, Nov. 2011.
- [28] C. Yang, C. Jin, and F. Chen, "Performances of micro-tubular solid oxide cell with novel asymmetric porous hydrogen electrode," *Electrochim. Acta*, vol. 56, no. 1, pp. 80–84, Dec. 2010.

- [29] C. Yang, C. Jin, A. Coffin, and F. Chen, "Characterization of infiltrated (La_{0.75}Sr_{0.25})_{0.95}MnO₃ as oxygen electrode for solid oxide electrolysis cells," *Int. J. Hydrogen Energy*, vol. 35, no. 11, pp. 5187–5193, Jun. 2010.
- [30] C. Yang, A. Coffin, and F. Chen, "High temperature solid oxide electrolysis cell employing porous structured (La_{0.75}Sr_{0.25})_{0.95}MnO₃ with enhanced oxygen electrode performance," *Int. J. Hydrogen Energy*, vol. 35, no. 8, pp. 3221–3226, Apr. 2010.
- [31] C. Yang, C. Jin, and F. Chen, "Micro-tubular solid oxide fuel cells fabricated by phase-inversion method," *Electrochem. commun.*, vol. 12, no. 5, pp. 657–660, May 2010.
- [32] Q. Liu, C. Yang, X. Dong, and F. Chen, "Perovskite Sr₂Fe_{1.5}Mo_{0.5}O_{6-δ} as electrode materials for symmetrical solid oxide electrolysis cells," *Int. J. Hydrogen Energy*, vol. 35, no. 19, pp. 10039–10044, Oct. 2010.
- [33] C. Jin, C. Yang, F. Zhao, A. Coffin, and F. Chen, "Direct-methane solid oxide fuel cells with Cu_{1.3}Mn_{1.7}O₄ spinel internal reforming layer," *Electrochem. commun.*, vol. 12, no. 10, pp. 1450–1452, Oct. 2010.
- [34] Z. Yang, C. Jin, C. Yang, M. Han, and F. Chen, "Ba_{0.9}Co_{0.5}Fe_{0.4}Nb_{0.1}O_{3-δ} as novel oxygen electrode for solid oxide electrolysis cells," *Int. J. Hydrogen Energy*, vol. 36, no. 18, pp. 11572–11577, Sep. 2011.
- [35] C. Jin, Z. Yang, H. Zheng, C. Yang, and F. Chen, "La_{0.6}Sr_{1.4}MnO₄ layered perovskite anode material for intermediate temperature solid oxide fuel cells," *Electrochem. commun.*, vol. 14, no. 1, pp. 75–77, Jan. 2012.
- [36] C. Jin, C. Yang, H. Zheng, and F. Chen, "Intermediate temperature solid oxide fuel cells with Cu_{1.3}Mn_{1.7}O₄ internal reforming layer," *J. Power Sources*, vol. 201, pp. 66–71, Mar. 2012.
- [37] G. Xiao, C. Jin, Q. Liu, A. Heyden, and F. Chen, "Ni modified ceramic anodes for solid oxide fuel cells," *J. Power Sources*, vol. 201, no. 3, pp. 43–48, Mar. 2012.
- [38] C. Yang, Z. Yang, C. Jin, G. Xiao, F. Chen, and M. Han, "Sulfur-tolerant redox-reversible anode material for direct hydrocarbon solid oxide fuel cells," *Adv. Mater.*, vol. 24, no. 11, pp. 1439–43, Mar. 2012.
- [39] L. Wu, Z. Jiang, S. Wang, and C. Xia, "(La,Sr)MnO₃–(Y,Bi)₂O₃ composite cathodes for intermediate-temperature solid oxide fuel cells," *Int. J. Hydrogen Energy*, vol. 38, no. 5, pp. 2398–2406, Feb. 2013.
- [40] B. Liu, W. Guo, F. Chen, and C. Xia, "Ga site doping and concentration variation effects on the conductivities of melilite-type lanthanum strontium gallate electrolytes," *Int. J. Hydrogen Energy*, vol. 37, no. 1, pp. 961–966, Jan. 2012.
- [41] Y. Wang, L. Zhang, F. Chen, and C. Xia, "Effects of doped ceria conductivity on the performance of La_{0.6}Sr_{0.4}Co_{0.2}Fe_{0.8}O_{3-δ} cathode for solid oxide fuel cell," *Int. J. Hydrogen Energy*, vol. 37, no. 10, pp. 8582–8591, May 2012.
- [42] J. W. Fergus, "Materials challenges for solid-oxide fuel cells," *Jom*, vol. 59, no. 12, pp. 56–62, 2007.

- [43] S. G. Kang and D. S. Sholl, "First principles assessment of perovskite dopants for proton conductors with chemical stability and high conductivity," *RSC Adv.*, vol. 3, no. 10, p. 3333, 2013.
- [44] Z. Li, B. Wei, Z. Lü, X. Zhu, X. Huang, Y. Zhang, Z. Guo, and W. Su, "Ba and Gd Doping Effect in $(\text{Ba}_x\text{Sr}_{1-x})_{0.95}\text{Gd}_{0.05}\text{Co}_{0.8}\text{Fe}_{0.2}\text{O}_{3-\delta}$ ($x = 0.1-0.9$) Cathode on the Phase Structure and Electrochemical Performance," *Fuel Cells*, vol. 12, no. 4, pp. 633–641, Aug. 2012.
- [45] J. Zou, J. Park, B. Kwak, H. Yoon, and J. Chung, "Effect of Fe doping on $\text{PrBaCo}_2\text{O}_{5+\delta}$ as cathode for intermediate-temperature solid oxide fuel cells," *Solid State Ionics*, vol. 206, pp. 112–119, Jan. 2012.
- [46] S.-F. Wang, H.-C. Lu, Y.-F. Hsu, C.-C. Huang, and C.-T. Yeh, "SrCo $_{1-x}$ Sb $_x$ O $_{3-\delta}$ cathode materials prepared by Pechini method for solid oxide fuel cell applications," *Ceram. Int.*, vol. 38, no. 7, pp. 5941–5947, Sep. 2012.
- [47] Z.-B. Yang, M.-F. Han, P. Zhu, F. Zhao, and F. Chen, "Ba $_{1-x}$ Co $_{0.9-y}$ FeyNb $_{0.1}$ O $_{3-\delta}$ ($x = 0-0.15$, $y = 0-0.9$) as cathode materials for solid oxide fuel cells," *Int. J. Hydrogen Energy*, vol. 36, no. 15, pp. 9162–9168, Jul. 2011.
- [48] H. Gu, H. Chen, L. Gao, Y. Zheng, X. Zhu, and L. Guo, "Effect of Co doping on the properties of Sr $_{0.8}$ Ce $_{0.2}$ MnO $_{3-\delta}$ cathode for intermediate-temperature solid-oxide fuel cells," *Int. J. Hydrogen Energy*, vol. 33, no. 17, pp. 4681–4688, Sep. 2008.
- [49] X. Li, H. Zhao, F. Gao, N. Chen, and N. Xu, "La and Sc co-doped SrTiO $_3$ as novel anode materials for solid oxide fuel cells," *Electrochem. commun.*, vol. 10, no. 10, pp. 1567–1570, Oct. 2008.
- [50] Z. Xie, H. Zhao, Z. Du, and T. Chen, "Effects of Co Doping on the Electrochemical Performance of Double Perovskite Oxide Sr $_2$ MgMoO $_{6-\delta}$ as an Anode Material for Solid Oxide Fuel Cells," *J. ...*, vol. 116, no. 17, pp. 9734–9743, 2012.
- [51] J. S. Yoon, M. Y. Yoon, C. Kwak, H. J. Park, S. M. Lee, K. H. Lee, and H. J. Hwang, "Y $_{0.08}$ Sr $_{0.92}$ FexTi $_{1-x}$ O $_{3-\delta}$ perovskite for solid oxide fuel cell anodes," *Mater. Sci. Eng. B*, vol. 177, no. 2, pp. 151–156, Feb. 2012.
- [52] F. Zhao, S. Wang, L. Dixon, and F. Chen, "Novel BaCe $_{0.7}$ In $_{0.2}$ Yb $_{0.1}$ O $_{3-\delta}$ proton conductor as electrolyte for intermediate temperature solid oxide fuel cells," *J. Power Sources*, vol. 196, no. 18, pp. 7500–7504, Sep. 2011.
- [53] S. Suthirakun, S. C. Ammal, G. Xiao, F. Chen, H.-C. Zur Loye, and A. Heyden, "Density functional theory study on the electronic structure of n-and p-type doped SrTiO $_3$ at anodic solid oxide fuel cell conditions," *Phys. Rev. B*, vol. 84, no. 20, p. 205102, 2011.
- [54] L. H. Gao, Z. Ma, and Q. B. Fan, "First-principle studies of the electronic structure and reflectivity of LaTiO $_3$ and Sr doped LaTiO $_3$ (La $_{1-x}$ Sr $_x$ TiO $_3$)," *J. Electroceramics*, vol. 27, no. 3–4, pp. 114–119, Sep. 2011.
- [55] a J. Du, S. C. Smith, X. D. Yao, and G. Q. Lu, "Hydrogen spillover mechanism on

a Pd-doped Mg surface as revealed by ab initio density functional calculation.,” *J. Am. Chem. Soc.*, vol. 129, no. 33, pp. 10201–4, Aug. 2007.

- [56] H. Kamisaka, T. Suenaga, H. Nakamura, and K. Yamashita, “DFT-Based Theoretical Calculations of Nb-and W-Doped Anatase TiO₂: Complex Formation between W Dopants and Oxygen Vacancies,” *J. Phys. Chem. C*, vol. 114, pp. 12777–12783, 2010.
- [57] R. Rurali, E. Hernández, P. Godignon, J. Rebollo, and P. Ordejón, “First-Principles Studies of N and P Dopant Interactions in SiC: Implications for Co-Doping,” *Mater. Sci. Forum*, vol. 433–436, pp. 649–652, 2003.
- [58] A. Ghazali and P. L. Hugon, “Density-Functional Approach to the Metal-Insulator Transition in Doped Semiconductors,” *Phys. Rev. Lett.*, vol. 41, no. 22, pp. 1569–1572, 1978.
- [59] J. D. Baniecki, M. Ishii, H. Aso, K. Kurihara, and D. Ricinschi, “Density functional theory and experimental study of the electronic structure and transport properties of La, V, Nb, and Ta doped SrTiO₃,” *J. Appl. Phys.*, vol. 113, no. 1, p. 013701, 2013.
- [60] F. Wang, C. Di Valentin, and G. Pacchioni, “Doping of WO₃ for Photocatalytic Water Splitting: Hints from Density Functional Theory,” *J. Phys. Chem. C*, vol. 116, pp. 8901–8909, 2012.
- [61] X. D. Zhang, M. L. Guo, Y. Y. Shen, C. L. Liu, Y. H. Xue, F. Zhu, and L. H. Zhang, “Electronic structure and optical transition in heavy metal doped ZnO by first-principle calculations,” *Comput. Mater. Sci.*, vol. 54, pp. 75–80, Mar. 2012.
- [62] R. Han, W. Yuan, H. Yang, X. Du, Y. Yan, and H. Jin, “Possible ferromagnetism in Li, Na and K-doped AlN: A first-principle study,” *J. Magn. Magn. Mater.*, vol. 326, pp. 45–49, Jan. 2013.
- [63] H. Yoshida, “Density functional theory calculation on the effect of local structure of doped ceria on ionic conductivity,” *Solid State Ionics*, vol. 160, no. 1–2, pp. 109–116, May 2003.
- [64] S. Liu, J. Li, Y. Zhang, X. Xu, and Z. Chen, “Density functional theory study on electronic structure of N-doped In₂O₃,” *J. Mol. Struct. THEOCHEM*, vol. 866, no. 1–3, pp. 75–78, Oct. 2008.
- [65] X. Jia, W. Yang, M. Qin, and J. Li, “Structure and magnetism in Mn-doped zirconia: Density-functional theory studies,” *J. Magn. Magn. Mater.*, vol. 321, no. 15, pp. 2354–2358, Aug. 2009.
- [66] Z. Zhou, X. Gao, J. Yan, and D. Song, “Doping effects of B and N on hydrogen adsorption in single-walled carbon nanotubes through density functional calculations,” *Carbon N. Y.*, vol. 44, no. 5, pp. 939–947, Apr. 2006.
- [67] M. Nolan and G. W. Watson, “The electronic structure of alkali doped alkaline earth metal oxides: Li doping of MgO studied with DFT-GGA and GGA+U,” *Surf. Sci.*, vol. 586, no. 1–3, pp. 25–37, Jul. 2005.
- [68] C. Alemán, D. Curcó, and J. Casanovas, “A density functional theory study of n-doped 3,4-ethylenedioxythiophene oligomers,” *Chem. Phys. Lett.*, vol. 386, no. 4–

- 6, pp. 408–413, Mar. 2004.
- [69] Y. Dai, S. Han, B. Huang, and D. Dai, “Study on n-type doping with phosphorous in diamond by means of density functional theory,” *Mater. Sci. Eng. B*, vol. 99, no. 1–3, pp. 531–535, May 2003.
- [70] G. Hautier, A. Jain, and S. P. Ong, “From the computer to the laboratory: materials discovery and design using first-principles calculations,” *J. Mater. Sci.*, vol. 47, no. 21, pp. 7317–7340, May 2012.
- [71] X. G. Guo, X. S. Chen, Y. L. Sun, L. Z. Sun, X. H. Zhou, and W. Lu, “Electronic band structure of Nb doped SrTiO₃ from first principles calculation,” *Phys. Lett. A*, vol. 317, no. 5, pp. 501–506, 2003.
- [72] F. Maldonado, R. Rivera, and A. Stashans, “Structure, electronic and magnetic properties of Ca-doped chromium oxide studied by the DFT method,” *Phys. B Condens. Matter*, vol. 407, no. 8, pp. 1262–1267, Apr. 2012.
- [73] M. Li, J.-Y. Zhang, Y. Zhang, and T.-M. Wang, “Oxygen vacancy in N-doped Cu₂O crystals: A density functional theory study,” *Chinese Phys. B*, vol. 21, no. 8, p. 87301, 2012.
- [74] J. Zhang, E.-J. Liang, Q. Sun, and Y. Jia, “Oxygen vacancy formation and migration in Sr- and Mg-doped LaGaO₃: a density functional theory study,” *Chinese Phys. B*, vol. 21, no. 4, p. 47201, 2012.
- [75] M. Shishkin and T. Ziegler, “The Electronic Structure and Chemical Properties of a Ni/CeO₂ Anode in a Solid Oxide Fuel Cell: A DFT+ U Study,” *J. Phys. Chem. C*, vol. 114, pp. 21411–21416, 2010.
- [76] S. Suthirakun, S. C. Ammal, G. Xiao, F. Chen, K. Huang, H.-C. zur Loye, and A. Heyden, “Obtaining mixed ionic/electronic conductivity in perovskite oxides in a reducing environment: A computational prediction for doped SrTiO₃,” *Solid State Ionics*, vol. 228, pp. 37–45, Nov. 2012.
- [77] X. Chen, S. Sun, X. Wang, F. Li, and D. Xia, “DFT Study of Polyaniline and Metal Composites as Nonprecious Metal Catalysts for Oxygen Reduction in Fuel Cells,” *J. Phys. Chem. C*, vol. 116, no. 43, pp. 22737–22742, Nov. 2012.
- [78] A. Muñoz-García and D. Bugaris, “Unveiling Structure–Property Relationships in Sr₂Fe_{1.5}Mo_{0.5}O_{6–δ}, an Electrode Material for Symmetric Solid Oxide Fuel Cells,” *J. ...*, vol. 134, pp. 6826–6833, 2012.
- [79] H.-T. Chen, Y. Choi, M. Liu, and M. C. Lin, “A first-principles analysis for sulfur tolerance of CeO₂ in solid oxide fuel cells,” *J. Phys. ...*, vol. 2, no. 111, pp. 11117–11122, 2007.
- [80] Y. Suga, “Density Functional Study of Catalytic Oxygen Reduction Reaction (ORR) on Cathode Electrode for Fuel Cell System,” *Chem. Lett.*, vol. 35, no. 12, pp. 1406–1407, 2006.
- [81] J. Zhou, G. Chen, K. Wu, Y. Cheng, B. Peng, J. Guo, and Y. Jiang, “Density functional theory study on oxygen adsorption in LaSrCoO₄: An extended cathode material for solid oxide fuel cells,” *Appl. Surf. Sci.*, vol. 258, no. 7, pp. 3133–3138, Jan. 2012.

- [82] H.-T. Chen, P. Raghunath, and M. C. Lin, "Computational investigation of O₂ reduction and diffusion on 25% Sr-doped LaMnO₃ cathodes in solid oxide fuel cells," *Langmuir*, vol. 27, no. 11, pp. 6787–93, Jun. 2011.
- [83] E. a. Ahmad, L. Liborio, D. Kramer, G. Mallia, a. R. Kucernak, and N. M. Harrison, "Thermodynamic stability of LaMnO₃ and its competing oxides: A hybrid density functional study of an alkaline fuel cell catalyst," *Phys. Rev. B*, vol. 84, no. 8, p. 085137, Aug. 2011.
- [84] W. An, D. Gatewood, B. Dunlap, and C. H. Turner, "Catalytic activity of bimetallic nickel alloys for solid-oxide fuel cell anode reactions from density-functional theory," *J. Power Sources*, vol. 196, no. 10, pp. 4724–4728, May 2011.
- [85] E. W. Bucholz, C. S. Kong, K. R. Marchman, W. G. Sawyer, S. R. Phillpot, S. B. Sinnott, and K. Rajan, "Data-Driven Model for Estimation of Friction Coefficient Via Informatics Methods," *Tribol. Lett.*, vol. 47, no. 2, pp. 211–221, May 2012.
- [86] K. Rajan, "Materials informatics part I: A diversity of issues," *JOM J. Miner. Met. Mater. ...*, no. March, p. 50, 2008.
- [87] K. Rajan, "Informatics and integrated computational materials engineering: Part II," *JOM J. Miner. Met. Mater. ...*, vol. 61, no. 1, p. 47, 2009.
- [88] L. M. Bartolo, S. C. Glotzer, C. S. Lowe, A. C. Powel, D. R. Sadoway, J. A. Warren, V. K. Tewary, M. J. M. Krane, and K. Rajan, "Materials informatics: Facilitating the integration of data-driven materials research with education," *JOM J. Miner. Met. Mater. ...*, vol. 60, no. March, pp. 51–52, 2008.
- [89] S. Broderick, C. Suh, J. Nowers, B. Vogel, S. Mallapragada, B. Narasimhan, and K. Rajan, "Informatics for combinatorial materials science," *JOM J. Miner. Met. Mater. ...*, no. March, pp. 56–59, 2008.
- [90] K. Rajan, "Learning from systems biology: an 'omics' approach to materials design," *JOM J. Miner. Met. Mater. ...*, no. March, pp. 53–55, 2008.
- [91] A. Jain, G. Hautier, C. J. Moore, S. Ping Ong, C. C. Fischer, T. Mueller, K. A. Persson, and G. Ceder, "A high-throughput infrastructure for density functional theory calculations," *Comput. Mater. Sci.*, vol. 50, no. 8, pp. 2295–2310, Jun. 2011.
- [92] G. Hautier, A. Jain, H. Chen, C. Moore, S. P. Ong, and G. Ceder, "Novel mixed polyanions lithium-ion battery cathode materials predicted by high-throughput ab initio computations," *J. Mater. Chem.*, vol. 21, no. 43, p. 17147, 2011.
- [93] G. Ceder, "Opportunities and challenges for first-principles materials design and applications to Li battery materials," *MRS Bull.*, vol. 35, no. September, pp. 693–702, 2010.
- [94] A. Jain, S.-A. Seyed-Reihani, C. C. Fischer, D. J. Couling, G. Ceder, and W. H. Green, "Ab initio screening of metal sorbents for elemental mercury capture in syngas streams," *Chem. Eng. Sci.*, vol. 65, no. 10, pp. 3025–3033, May 2010.
- [95] Y. Wu, P. Lazic, G. Hautier, K. Persson, and G. Ceder, "First principles high throughput screening of oxynitrides for water-splitting photocatalysts," *Energy Environ. Sci.*, vol. 6, no. 1, p. 157, 2013.

- [96] D. P. Stucke and V. H. Crespi, "Predictions of New Crystalline States for Assemblies of Nanoparticles: Perovskite Analogues and 3-D Arrays of Self-Assembled Nanowires," *Nano Lett.*, vol. 3, no. 9, pp. 1183–1186, Sep. 2003.
- [97] R. Ramprasad, V. Kumar, L. R. C. Fonseca, and B. R. Tuttle, "Recent advances in first principles computations in materials research," *J. Mater. Sci.*, vol. 47, no. 21, pp. 7313–7316, Jul. 2012.
- [98] S. Curtarolo, W. Setyawan, G. L. W. Hart, M. Jahnatek, R. V. Chepulskii, R. H. Taylor, S. Wang, J. Xue, K. Yang, O. Levy, M. J. Mehl, H. T. Stokes, D. O. Demchenko, and D. Morgan, "AFLOW: An automatic framework for high-throughput materials discovery," *Comput. Mater. Sci.*, vol. 58, pp. 218–226, Jun. 2012.
- [99] S. Curtarolo, W. Setyawan, S. Wang, J. Xue, K. Yang, R. H. Taylor, L. J. Nelson, G. L. W. Hart, S. Sanvito, M. Buongiorno-Nardelli, N. Mingo, and O. Levy, "AFLOWLIB.ORG: A distributed materials properties repository from high-throughput ab initio calculations," *Comput. Mater. Sci.*, vol. 58, pp. 227–235, Jun. 2012.
- [100] W. Setyawan and S. Curtarolo, "High-throughput electronic band structure calculations: Challenges and tools," *Comput. Mater. Sci.*, vol. 49, no. 2, pp. 299–312, Aug. 2010.
- [101] P. Hohenberg and W. Kohn, "Inhomogeneous Electron Gas," *Phys. Rev.*, vol. 155, no. 1962, pp. 864–871, 1964.
- [102] J. H. Holland, *Adaptation in Natural and Artificial Systems: An introductory analysis with applications to biology, control, and artificial intelligence*. 1975.
- [103] J. R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection (Complex Adaptive Systems)*. MA: The MIT Press, 1992.
- [104] J. R. Koza, M. A. Keane, J. Yu, W. Mydlowec, and F. H. Bennett III, "Automatic synthesis of both the topology and parameters for a controller for a three-lag plant with a five-second delay using genetic programming," *Real-World Appl. Evol. Comput.*, no. Springer Berling Heidelberg, pp. 168–177, 2000.
- [105] J. D. Perkins, T. R. Paudel, A. Zakutayev, P. F. Ndione, P. A. Parilla, D. L. Young, S. Lany, D. S. Ginley, A. Zunger, N. H. Perry, Y. Tang, M. Grayson, T. O. Mason, J. S. Bettinger, Y. Shi, and M. F. Toney, "Inverse design approach to hole doping in ternary oxides: Enhancing p-type conductivity in cobalt oxide spinels," *Phys. Rev. B*, vol. 84, no. 20, p. 205207, Nov. 2011.
- [106] H. C. Wang, C. L. Wang, W. Bin Su, J. Liu, Y. Sun, H. Peng, and L. M. Mei, "Doping effect of La and Dy on the thermoelectric properties of SrTiO₃," *J. Am. Ceram. Soc.*, vol. 94, no. 3, pp. 838–842, 2011.
- [107] A. J. Heeger, "Semiconducting and Metallic Polymers: The Fourth Generation of Polymeric Materials (Nobel Lecture)," *Rev. Mod. Phys.*, vol. 73, no. 3, pp. 681–700, 2001.
- [108] G. Kresse and J. Furthmüller, "Efficient iterative schemes for ab initio total-energy calculations using a plane-wave basis set," *Phys. Rev. B*, vol. 54, no. 16, pp.

11169–11186, 1996.

- [109] A. Jain, S. P. Ong, G. Hautier, W. Chen, W. D. Richards, S. Dacek, S. Cholia, D. Gunter, D. Skinner, G. Ceder, and K. A. Persson, “Commentary: The materials project: A materials genome approach to accelerating materials innovation,” *APL Materials*, vol. 1, no. 1. American Institute of Physics Inc., 2013.
- [110] J. P. Perdew, K. Burke, and M. Ernzerhof, “Generalized Gradient Approximation Made Simple,” *Phys. Rev. Lett.*, vol. 77, no. 18, pp. 3865–3868, Oct. 1996.
- [111] P. E. Blöchl, “Projector augmented-wave method,” *Phys. Rev. B*, vol. 50, no. 24, pp. 17953–17979, 1994.
- [112] S. Hui and A. Petric, “Electrical conductivity of yttrium-doped SrTiO₃: Influence of transition metal additives,” *Mater. Res. Bull.*, vol. 37, no. 7, pp. 1215–1231, 2002.
- [113] J. R. Koza, “Genetic programming as a means for programming computers by natural selection,” *Stat. Comput.*, vol. 4, no. 2, pp. 87–112, 1994.
- [114] C. Gagné and M. Parizeau, “Open BEAGLE: A New Versatile C++ Framework for Evolutionary Computation,” in *GECCO Late Breaking Papers*, 2002, pp. 161–168.
- [115] D. Batenkov, “Open BEAGLE: a generic framework for evolutionary computations,” *Genet. Program. Evolvable Mach.*, vol. 12, no. 3, pp. 329–331, Mar. 2011.
- [116] J. R. Koza, “Human-competitive results produced by genetic programming,” *Genet. Program. Evolvable Mach.*, vol. 11, no. 3–4, pp. 251–284, May 2010.
- [117] J. R. Koza, S. H. Al-Sakran, and L. W. Jones, “Automated re-invention of six patented optical lens systems using genetic programming,” *Proc. 2005 Conf. Genet. Evol. Comput. - GECCO '05*, p. 1953, 2005.
- [118] F. H. Bennett III, J. R. Koza, J. Yu, and W. Mydlowec, “Automatic synthesis, placement, and routing of an amplifier circuit by means of genetic programming,” *Evolvable Syst. From Biol. to Hardw.*, no. Springer Berlin Heidelberg, pp. 1–10, 2000.
- [119] D. Montana, “Strongly typed genetic programming,” *Evol. Comput.*, pp. 1–34, 1995.
- [120] J. Branke, K. Deb, H. Dierolf, and M. Osswald, “Finding knees in multi-objective optimization,” 2004.
- [121] K. Deb and S. Chaudhuri, “I-EMO: An interactive evolutionary multi-objective optimization tool,” Kanpur, India, 2005.
- [122] M. a. Abido, “Environmental/economic power dispatch using multiobjective evolutionary algorithms,” *IEEE Trans. Power Syst.*, vol. 18, no. 4, pp. 1529–1537, Nov. 2003.
- [123] N. Padhye and K. Deb, “Multi-objective optimisation and multi-criteria decision making in SLS using evolutionary approaches,” *Rapid Prototyp. J.*, pp. 1–30, 2011.

- [124] J. D. Gale, "GULP: Capabilities and prospects," *Zeitschrift für Krist.*, vol. 220, no. 5/6/2005, pp. 552–554, Jan. 2005.
- [125] J. J. D. Gale, "GULP: A computer program for the symmetry-adapted simulation of solids," *J. Chem. Soc. Faraday Trans.*, vol. 93, no. 4, pp. 629–637, 1997.
- [126] L. Wang, L. Zhang, J. Li, J. Gao, C. Jiang, and X. He, "First-Principles Study of Doping in LiMnPO₄," *Int. J. Electrochem. Sci.*, vol. 7, pp. 3362–3370, 2012.
- [127] C. Zhang, Y. Jia, Y. Jing, Y. Yao, J. Ma, and J. Sun, "DFT study on electronic structure and optical properties of N-doped, S-doped, and N/S co-doped SrTiO₃," *Phys. B Condens. Matter*, vol. 407, no. 24, pp. 4649–4654, Dec. 2012.
- [128] S. Pesant and M. Côté, "DFT + U study of magnetic order in doped La₂CuO₄ crystals," *Phys. Rev. B*, vol. 84, no. 8, p. 085104, Aug. 2011.
- [129] J.-N. YUN and Z.-Y. ZHANG, "Effect of in and Sc Doping on the Electronic Structure and Optical Properties of SrTiO₃," *Acta Physico-Chimica Sin.*, vol. 26, no. 3, pp. 751–757, 2010.
- [130] B. Li and H. Metiu, "DFT studies of oxygen vacancies on undoped and doped La₂O₃ surfaces," *J. Phys. Chem. C*, vol. 114, pp. 12234–12244, 2010.
- [131] A. Walsh, Y. Yan, M. M. Al-Jassim, and S.-H. Wei, "Electronic, energetic, and chemical effects of intrinsic defects and Fe-doping of CoAl₂O₄: a DFT+ U study," *J. Phys. Chem. C*, vol. 112, pp. 12044–12050, 2008.
- [132] L. Benco, J. Hafner, Z. Lences, and P. Sajgalik, "Density functional study of structures and mechanical properties of Y-doped α -SiAlONs," *J. Eur. Ceram. Soc.*, vol. 28, no. 5, pp. 995–1002, Jan. 2008.
- [133] L. Liu, K. Bai, H. Gong, and P. Wu, "First-principles study of Sn and Ca doping in CuInO₂," *Phys. Rev. B*, vol. 72, no. 12, p. 125204, Sep. 2005.
- [134] I. I. Tupitsyn, a. Deineka, V. Trepakov, L. Jastrabik, and S. Kapphann, "Li-doping effect on the energy structure of KTaO₃," *Ferroelectrics*, vol. 237, no. 1, pp. 9–16, Jan. 2000.
- [135] K. C. Kim, S. G. Kang, and D. S. Sholl, "Predictions of Sulfur Resistance in Metal Membranes for H₂ Purification Using First-Principles Calculations," *Ind. Eng. Chem. ...*, vol. 51, pp. 301–309, 2012.
- [136] K. K. C. Kim and D. D. S. Sholl, "Crystal structures and thermodynamic investigations of LiK(BH₄)₂, KBH₄, and NaBH₄ from first-principles calculations," *J. Phys. Chem. C*, vol. 114, pp. 678–686, 2010.
- [137] R. B. Rankin, S. Hao, D. S. Sholl, and J. K. Johnson, "DFT characterization of adsorption and diffusion mechanisms of H, As, S, and Se on the zinc orthotitanate(010) surface," *Surf. Sci.*, vol. 602, no. 10, pp. 1877–1882, May 2008.
- [138] B. Dai, D. S. Sholl, and J. K. Johnson, "First-principles investigation of adsorption and dissociation of hydrogen on Mg₂Si surfaces," *J. Phys. Chem. ...*, vol. 111, pp. 6910–6916, 2007.
- [139] A. Asthagiri and D. S. Sholl, "DFT study of Pt adsorption on low index SrTiO₃ surfaces: SrTiO₃(100), SrTiO₃(111) and SrTiO₃(110)," *Surf. Sci.*, vol. 581, no. 1,

pp. 66–87, Apr. 2005.

- [140] A. Asthagiri and D. S. Sholl, “First principles study of Pt adhesion and growth on SrO- and TiO₂-terminated SrTiO₃(100),” *J. Chem. Phys.*, vol. 116, no. 22, p. 9914, 2002.
- [141] R. Armiento, B. Kozinsky, M. Fornari, and G. Ceder, “Screening for high-performance piezoelectrics using high-throughput density functional theory,” *Phys. Rev. B*, vol. 84, no. 1, p. 014103, Jul. 2011.
- [142] T. Watanabe and D. S. Sholl, “Accelerating applications of metal-organic frameworks for gas adsorption and separation by computational screening of materials,” *Langmuir*, vol. 28, no. 40, pp. 14114–28, Oct. 2012.