Development of a Monte Carlo Algorithm for the Prediction of Acid Site Distribution in Zeolites

by

Shu Ning Hiew

A thesis submitted to

the Department of Chemical and Biomolecular Engineering,

Cullen College of Engineering

Bachelor of Science

in Chemical Engineering

Chair of Committee: Dr. Lars Grabow

Committee Member: Dr. Jeremy Palmer

Committee Member: Dr. Fritz Claydon

University of Houston

May 2020

ACKNOWLEDGEMENTS

First and foremost, I really appreciate and am thankful for all the guidance I received from my thesis director, Dr. Lars C. Grabow. He gave me the opportunity to explore the computational side of the chemical engineering field. He has been my mentor for a few research projects since my sophomore year as undergraduate student in University of Houston. I am very glad that I learnt a lot of new knowledge about catalysis and state-of-the-art modeling methods to analyze the interaction of chemical molecules on the surfaces of catalysts.

I would also like to thank Xiao Li, a graduate student in Dr. Grabow's group and my second mentor for this thesis project. She was very patient, helpful and answered all my questions when I first started picking up Python and working on the algorithm. It was a great pleasure working with her on this project related to computational modelling of aluminum distributions in zeolite structures, as well as on my previous project related to the reaction selectivity of substituted phenols.

I also want to thank the Computational Catalysis and Interface Chemistry (CCIC) group (Dr. Grabow's group) at the University of Houston for their help whenever I had problem with my scripts. Thanks to the Research Computing Data Core (RCDC) at the University of Houston for providing resources for high-performance computing, such as the Sabine and Opuntia cluster.

I want to express my gratitude to other committee members for this senior honors thesis defense. Many thanks to Dr. Jeremy Palmer and Dr. Fritz Claydon, who take interest in the details about my project and gave critical feedback on this paper. I am also grateful to The

Honors College of the University of Houston for giving me an opportunity to participate in research projects, such as Summer Undergraduate Research Fellowship (SURF) and this Senior Honors Thesis.

Finally, I would like to thank my loving parents for always supporting me and believing that I will achieve my dreams. They always care about my wellbeing, success stories and struggle for these four years at the University of Houston. Even though the journey for my undergraduate studies is coming to a closure, I am looking forward for my future career pathway wherever I would be.

ABSTRACT

Zeolites are porous aluminosilicates that afford exceptional benefits as catalysts in the petrochemical industry, or for vehicle emission control, to name a few. The presence of aluminum (Al) in the silicate (SiO_4) framework creates a charge defect, which leads to Brønsted acid sites when compensated with a proton. If the defect is compensated by a metal cation, a Lewis acid site is formed. For metal-exchanged zeolites, in particular, the Al distribution and probability of forming paired Al sites in the framework determines the metal speciation and, in turn, the selectivity and activity of the catalyst. This thesis project aims to create an algorithm that can estimate the Al distribution and probability of forming paired Al sites in zeolites based on user-defined parameters: a) Silicon-to-aluminum ratio (SAR), b) simulation temperature, c) number of trials generated for statistical analysis, and d) Al-Al interaction energy. Two different implementations of a Monte Carlo algorithm were tested and evaluated for zeolites with CHA, MFI and MWW framework structures. The traditional (MC1) and modified (MC2) Metropolis Monte Carlo Methods have different means in generating the initial guess for the Al distribution. In MC1, the initial population is random while conforming to Löwenstein's rule, while in MC2, Löwenstein's rule is used to generate an initial guess with maximized density of Al site pairs. Even though MC2 has a higher computational cost than MC1, MC2 is better than MC1 because it is less prone to error and has a higher overall efficiency. The final algorithm generates population statistics that are in full agreement with expected results for the provided Al substitution energies and the probability of forming paired Al sites.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

NOMENCLATURE TABLE

Table 1: Nomenclature Table

Å	Ångstrom
Al	Aluminum
ASE	Atomic Simulation Environment
CHA	Chabazite
COF	Covalent Organic Framework
DFT	Density Functional Theory
DNP-NMR	Dynamic nuclear polarization – Nuclear Magnetic Resonance
d_{min}	Minimum distance
d_{max}	Maximum distance
EMR	Even-number-membered ring
E_N	Relative Stability of the new T-site
E_O	Relative Stability of the original T-site
eV	Electron-Volt
FD	Framework Density
IRMS-TPD	InfraRed Mass Spectroscopy - Temperature-Programmed Desorption
IZA-SC	The Structure Commission of the International Zeolite Association
M	Cationic Transition Metal Species
MC	Monte Carlo

MOF	Metal-Organic Framework
MRs	Membered Rings
O	Oxygen
OMR	Odd-number-membered ring
os	Operating system
P	Probability
R	Boltzmann constant
SAR	Silicon-to-Aluminum Ratio
Si	Silicon
T	Temperature
T_O	Original T-site
T_N	New T-site
ΔE	Delta Energy (Final E – Initial E)

CHAPTER ONE – INTRODUCTION

1.1 Zeolites and Löwenstein's rule

Zeolites are porous crystalline materials consisting of Si and O atoms, as well as heteroatoms, such as Al. They are widely used as catalysts in the production of fuels and chemicals from petroleum-based feedstock. They have also shown potential for upgrading biomass feedstock as well. Catalysis by zeolites is controlled by confinement effects in their narrow pores and channels of various sizes as well as their Brønsted acidity resulting from compensating the negative charge on a framework Al atom with a proton. Moreover, their catalytic capabilities can be greatly expanded by introduction of Lewis acidity, resulting from framework defects or ion-exchange with metal cations. Without Lewis acid sites, some intriguing chemical reactions, such as aldol condensations and glucose isomerization, cannot be catalyzed [1]. Zeolites also catalyze selective oxidation reactions of organic compound with hydrogen peroxide or organic peroxides, such as oxidation of alkanes to alcohols or ketones and epoxidation of olefins [1]. Since zeolites are crystalline, shape selective and some are hydrophobic, they can catalyze Lewis acid chemistry even in water [1].

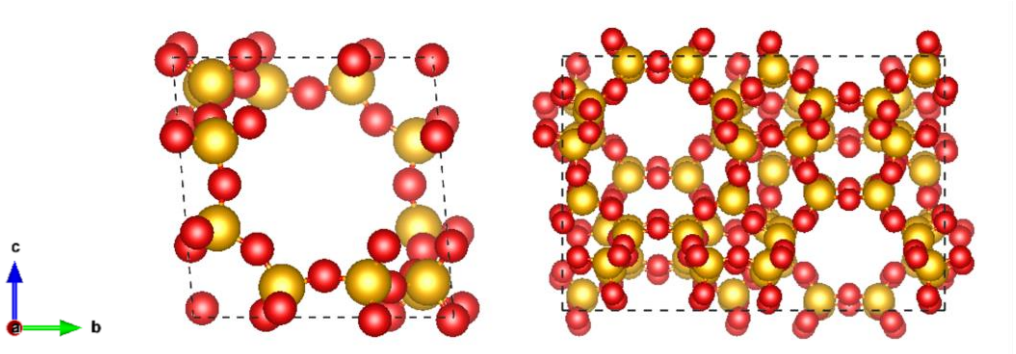


Figure 1: Zeolite framework structure of Chabazite (left) and ZSM-5 (right) projected along [1 0 0] with upward vector [1 0 1]. (Yellow – Silicon ; Red – Oxygen ; Silver – Aluminum)

Zeolites can be easily distinguished from zeolite-like materials based on the framework density (FD), which is the number of tetrahedrally coordinated framework atoms (T-atoms) per 1000 cubic Angstroms ($Å^3$) [2]. Metal-Organic Frameworks (MOFs), Covalent Organic Frameworks (COFs), and hierarchical zeolite materials are some examples in the zeolite-like materials category [3]. The maximum FD for zeolite ranges from 19 to 21 T-atoms per 1000 $Å^3$ depending on the average size of the smallest ring. FD values may depend on the chemical compositions and is related to pore volume but does not reflect the size of pore openings. The typical zeolite framework has each T-atom connected to 4 neighbouring T-atoms through oxygen bridges [2]. Zeolite frameworks contain open cavities in the form of channels and cages. Water molecules and exchangeable cations occupy these open cavities during synthesis, but under reaction conditions the void spaces are filled with reactants, intermediates and products [4]. The Structure Commission of the International Zeolite Association (IZA-SC) has adopted a three-capital-letter universal code to identify all different types of zeolite frameworks. These universal codes will be used in this thesis, e.g., Chabazite (CHA). Figure 1 shows two example of zeolite framework structures, CHA and ZSM-5 (MFI). They share a common structure composition where each T-site is surrounded by 4 oxygen atoms.

Due to the long-range crystalline structure of zeolites, they can be more systematically investigated than mixed metal and metal oxide catalysts. Zeolites also have various beneficial properties, such as their tunable acidity and microporosity that contributes to the shape selectivity [5]. When one or more Si atoms are substituted with Al atoms, the Silicon-to-Aluminum Ratio (SAR) will be used to identify if the zeolite is Si-rich or Al-rich. Si-rich zeolites with high acid strength are applied in hydrocarbon processing industry while

Al-rich zeolites (SAR = 1 – 6) are used in catalytic cracking and fuel reforming processes as catalyst [6].

Brønsted acids can donate protons while Lewis acids can accept electron pairs. Both acid sites are observed in zeolites. Brønsted acidic protons are located at the bridging O atom between the tetrahedrally coordinated Si and Al atoms. A Brønsted acid site is produced by replacing a Si atom (or T-atom) with a di- or trivalent atom, such as Al atom. Since Al has a different charge as compared to Si, the negatively charged (anionic) silicate-based framework is counterbalanced by adding cations, such as H^+ [5]. The counterions can be cationic transition metal species (M). The general composition formula for aluminum substituted zeolite is $M^{n+}_{x/n}Al_xSi_{1-x}O_2$, where *n* is the valence of counterion M and *x* is the substitution amount of Al.

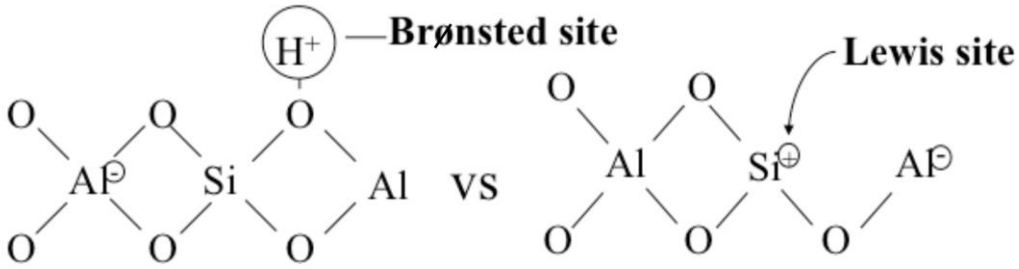


Figure 2: Brønsted acid sites donate protons while Lewis acid sites accept electron pairs (Figure source: Dr. Semih Eser – Penn State) [7]

The Lewis acid centers in zeolites are electron deficient sites or cationic sites formed near defects. Both acid sites shown in Figure 2 are often present simultaneously in zeolite structures at high temperatures [8]. A paired Lewis acid substituted zeolite can be created by combining a silicon source, a paired Lewis acid monomer and a structure-directing agent to form a precursor gel (if necessary) in an aqueous solution. A structure directing

agent can be containing a quaternary or tetraalkyl ammonium hydroxide group while a silicon source can be a tetraethylorthosilicate. Another method would be reacting a precursor gel under effective condition to form a zeolitic precursor and then treating it with a microporous crystalline framework to form an isomorphous zeolite [1].

1.2 Motivation for studying Acid Site Distribution in Zeolite

This thesis project focuses on creating an algorithm that can estimate the distribution of Al sites in an arbitrary zeolite framework with a given SAR value. This distribution contains information about the probability of finding paired acid sites, which are believed to have different catalytic properties than isolated sites. A paired acid site exists in the form of [-O-Al-O-Si-O-Al-] where two Al atoms cannot share the same O atom neighbor (Figure 3). For example, in the enzymes that are used in aldol condensations and glucose isomerization, catalytic pairs of metal centers are thought to increase their performance, such as selectivity and activity [1]. By translating the beneficial features of enzymes to heterogeneous catalytic materials, such as zeolites, we anticipated that their performance can be greatly increased.

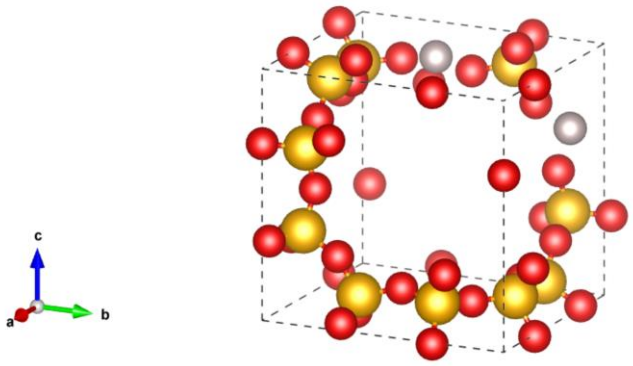


Figure 3: This diagram shows a CHA zeolite structure with one paired acid site where T-site #1 and #8 are substituted with Al atoms. The Al atoms cannot be next to each other, but they can exist as paired acid site in form of [-O-Al-O-Si-O-Al-].

In a paired Lewis acid site of a zeolite, the distance between first heteroatom and second heteroatom should be less than 10 Å [1]. The arrangement should also abide by Löwenstein's rule and thus no metal center or heteroatom should be directly next to each other. The formula of general paired acid sites is defined by this equation : $[-M^1 - O - (Si - O)_n - M^2-]$. M^1 is the first heteroatom while M^2 is the second heteroatom. The distance between two heteroatoms is usually more than 10 Å when *n* is more than 1 but the general equation is simplified with $n = 1$ when heteroatoms are in pairs for this thesis.

The paired acid sites can be comprised of the same or different heteroatoms. An advanced spectroscopy method, such as DNP NMR, can be used to verify the incorporation of paired acid sites in a zeolite structure. In previous research, a powerful method consisted of Density Functional Theory (DFT) calculation with IRMS-TPD experiment has improved the characterization of zeolite's acidity [5].

Hence, the algorithm in this research paper includes the two main factors (Al/Si substitution and Al-Al interaction energy) that affect the distributions of acid sites or the locations of the heteroatoms during the process of generating random arrangement of Al atoms in the zeolite framework. In general, the catalytic sites of heterogeneous catalytic materials are randomly distributed or in non-uniform arrangement [1]. The exact location of Al atoms in a zeolite framework and the resulting acid site distribution are difficult to measure or control. However, the location of the Al substitute is important because they are linked directly to Brønsted active sites [9].

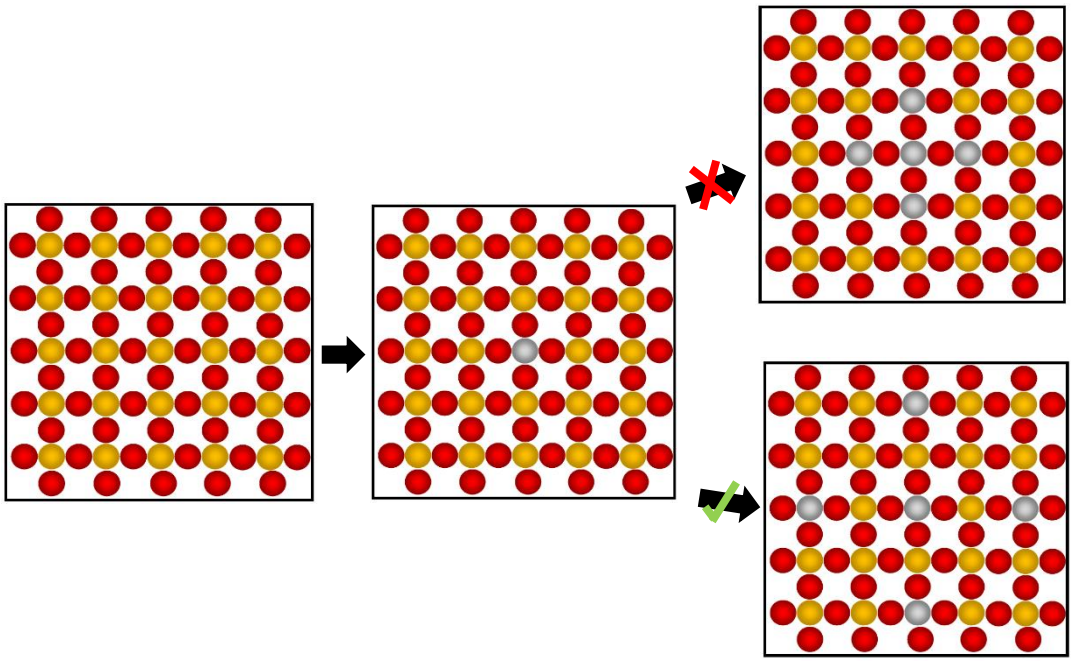


Figure 4: Löwenstein's rule states that the neighboring T-sites of the first Al-substituted T-site cannot be substituted with Al atoms but they can exist in the form of -(-O-Al-O-Si-O-Al-O-)-.

A few heuristic rules, however, exist. Löwenstein's rule excludes the sharing of an oxygen atom by two AlO_4 tetrahedra (due to low stability) and results in the absence of Al-O-Al sequences in the zeolite framework (Figure 4). Dempsey's rule states that Al ions tend to distance themselves as much as possible from each other in the zeolite framework at a given Si/Al ratio. Takaishi and Kato challenged Dempsey's rule and stated that two Al atoms cannot be present only in 5-membered rings (MRs). However, Takaishi's and Kato's argument only applies to some specific conditions and does not directly contradict Dempsey's rule. Hence, the reliability of those rules still requires further verification [6] [10].

The final algorithm of this thesis project can help researchers to get a good estimate before carrying out experiment related to substitutions of Al atoms into zeolite silicate

framework. This code requires the user to define the SAR value, temperature of experiment, the number of trials with selected framework and Al-Al interaction energy.

1.3 Metropolis Monte Carlo Method

A Monte Carlo simulation is a type of simulation that relies on repeated random sampling and statistical analysis to compute a set of data or results. This method is closely related to simple random experiments (where a set of experiment can be just repeated for certain number of times to get the average value). The common components for a Monte Carlo simulation are the input parameters (depend on the external factors), a mathematical model and experiment output. A typical Monte Carlo simulation follows four steps [11]:

1. Static Model Generation
2. Input Parameter Identification
3. Random Variable Generation
4. Analysis and Decision Making

Monte Carlo (MC) simulations are capable of methodically investigating the complete range of possible scenarios (from the worst case to the best outcome) for each input variable. By collecting and analyzing output values from several simulation runs (with different input parameters), researchers can make better decisions or get better results from a statistical analysis [11]. For years, Monte-Carlo simulations and the configuration matrix approach were used to indicate the distribution of Al atoms, mainly the number of paired acid sites (Al-O-Si-O-Al) sequences in Si-rich zeolites. However, these methods cannot predict the framework Al distribution accurately [6]. Hence, this thesis focuses on applying the Al/Si substitution energy (stability relativity) and Al-Al interaction energy (attractive

and repulsive force between heteroatoms) to make a better prediction of Al distribution in zeolite framework.

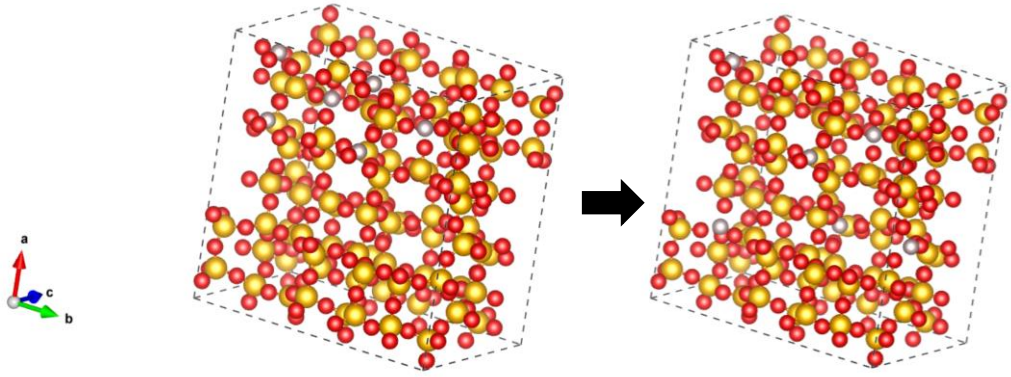


Figure 5: (Left) The Metropolis MC method starts by substituting Si atoms with Al atoms until the desired SAR value is achieved. (Right) The Al atoms will move around to find the most stable configuration using the probability equation.

The traditional and modified Metropolis Monte Carlo methods are both used in the algorithm to analyze the benefits and disadvantages of both methods. The traditional Metropolis MC method randomly picks one Si atom in the pure-silicon zeolite framework to start the Al substitution and continues with randomly selecting another Si atoms (all abiding by the Löwenstein's rule) to be substituted until the SAR value is satisfied (Figure 5). The modified Metropolis MC method starts the same way as the traditional MC method, but it continues with replacing all the Al atoms into the framework as the neighbors of the initial Al atoms. Then, these Al atoms will move around within the framework to find the most stable configuration. The details of these two methods will be further discussed in Chapter Two – Methods of this paper.

1.4 Project Statement and Scope

The main objective of this thesis is to create a Monte Carlo code to determine the Al distribution and number of paired active site in a zeolite framework and plot the distribution of paired active site with the parameters selected for the structure framework analysis. Previous research determined the location of the Al neighbors using atom probe tomography and analyze the distribution of Al over different T-sites in zeolites using X-ray emission spectroscopy [12] [13]. For this thesis, Al substitution relativity data and random number code are used to determine which atoms in the unit cell are to be substituted.

CHA, MFI (ZSM-5) and MWW (MCM-22) structure framework are used to test run the algorithm. By using previous Al substitution relative energies from DFT calculations, several aluminosilicate zeolites are generated with random Al arrangement with consideration of Al substitution and Al-Al interaction energies. There is a closely related published research paper that evaluates the energetics of 209 existing framework topologies with different SAR, Al locations and proton configuration. They used the Monte Carlo sampling method to assist them in analyzing the energetics distribution of randomly Al-substituted zeolite [9]. This research project is approaching the problem from the opposite direction, where the energetics of zeolite framework are used to generate distributions of Al in zeolite frameworks by implementing the Metropolis Monte Carlo sampling method.

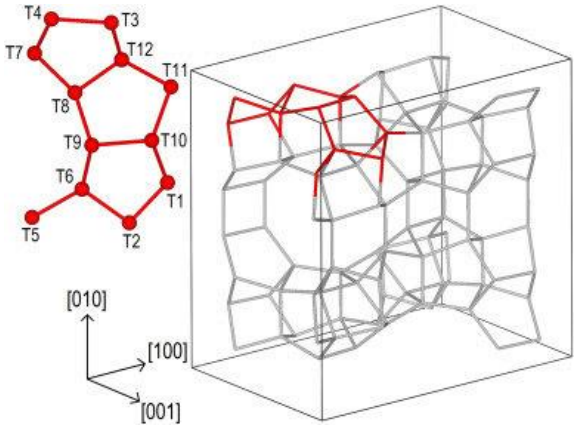


Figure 6: Location of twelve distinct T-site in a MFI zeolite unit cell (Figure Source: Hernandez-Tamargoetal, 2016) [14].

Chabazite (CHA) has a very simple structure with all equivalent T-sites characteristics, making it a suitable zeolite framework to start with. It also allows fast calculations because there are only 36 Si and O atoms in the unit cell [5]. CHA only has one distinct T-site and four distinct O sites. On the other hand, one typical unit cell of MFI is composed of 96 T atoms and 192 O atoms with two types of interconnected channels (straight and sinusoidal). There are 12 geometrically distinct T-site locations and 26 distinct O sites (Figure 6). MWW structure framework has 8 T distinct sites and 13 distinct O sites.

The algorithm for this research project requires four parameters: SAR value, temperature, numbers of trials for selected framework and Al-Al interaction energy (optional). The Al/Si substitution energy and the Al-Al interaction energy are the main factors that affect the distribution of Al atoms in the zeolite framework. By generating a few trials (ranging from 1 to 50) of Al substituting Si in zeolite framework, the typical distribution of acid site can be determined by statistical analysis and graphing the results.

CHAPTER TWO – METHODS

2.1 Overview of the Algorithm

The algorithm is divided into several functional blocks, performing tasks such as importing the key modules into Python or graphing the acid site distribution determined by the user inputs. An example of a module typically imported at the beginning of each Python script is the Atomic Simulation Environment (ASE) module. It provides important functionality and a database that allow the user to visualize, read and write atoms, molecules and chemical compounds. Other modules, such as numpy, matplotlib, os (operating system dependent functions) and math, are also accessed through the import function.

The purpose of the developed algorithm is the prediction and analysis of acid site distribution in zeolite framework based on SAR, simulation temperature, number of trials and Al-Al interaction energy. Currently, this code can be used on CHA, MFI (ZSM-5) and MWW (MCM-22) structure frameworks. This code can be applied for any zeolite if there is information on the relative stability of T-sites. Hence, the user can choose one zeolite framework and set the other four parameters. The code is developed by leveraging existing modules (e.g., ASE for visualization) and logical thinking (with basic programming language knowledge). By viewing the structure on ASE, modifications are done to improve the algorithm accuracy.

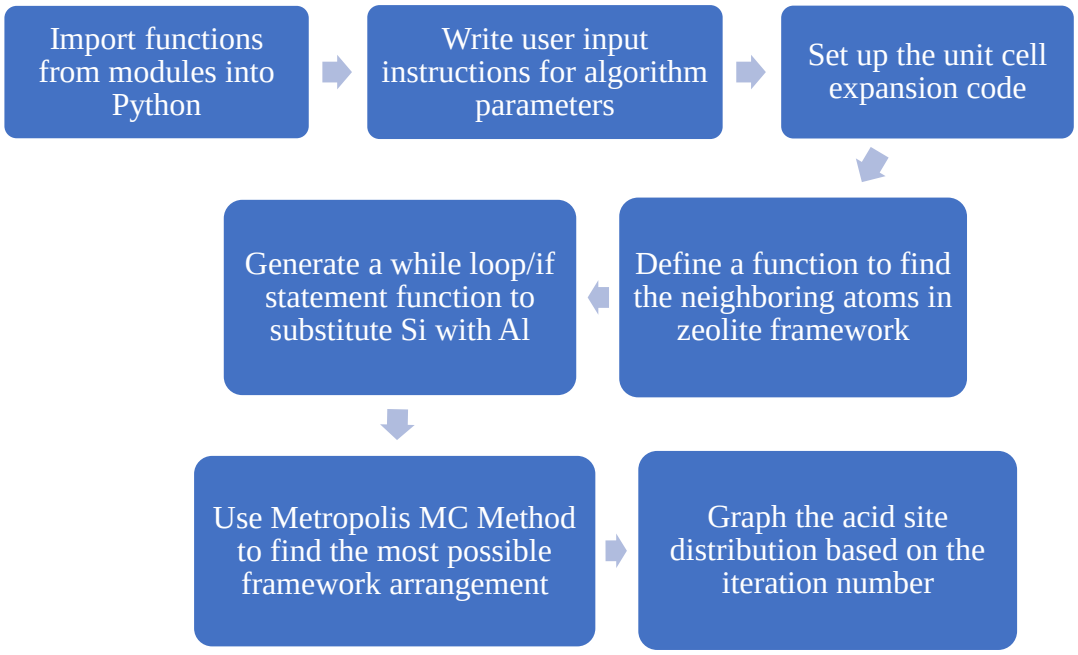


Figure 7: The overall algorithm process flowchart

The first critical part of the algorithm is the code for generating the initial guess by selecting the Si atoms to be substituted with Al atoms. First, an expansion code for unit cell is written to include the neighbors of the atoms on the border (corner) of a unit cell. Next, several other functions are written to record the neighbors of the selected Si atom and use while loops with if statements to substitute Si atoms with Al atoms (while abiding by Löwenstein's rule). This part of the algorithm was the most challenging to code because zeolite geometry is more complicated with different types of channels. Finally, the location of Al atoms is randomized with and without energetic values using the Metropolis Monte Carlo method to sample the most plausible Al-populated zeolite frameworks, one for each trial run (Figure 7).

In addition to a random distribution, the algorithm of this thesis considers two energy-biased cases that will affect the equilibrium Al distribution in zeolite framework. The two

energy contributions are the Al/Si substitution energies at any geometrically distinct T-site, and the Al-Al interaction energy (related to Dempsey's rule). The interaction energy is not only determined by Coulombic interactions, but it also depends strongly on the density of the framework, especially within the *xy* plane [10]. It was found in previous research that the interaction energy changes approximately as the inverse square distance between the two Al atoms [10]. Since the distribution of Al atoms is not random, the normal Monte Carlo simulation method (that is based on pure random calculation) cannot be used. Finally, the acid site distribution graphs are generated to show the occupation of T-sites and the probability of finding paired acid sites based on the selected SAR and simulation temperature. The main blocks of the algorithm are described next in more detail.

2.2 Random Selection of Silicon Atom and Zeolite Unit Cell Expansion

The process of substituting Si atoms with Al atoms into the framework starts with picking one Si atom. This is done by using a simple algorithm that generates random number. However, there is a modification to the algorithm because the Si atoms' number in the zeolite framework sometimes does not start with number 1. Hence, the amount of Si and O atoms in framework is calculated using another algorithm with while loops and ifs statement (Refer to Algorithm 1 in Appendix). By using the random module in Python to generate pseudo-random numbers, a random integer between the range of Si atoms number tags is generated. Algorithm 2 creates a list of random integers together without repetition (Refer to Algorithm 2 in Appendix).

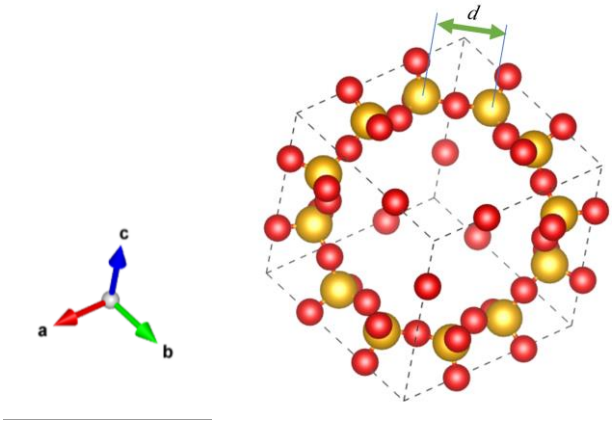


Figure 8: The minimum and maximum bond length (d_{min} and d_{max}) are determined through two neighboring Si atoms. Most bond lengths are in range of 3.0 to 3.3 Å.

Since the zeolite framework is analyzed within a size of unit cell, it is hard to find the neighboring atoms of a selected atoms at the border (e.g., corner) of the unit cell. The problem was solved by writing an algorithm to expand the unit cell according to its periodic boundary conditions. The code can just focus on the selected atom (Si_0) and expand the unit cell around it in three dimensions. Then, two lengths are determined based on the average bond length between two neighboring Si atoms: minimum and maximum distance (d_{min} and d_{max}). These two values are used to find the selected atom's neighbors (Figure 8). The whole section of algorithm is shown in appendix (Algorithm 3).

The first Si to be substituted would be the first number (Si_0) in the list of random integers. From this point, two different Metropolis Monte Carlo (MC) method are tested to evaluate which one is more feasible and accurate. The Metropolis MC method originated from a research done by Nicholas Metropolis et al., on "Equation of State Calculation by Fast Computing Machines" [15]. Instead of choosing a configuration randomly, the original configuration is changed by weighing them with a probability proportional to $e^{(\frac{-\Delta E}{RT})}$.

1. The traditional MC method (*MC1*) will pick the second number (Si_1) from the random integer list and run the number through another algorithm to make sure that Si_0 and Si_1 are not next to each other in the framework. This selection process will continue while abiding to Löwenstein's Rule. If the location of Si_1 does not agree with the Löwenstein's rule, the third number (Si_2) will be selected and go through the same selection process. This process will stop once SAR value is achieved (Refer to Algorithm 4 in Appendix). The algorithm also has a component that will calculate the substitution energy with a probability before Al atom replaces Si atom (Refer to Algorithm 5 in Appendix).

2. The modified Metropolis MC method (*MC2*) initializes the framework with the highest possible local Al density, i.e., it maximizes the number of neighboring Al atoms. Since the arrangement must uphold the Löwenstein's rule, the closest arrangement of two Al atoms nearby each other would be Al_0-O-Si_1-O-Al_2. By using a code to find the neighbors of Si_1, three Si atoms (other than the first Si_0 that already substituted with Al_0) will be substituted one by one. This process continues until the SAR value is achieved. Then, the Al atoms in the zeolite framework will move around by using a probability equation ($P = e^{(\frac{-\Delta E}{RT})}$). The difference between the substitution energy of the new and the original location is denoted ΔE. For example, ZSM-5 has 12 distinct T-sites and all distinct T-site have different substitution energy values. By taking the difference between E_{T1} and E_{T2}, the value of ΔE will be determined. R is the Boltzmann constant while T is the simulation temperature.

2.3 Al/Si Substitution and Al-Al Interaction Energy Calculations

The basic version of the algorithm does not use any Al/Si substitution or Al-Al interaction energies and generates a randomized arrangement that obeys Löwenstein's rule. After that, the code is expanded by taking into account the actual Al/Si substitution energy values. Note, for the CHA zeolite framework, this part of algorithm is not needed, because all the T-sites are equivalent and have the same substitution energy value. However, MFI (ZSM-5) and MWW (MCM-22) have 12 and 8 distinct T-sites, respectively, each with a different substitution energy (Table 2). The substitution energy values were previously obtained through DFT calculations in Dr. Grabow's research group [16].

Table 2: The relative stability of each distinct T-site of H-ZSM-5. As the value of relative stability becomes larger, i.e., the position becomes less stable, the lower the possibility of Al atom to be substituted at the T-site [16].

T-site	Relative Stability
1	0.26
2	0.28
3	0.26
4	0.10
5	0.32
6	0.35
7	0.00
8	0.21
9	0.38
10	0.17
11	0.26
12	0.20

The trend in relative energies can be translated to Boltzmann probabilities. Generally, the probability of finding Al atoms at one of two specific sites (P_1 and P_2) is proportional to the negative exponential of their total energy difference ($e^{(\frac{-\Delta E}{RT})}$) [10]. The outcome of the Boltzmann equation (P) is convenient because the exponential component in the probability equation can normalize the range to between 0 and 1, if we require ΔE to be positive. This probability equation is used in the Al/Si substitution energy part of the algorithm.

For both the traditional and modified Metropolis MC method (MC1 and MC2), a random number (R) is generated for the cases where the ΔE in the probability equation is a positive value (which means that the original position is more stable for Al atom to substituted than the new positions). This random factor is a "wild factor" and it is used to give the Si atom another chance to be substituted with an Al atom. When the probability (P) is equal or bigger than R, the substitution process can occur. If not, a new position will undergo the same substitution procedure.

Figure 9 shows the three possible pathways that leads to two outcomes with certain criteria from this energy part of the code:

1) If the relative stability (E_O) of the original T-site (T_O) that is initially substituted by Al is **lower** than the relative stability (E_N) of the suggested new T-site (T_N) populated by Si, Al will move to the new T-site and substitution will occur.
2) If the relative stability (E_O) of the original T-site (T_O) that is initially substituted by Al is **higher** than the relative stability (E_N) of the suggested new T-site (T_N)

populated by Si, the probability value (P) will be calculated using $P = e^{(\frac{-\Delta E}{RT})}$ and a random factor (R) between 0 and 1 will be generated.

a. If P is **bigger** than R, the Al atom will move to the new T-site and substitution will occur.

b. If P is **smaller** than R, the Al atom will stay at the original T-site and substitution will not occur.

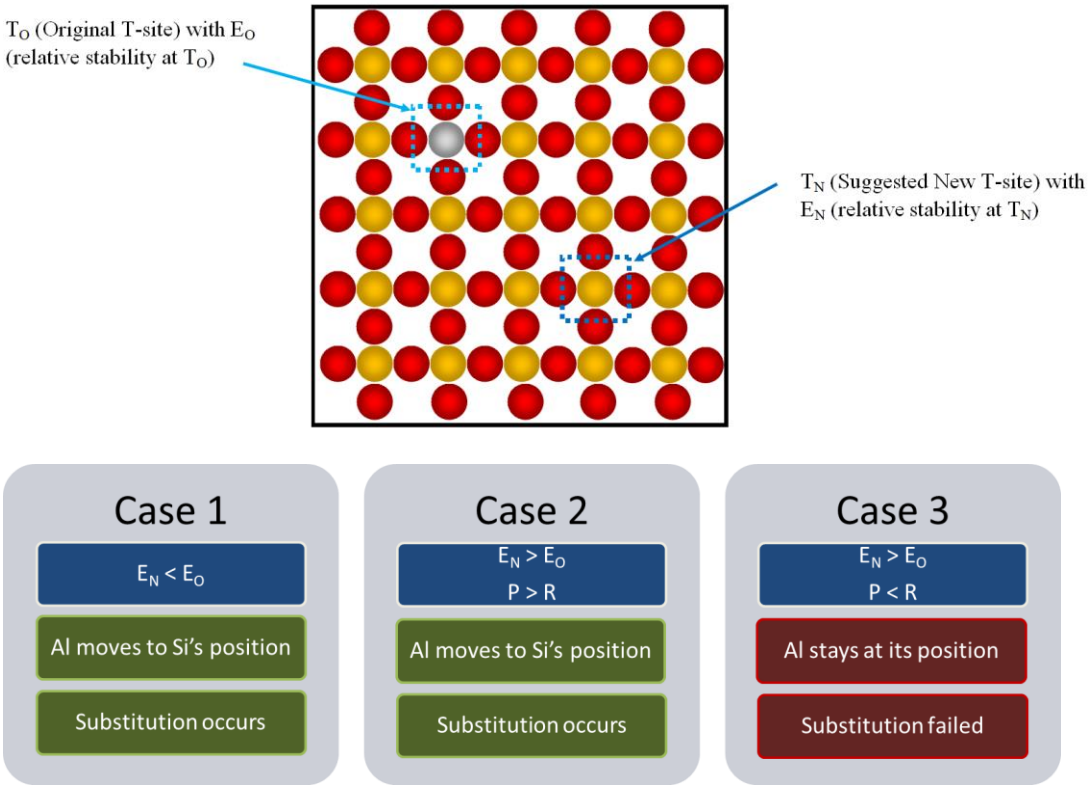


Figure 9: Three possible cases when the energetics part of the code is prompted.

To account for the Al/Si substitution energies in the acceptance criterion as mentioned before for the Metropolis Monte Carlo code (MC1 and MC2), a section of the algorithm is added to create a dictionary for the energy values of all distinct T-sites. First, all the Si atoms in the zeolite structure framework are categorized into own their T-site group. Then,

all relative stability values are assigned to each T-site group. (Algorithm 6 in Appendix) Whenever the code that considers the energetics is prompted, this section of algorithm is used (Algorithm 7 in Appendix). After that, an Al-Al interaction energy factor is added into the equation using another algorithm. This factor is similar to Dempsey's rule where Al atoms are more likely to stay as far as possible from each other, but each T-site has different interaction energy value.

2.4 Graphing of Potential Acid Site Distributions

After a user-defined number of trials was performed and several zeolite framework arrangements were generated, all the information related to lattice geometry and ionic positions are stored in a POSCAR file, which is a structure file used by the electronic structure code VASP and readable in ASE. This POSCAR file can be opened to visualize the T-site occupation by Al and Si atoms. Another part of the code will detect the number of paired acid site (-O-Al-O-Si-O-Al-O-) in every POSCAR and graph the number of paired acid site against the number of trials. The histogram should illustrate the trend of the paired acid site distribution and show the average number of paired acid site from the SAR, simulation temperature and Al-Al interaction energy parameters (Figure 10).

Based on the histogram, the user can determine which SAR value, simulation temperature and Al-Al interaction energy will create the zeolite framework with the ideal amount of paired acid site. For a zeolite framework structure with high SAR value and/or repulsive Al-Al interaction energy, the number of paired acid sites will be lower because there is a smaller amount of Al atoms substituted in the framework. It is hard to find paired acid sites because Al atoms will usually substitute themselves at T-sites as single isolated acid sites. They will preferentially substitute into the most stable sites, such as the T7 sites

in ZSM-5 (MFI) that are far apart from each other. If the synthesis happens at a higher temperature, atoms have higher kinetic energy to move around the framework. This kinetic energy will make the distribution more random and enable more Al atoms to populate the zeolite framework at lower SAR. Hence, it made the zeolite frameworks with a lower SAR value easier to be formed.

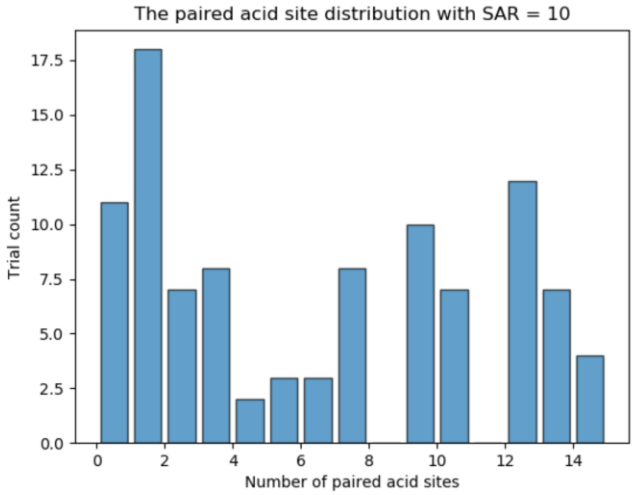


Figure 10: First histogram of paired acid site distribution (before algorithm modification) with SAR = 10 at simulation temperature of 1400 K and no Al-Al interaction energy (Total trial count = 100)

The histogram is graphed by using a collection of command style functions in `pyplot` from the `matplotlib` module. These graphing functions enable Python to access matplotlib like MATLAB, create figures and modify plots to suit the purpose of users [17]. Since the algorithm is coded in Python, matplotlib is the obvious and easiest way to visualize data directly from the generated results. The portion of the code that generates plots is attached in Appendix (Algorithm 8).

CHAPTER THREE – RESULTS AND DISCUSSION

3.1 Development of Plausible Al/Si Substituted Zeolite Frameworks

Catalysts with CHA framework structure are commonly used, for instance in gasoline or diesel engine exhaust gas treatment. Most notably, copper-exchanged chabazite catalyst exhibits excellent hydrothermal stability and high performance for the reduction of nitrogen oxides over a wide range of temperatures [18]. Moreover, the T-sites in CHA are all equivalent, which greatly simplifies the development and testing of the code for paired acid site distribution. Hence, we first validate the algorithm on the CHA framework without consideration of T-site specific Al/Si substitution energies or Al-Al interactions.

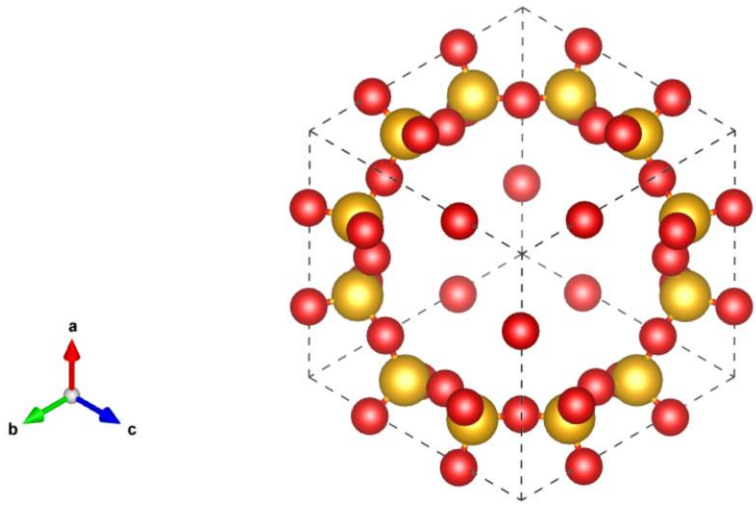
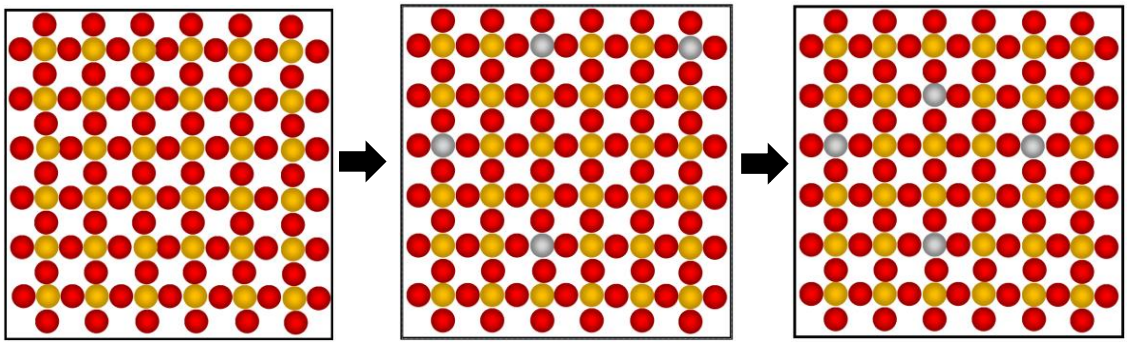


Figure 11: View of a CHA unit cell with projection vector [u v w] = [1 1 1]. This small pore framework has 4, 6, and 8 membered rings (MR) and all 12 T-sites are equivalent.

The unit cell of CHA contains 12 Si atoms and 24 O atoms, where each Si atom is surrounded by four O atoms (Figure 11). Since there is only one distinct T-site, the algorithm only needs to generate a random number to pick one T-site for Al/Si substitution without considering the Al/Si relative substitution energy. Hence, MC Method 1 with

randomly selected substitution sites for initial guess is applied on this structure framework to evaluate the efficiency of the algorithm. Usually, Al atoms will be substituted at the most stable T-site and continue to populate the framework in the same trend until SAR is reached. The code works fine for generating zeolite frameworks with high SAR values, because only a small number of Al atoms needs to be substituted into the pure silicate zeolite structure. However, when MC Method 1 is tested for low SAR values, the code will stop and display an error message to show that the SAR value cannot be reached and no more Si atoms in the zeolite structure can be substituted with Al atoms (Figure 12). The amount of error trials is more than the amount of successful trials.



(a) (b) (c)

Figure 12: The MC Method 1 randomly substitutes Si with Al throughout the entire process. The example in this figure has SAR of 8. (a) A silicate framework is schematically depicted. (b) Al atoms are substituted into T-sites at random positions without violating Löwenstein's rule. (c) Al atoms are relocated by using the Boltzmann probability function.

The reason behind the unsuccessful attempts is due to the randomized T-site selection that results in non-alternate arrangement on the zeolite framework. For example, when SAR = 1, all 12 T-sites of the CHA framework need to be in the form of $(\text{-O-Al-O-Si-})_6$. In this case, there are only two possible configurations: a) $(\text{-O-Al-O-Si-})_6$, where first T-site is an Al atom; b) $(\text{-O-Si-O-Al-})_6$, where first T-site is a Si atom. Hence, if the first and

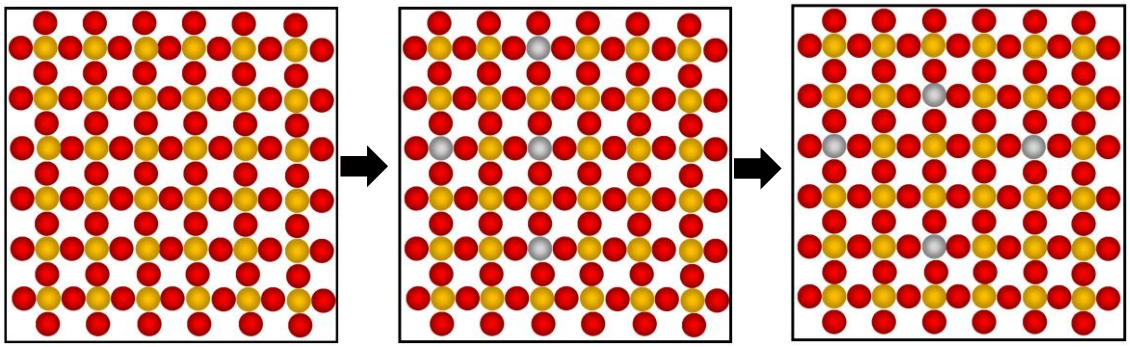
second T-sites that are chosen to be substituted is in the (-O-Al-O-Si-O-Si-O-Al-) arrangement, it is impossible for the other four Al-atoms to be substituted into the framework without violating Löwenstein rule.

The Al atoms substitution code becomes more complicated for framework structures that contain odd-number-membered rings (OMR), such as ZSM-5 and MCM-22. To rigorously test this part of the code, we use MFI (ZSM-5) with five-membered rings. For a zeolite framework with even-number-membered rings (EMR), the Al substitution process is faster and simpler, because more than one T-site can be substituted simultaneously. In contrast, MFI's T-sites can only be substituted one-by-one, thus slowing down the algorithm. Zeolites with OMR structure can never reach SAR = 1 and the first substituted T-site position will determine its lowest possible SAR value.

MC Method 2 uses a non-random initial guess and we evaluate its functionality and performance in comparison to MC Method 1, especially for trial runs with low SAR values. This method is developed by using the MFI framework structure of zeolite ZSM-5, which possesses 12 crystallographically and energetically distinct T-sites. MFI has a peculiar 3-dimensional 10-ring channel system with intersecting straight and sinusoidal channels. This kind of structure contributes to the exceptional catalytic properties of the MFI framework. ZSM-5 is one of the most versatile material used in heterogenous catalysis for petrochemical processes, such as xylene isomerization, methanol-to-gasoline conversion, fine chemistry and pollution control [10].

There are two main segments in MC Method 2: (1) From the SAR input value, the total amount of Al atoms is substituted into the zeolite framework where all the Al atoms exist as paired acid sites. (2) The Al atoms will be moved around by using a Boltzmann rejection

criterion with energy consideration until a low energy structure has been found (Figure 13). It must be noted, that a single unit cell of MFI can only have a maximum of 13 Al atoms in a unit cell (minimum SAR = 6), because it consists of five-membered rings, and the perfect alternating $(\text{-O-Al-O-Si-})_n$ required for SAR = 1 cannot be obtained. Frameworks that can reach an SAR of unit must have rings with an even number of Si/Al members, such as SAT, for example.



(a) (b) (c)

Figure 13: MC Method 2 is carried out in two stages. (a) A silicate framework is schematically depicted. (b) Some T-sites that are neighboring to each other are substituted in pairs with Al atoms. (c) Al atoms are relocated to other T-sites with greater relative stability, according to the Boltzmann probability criterion.

The second part of MC Method 2 exchanges Al positions following Boltzmann probability statistics. It works well and efficiently for trial runs with high SAR value and higher temperature, because the majority of randomly suggested moves are accepted. As the SAR value becomes lower, the zeolite unit cell becomes saturated with Al atoms and the movement of Al atoms to other T-sites is restricted. Thus, the MC algorithm faces a large number of rejected moves. For example, when the SAR value is set to a value lower than 12, the algorithm will run, but it will spend substantially more time in the energy consideration's *for loop* statement. When the code is run with a higher temperature

parameter (e.g., 850 K), the likelihood of accepting energetically unfavorable Al exchanges according to the Boltzmann probability increases, and the algorithm completes the predetermined number of successful moves faster.

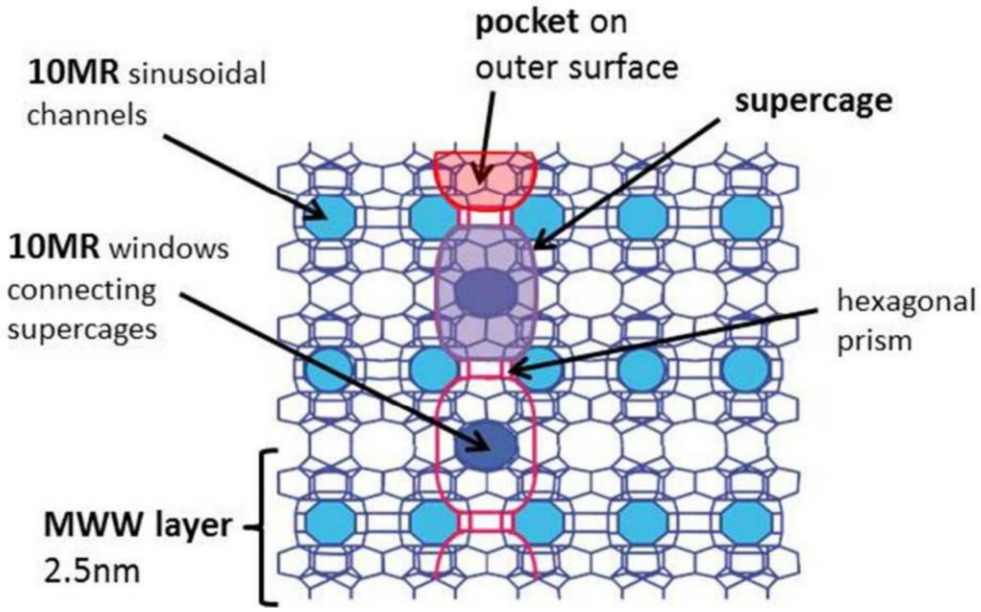


Figure 14: The MWW framework with characteristic locations highlighted (Figure Source: Martina Štekrová et al., (2018)) [19].

The framework MWW of the MCM-22 zeolite is also used in this research to evaluate the performance of the algorithm. MCM-22 was first synthesized by Mobil researchers and is different from other structures (MCM-49 and MCM-56) in the same family of MWW due to the packing structure of its layers [20]. MCM-22 is interesting for researchers due to its unique channel structure and its capability to produce ethylbenzene from the liquid phase alkylation of benzene and ethylene. It is also used in the reactions of large molecules because of its 12 MR half-supercages (surface pockets) on the external surface of zeolite (Figure 14) [21]. Zeolites with MWW framework are theoretically suitable for low SAR cases.

3.2 Algorithm Troubleshooting

The most challenging part of the algorithm is the Al atom substitution code that abides by Löwenstein's Rule. Since Al atoms cannot exist in the zeolite framework in -(O-Al-O-Al-O)- motifs, the code requires functionality to exclude all neighboring T-sites of Al-atoms in the framework. As an example, the first T-site to be substituted in the framework will be labelled as Al_0. By using a code to find the neighboring T-sites, all four Si atoms will be grouped together in a list (L_N) to avoid them from being substituted. Then, another T-site will be chosen to be substituted as Al_1 and its neighbors are also being grouped into the L_N. This process continues until SAR is reached or no more available T-sites that can be substituted.

Both MC methods developed in this project take more time to complete when the simulation temperature value is below ca. 400 K. As currently implemented, the code executes a pre-defined number of iterations that lead to a successful substitution attempt. A substitution attempt is only considered successful when a T-site that follows the Löwenstein rule is chosen and accepted according to the Boltzmann probability criterion based on the relative stability change between T-sites. At a lower SAR value, the zeolite framework is more saturated with Al atoms. Hence, there are less available T-sites that can be selected. For simulations with SAR value below 12, the algorithm slows down because the chances of running into errors is higher and the trial runs might take several minutes up to an hour depends on number of iterations. For the trials with very low SAR, the iteration loop function will be forced to stop when it reaches the limit of iterations.

The number of iterations in the Al/Si substitution code is important to ensure convergence to low energy structures and possibly the global energy minimum (Algorithm

9 in Appendix). The algorithm is set to let the Al/Si substitution code perform 200 iterations per number of Al atoms for SAR more than 12 and 400 iterations per number of Al atoms for SAR equal to or less than 12 in the zeolite framework. If the number of iterations is insufficient, the final structure will not correspond to the most stable form of the zeolite framework conformation and the average number of paired acid site will not be reliable. Hence, a lot of troubleshooting steps were needed to improve the capability of the algorithm.

The algorithm developed in this thesis project can create multiple output structures with a few simple steps. The Monte Carlo program is executed by a simple python command requiring four user inputs: SAR, number of trials, simulation temperature and Al-Al interaction energy value. The SAR value depends on the type of zeolite structure framework (e.g., ZSM-5's SAR is between 6 and 95). The higher the number of trials, the more reliable is the statistical analysis for the average number of paired acid sites. As the simulation temperature rises, the calculation speed of algorithm will increase, but the final structures can deviate from the low energy configurations. An Al-Al interaction energy with positive value (e.g., +0.1 eV) represents repulsive forces while a negative value (e.g., -0.2 eV) represents attractive interactions. Repulsive Al-Al interaction will decrease the average number of paired acid sites and vice versa.

3.3 Acid Sites Distribution Graph Analysis

The developed code includes functions to plot occupation statistics for easier analysis. We have tested the algorithm for consistency and reliability using selected test cases with known outcomes. The following list shows four types of plots that will aid in the interpretation of the simulation results:

a. Comparison of constant versus distinct T-sites relative stability

The MFI framework is a good model to test the population of Al atoms on different T-sites with either all equivalent T-sites or T-sites with distinct relative stabilities. Assuming all T-sites are identical with a relative stability of 0 eV, Figure 15 shows the output distribution of Al atoms on different T-sites. This simulation used 50 trial runs at SAR = 23, a simulation temperature of 400 K and no Al-Al interaction energy. As expected, we observe no preferential Al substitution of T-sites.

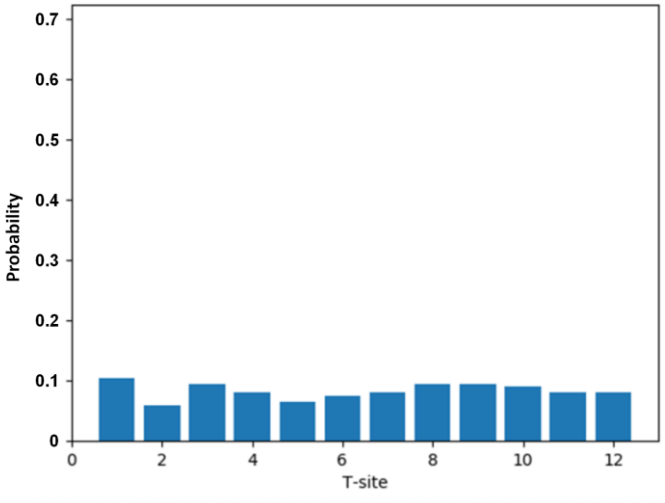


Figure 15: Distribution of Al atoms substituted on different T-sites of MFI assuming identical relative stability. (SAR = 23, simulation temperature = 400 K and trial runs = 50)

For comparison, we use the same algorithm but the values of the T-sites' relative stability are based on DFT calculations. As shown in Table 2, T-site number 7 (T7) is the most stable among the T-sites, and we expect Al atoms are prone to substitute the Si atoms at T7 positions more than the other T-sites. Figure 16 shows the result from 50 trial runs with SAR of 23, a simulation temperature of 400 K and

no Al-Al interaction energy. As expected, the highest peak from the bar graph is T7 (relative stability = 0 eV), followed by T4 (relative stability = 0.1 eV) and T10 (relative stability = 0.17 eV). The remaining T-sites are significantly less stable for Al atoms to replace Si atoms. This outcome suggests that the Al/Si substitution energy part of the algorithm is working as intended.

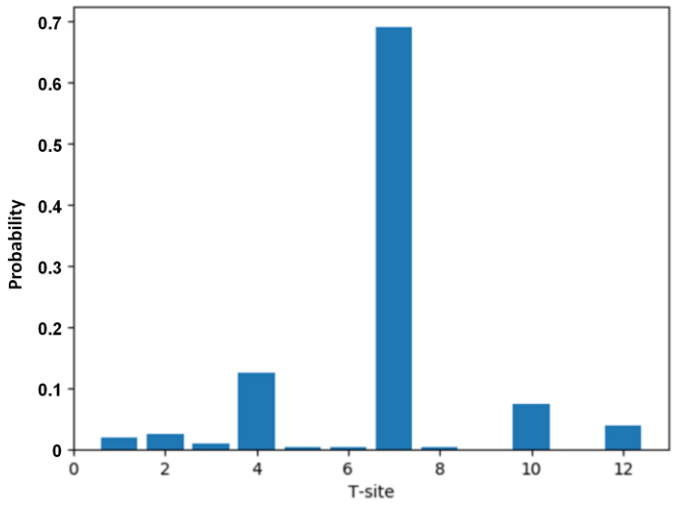


Figure 16: Distribution of Al atoms substituted on different T-sites of MFI with distinct relative stability. (SAR = 23, simulation temperature = 400 K and trial runs = 50)

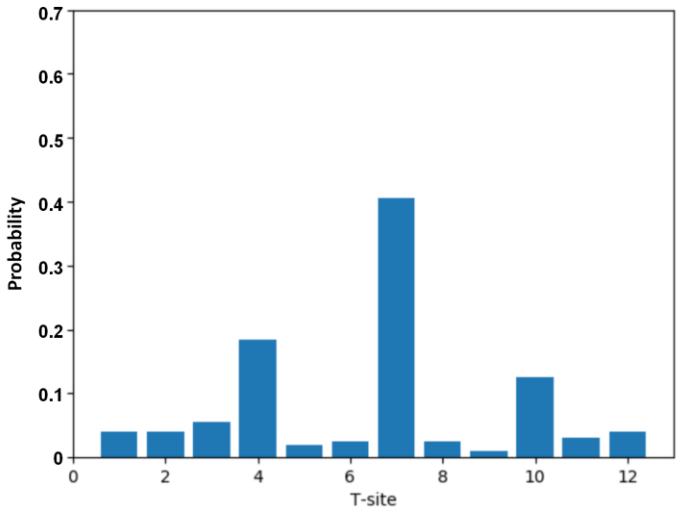


Figure 17: Distribution of Al atoms substituted on different T-sites of MFI with distinct relative stability. (SAR=23, simulation temperature = 850 K and trial runs = 50)

As the simulation temperature increases, the distribution of Al atoms becomes more random due to the value of probability (P) from Boltzmann equation that is getting closer to 1. This trend is consistent with the phenomena of atoms or molecules moving around with greater kinetic energy at higher surrounding temperature. In this case, Al atoms in a zeolite framework can substitute Si atoms easier and more randomly at 850 K than at 400 K (Figure 17). When the simulation temperature is increased to 2500 K (Figure 18), the distribution of Al atoms on distinct T-sites with different relative stabilities is closer to the distribution pattern of Al atoms on different T-sites with identical relative stability (Figure 15).

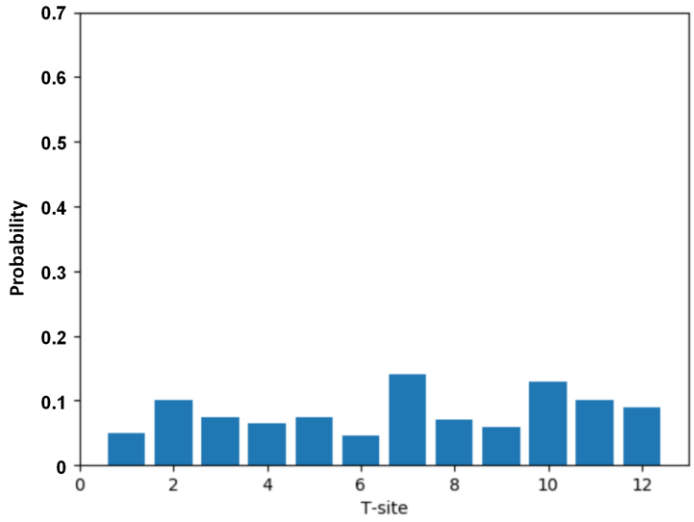


Figure 18: Distribution of Al atoms substituted on different T-sites of MFI with distinct relative stability. (SAR=23, simulation temperature = 2500 K and trial runs = 50)

b. Average number of paired acid sites per trial for a range of SAR

Next, we discuss the analysis for finding the average number of paired acid sites in a zeolite structural framework. We again chose the MFI structure for this test due to its complex structure arrangement and distinct T-sites relative stability values. Each trial run of the algorithm for different parameters is repeated for 25 times. The

average number of paired acid sites is obtained by dividing the sum of paired acid sites with the total number of successful trials. A trial run is only counted as successful when at least one Al atom was able to move to another T-site location after passing through the initialization part of the algorithm.

This part of the algorithm test is carried out for SAR = [95, 47, 31, 23, 18, 15, 12, 11, 9, 8, 7.5, 7, 6] when the number of Al atoms are [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13] respectively. Different simulation temperatures (400 K, 850 K, 2500 K) are used to study its effect on the average number of paired acid sites (Figure 19). An extreme temperature value (2500 K) is used to investigate if trends start to approximate random populations at high temperature. It is not meant to reflect any physically meaningful system. The outcomes are compared to another calculation (labelled as *Random*), in which the T-site stability was assumed to be identical for all positions.

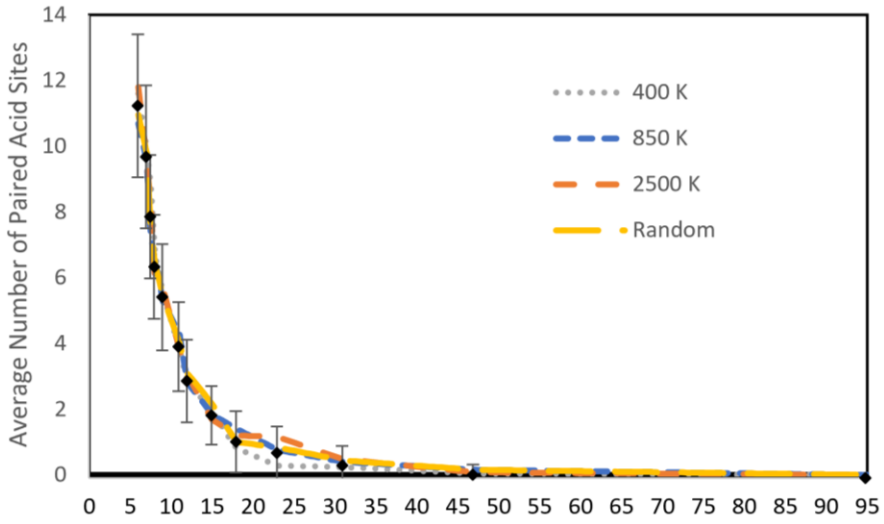


Figure 19: Average number of paired acid sites as function of SAR values at three different simulation temperatures. The error bars are plotted based on the overall average number of paired acid sites from the four sets of simulations.

Previous research groups reported that paired acid sites (Al-O-Si-O-Al sequence) are negligible in Si-rich zeolite frameworks with SAR > 12 [6]. For ferrierites and mordenites, the possibilities of paired acid sites in zeolite framework are very small at an even higher Al concentration when SAR > 8 [6]. Paired acid sites are predominantly formed at ZSM-5 when its framework is close to the limiting value (SAR = 7) [6]. The previous research outcome by Dědeček *et al.*, also predicted that ZSM-5 has paired acid sites when SAR < 12. The result from our calculation is close to the findings from the previous research. In Figure 19, the paired acid sites in ZSM-5 remain non-existent until SAR ≤ 18. The difference between the reported values and the algorithm calculation value may be due to the effect of Al-Al interaction energy (Dempsey's rule). In agreement with these prior studies, we observe a steep slope between SAR of 6 and 9.

c. Average number of paired acid sites as function of Al-Al interaction energies

Previous research shows that the number of paired acid site should be negligible for zeolite structures with SAR > 12 generally [6]. However, our result in Figure 19 shows that there is a small possibility of having paired acid sites in zeolites between SAR of 12 and 45 because the simulation does not take into account of the Al-Al interaction energy. Hence, another set of simulation is done to show that Al-Al interaction energy affects the distribution of acid sites in zeolite structures. A positive Al-Al interaction energy value represents repulsive force and it decreases the number of paired acid sites in zeolite.

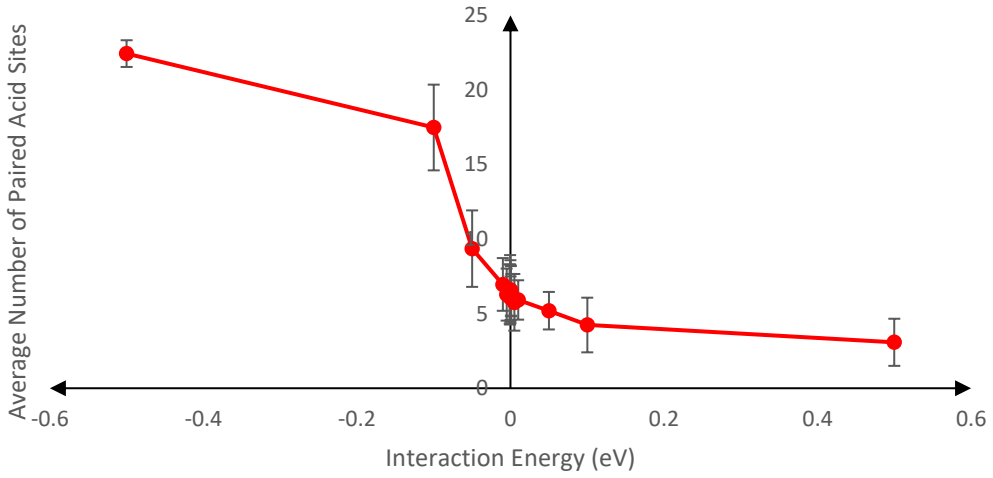


Figure 20: The number of paired acid sites is affected by the Al-Al interaction energy. Positive interaction energy represents repulsive force between Al atoms while negative interaction energy represents attractive force between Al atoms. (Simulation paremeters: SAR = 8, T = 850 K, Number of trials = 25)

This simulation is done by setting three parameters as fixed variables (SAR = 8, T = 850 K and number of trials = 25) and let the Al-Al interaction energy become the manipulated variable, ranging from -0.5 eV to +0.5 eV. Figure 20 shows that the algorithm can calculate the average number of paired acid sites effectively with the consideration of Al-Al interaction energy values. Attractive force (< 0 eV) increases the number of paired acid site and vice versa. The value of Al-Al interaction energy is multiplied by the amount of Al atoms that surrounds a T-site in [-Al-O-Si-O-Al-] manner. For example, if a T-site has initial relative stability of -0.3 eV and two Al atoms are connected in pairs with that T-site while Al-Al interaction energy is set at -0.01 eV, the T-site will have relative stability of -0.32 eV. Dempsey's rule mentioned that Al ions tends to distance themselves away from each other as much as possible. If there is a repulsive force that acts on the Al atoms during the Al/Si substitution process, the average paired acid site should be even

lower than the calculated result in Figure 19. By applying Al-Al interaction energy into the algorithm, generated results will be more accurate.

d. Average number of paired acid sites as function of simulation temperature

To further investigate the temperature dependence of our analysis, we reanalyze the data from Figure 19, and present another comparison highlighting the effect of simulation temperature towards the average number of paired acid sites. Figure 21 shows the average number of paired acid sites against different simulation temperature (400 K, 850 K, 2500 K and *Random* – i.e., the case without consideration of individual T-site stabilities). The *Random* case can be viewed as the trial run with simulation temperature of infinity where the value of probability from Boltzmann energy equation will be equivalent to 1 ($P = e^{\frac{-\Delta E}{RT}} \rightarrow lim\ P = 1$ when $T \rightarrow \infty$). When the probability value is close to 1, all T-sites have the same probability for Al substitution. Hence, as the simulation temperature (400 K, 850 K, 2500 K, ∞) increases, the random case should be recovered.

Based on the graph, there is no clear correlation between simulation temperature and the number of paired acid sites (Figure 21). We would have expected that with increasing temperature the simulations recover the Random case as limiting case. The probability (P) value is within the range of 0.17 (for maximum ΔE) and 1 (for minimum ΔE – i.e., 0) when the simulation temperature is 2500 K. As the temperature increases, the probability will get closer to 1. Surprisingly, we did not find a consistent trend with temperature. Since the occupation statistics in Figure 16 show the anticipated trend, we believe that the energy considerations are correctly captured in the algorithm. The lack of distribution trend with temperature

is therefore attributed to small sample sizes for the extraction of paired acid site statistics.

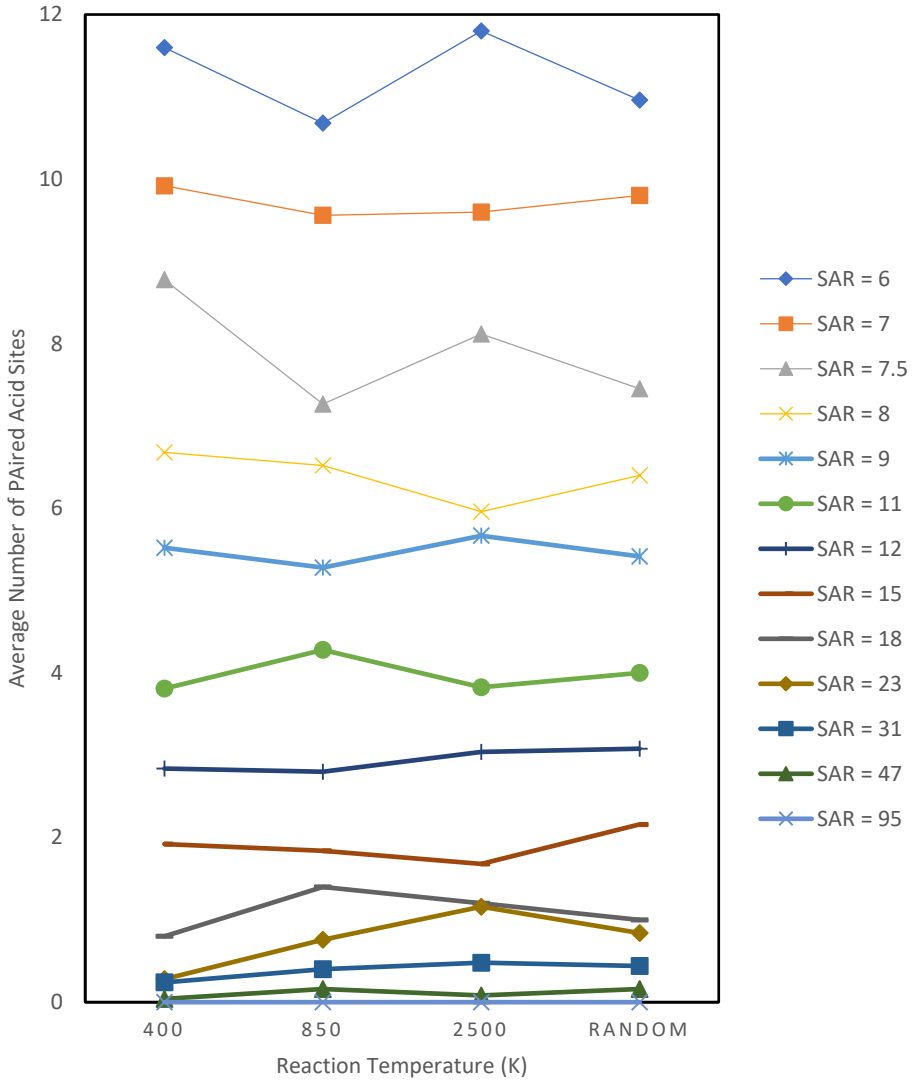


Figure 21: The simulation temperature does not seem to have significant effects on the number of paired acid sites. All the trial runs with SAR = [6, 7, 7.5, 8] are compared separately as function of temperature.

CHAPTER FOUR – SUMMARY AND CONCLUSIONS

The purpose of the algorithm is to estimate the distribution of Al sites in zeolite frameworks, either forming isolated single acid sites or paired acid sites. The equilibrium Al distribution is affected by two factors: the individual site preference for Al substitution and the interaction energy between Al neighbors. The Al-Al interaction energy difference is as significant as the Al/Si substitution energy value because the probability of occurrence decay rapidly with the configuration energy due to the exponential nature of Boltzmann's distribution. The Al-Al interactions are even more significant at lower SAR because more Al ions are incorporated in the zeolite framework [10].

This algorithm enables researchers to specify a few input parameters about the zeolite of their interest and analyze the acid site distribution from the generated plots. The performance and efficiency of the algorithm is tested by running them for a different range of parameters (simulation temperature, SAR, number of framework trials and Al-Al interaction energy values). The parameters of the algorithm will affect the duration of the calculation and the number of effective trials. Two types of methods are used to execute the algorithm. The methods differ in the generation of the initial positions of Al atoms in the Al-substituted zeolite framework.

The traditional Metropolis Monte Carlo Method 1 (MC1) has a lower overall efficiency, despite running faster than the modified Metropolis Monte Carlo Method 2 (MC2). The algorithm with MC1 runs faster for simulations with high SAR (low Al atom content) and high simulation temperature. It is easier to populate a small amount of Al atoms randomly in the zeolite framework without violating Löwenstein's rule. When the simulation is running at a very high temperature, it is easier for Al atoms to move around and substitute

themselves into a new T-site positions. Hence, the duration of calculation will become shorter. If MC1 is executed at a low SAR value, however, the majority of outcomes from the algorithm result in errors for violating Löwenstein's rule and the zeolite framework will not reach the specified SAR value.

MC2 has a higher overall efficiency but a lower calculation speed than MC1. MC2 has an advantage because the design of this method is less prone to errors when the simulation is executed for a low SAR and low temperature values. The results from low SAR simulations are of particular interest, because paired acid sites usually exist in zeolite framework with SAR < 12. Some of the zeolite frameworks will only have paired acid sites at a higher Al atom contents corresponding to SAR < 8. At a lower simulation temperature, the probability of rejecting moves in the Metropolis MC algorithm increases, resulting in a longer execution time. For example, simulations with SAR = 18 are completed in 25 and 15 minutes, when they are carried out at 400 K and 850 K, respectively. The complete algorithm (MC2) is described in Appendix as Algorithm 10.

These two methods (MC1 and MC2) are tested for their reliability and efficiency. Four different sets of simulations are carried out to compare a) the distribution of Al atoms at T-sites with constant and distinct T-sites relative stability, b) the average number of paired acid sites at a range of possible SARs, c) the average number of paired acid sites at a range of Al-Al interaction energies and d) the average number of paired acid sites at different simulation temperatures. Based on the results from first three sets of simulations, the energy part of the code is working efficiently, and the substitution code is comparable to the previous research and literature. The outcome of the third set shows that there is no direct correlation between number of paired acid sites and simulation temperature.

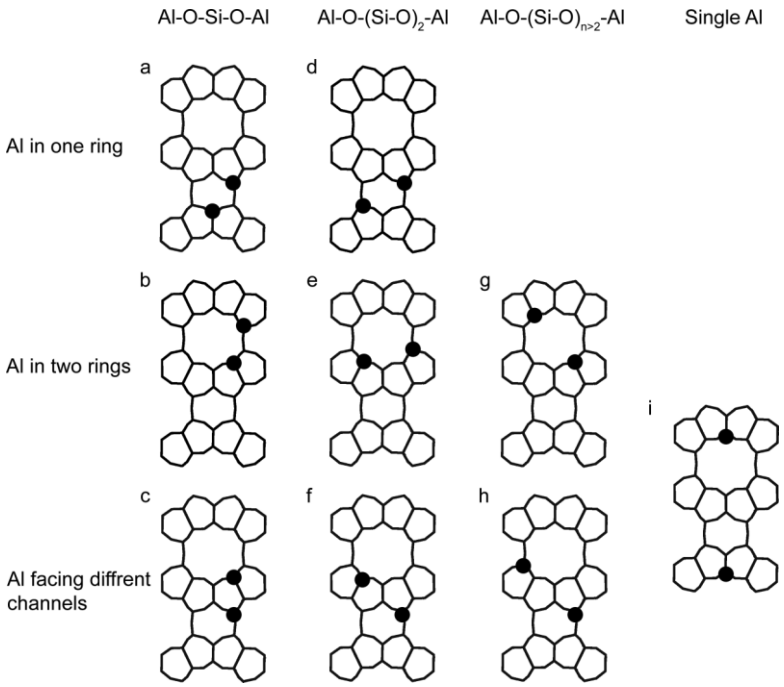


Figure 22: Different types of paired acid sites that are possible in a zeolite framework (Figure Source: Dědeček, J. et al., (2012)) [6].

There are a few ways to optimize and further improve the algorithm to find the Al distribution in a zeolite framework. For example, the iterative part of the algorithm can be optimized by smarter bookkeeping for a faster calculation speed by creating a list of all possible moves to avoid redundant substitution attempts. The algorithm was tested on CHA, MFI and MWW, but can be easily extended to other kinds of zeolite structure frameworks for which relative stability data is available for Al substitution at different T-sites. Interaction parameters for Al-Al interactions were assumed in this work and measured or computed interaction energy values will greatly improve the accuracy of the algorithm. Finally, the definition of paired acid sites can be broadened to include -[-Al-O-Si-O-Si-O-Al-]- motifs shown in Figure 22, which can also constitute a paired site in small and medium pore zeolites [6]. This can be achieved by modifying the cutoff radii d_{min} and d_{max} accordingly.

REFERENCES

[1] N. Brunelli, N. Deshpande and A. Parulkar, "Zeolitic Materials Including Paired Lewis Acid Catalytic Sites," United States of America Patent US20180133700A1, 17 May 2018.

[2] C. Baerlocher, L. B. Mccusker and D. H. Olson, Atlas of zeolite framework types, 6th rev. ed., Amsterdam: Elsevier, 2007.

[3] B. F. Sels and L. M. Kustov, Zeolites and Zeolite-like Materials, 1st ed., Elsevier, 2016.

[4] S. Yang, M. Lach-hab, I. I. Vaisman, E. Blaisten-Barojas, X. Li and V. L. Karen, "Framework-Type Determination for Zeolite Structures in the Inorganic Crystal Structure Database," *Journal of Physical and Chemical Reference Data,* vol. 39, no. 3, p. 033102, September 2010.

[5] M. Niwa, N. Katada and K. Okumura, Characterization and Design of Zeolite Catalysts: Solid Acidity, Shape Selectivity and Loading Properties, Heidelberg ; New York: Springer-Verlag, 2010, p. 184.

[6] J. Dedecek, Z. Sobalík and B. Wichterlová, "Siting and Distribution of Framework Aluminium Atoms in Silicon-Rich Zeolites and Impact on Catalysis," *Catalysis Reviews: Science and Engineering,* vol. 54, no. 2, pp. 135-223, 18 April 2012.

[7] S. Eser, "Bronsted and Lewis Acid Sites," The John A. Dutton e-Education Institute, [Online]. Available: https://www.e-

education.psu.edu/fsc432/content/bronsted-and-lewis-acid-sites. [Accessed 15 February 2020].

[8] R. C. Deka, "Acidity in zeolites and their characterization by different spectroscopic methods," *Indian Journal of Chemical Technology,* vol. 5, pp. 109-123, May 1998.

[9] K. Muraoka, W. Chaikittisilp and T. Okubo, "Energy Analysis of Aluminosilicate Zeolites with Comprehensive Ranges of Framework Topologies, Chemical Compositions, and Aluminum Distributions," *Journal of The American Chemical Society,* vol. 138, no. 19, pp. 6184-6193, 18 May 2016.

[10] A. Ruiz-Salvadora, R. Grau-Crespo, A. Gray and D. Lewis, "Aluminium distribution in ZSM-5 revisited: the role of Al-Al interactions," *Journal of Solid State Chemistry,* vol. 198, pp. 330-336, February 2013.

[11] S. Raychaudhuri, "Introduction To Monte Carlo Simulation," *2008 Winter Simulation Conference,* pp. 91-100, November 2008.

[12] R. Bohinca, J. Hoszowskab, J.-C. Dousse, W. Błachucki, F. Zeeshan, Y. Kayser, M. Nachtegaal, A. B. Pinara and v. B. J. A, "Distribution of aluminum over different T-sites in ferrierite zeolites studied with aluminum valence to core X-ray emission spectroscopy," *Physical Chemistry Chemical Physics,* vol. 19, no. 43, pp. 29271-29277, 11 October 2017.

[13] D. E. Perea, I. Arslan, J. Liu, Z. Ristanovic, L. Kovarik, B. W. Arey, J. A. Lercher, S. R. Bare and B. M. Weckhuysen, "Determining the location and nearest

neighbours of aluminium in zeolites with atom probe tomography," *Nature Communications,* vol. 6, no. 1, p. 7589, 2 July 2015.

[14] C. E. Hernandez-Tamargoetal, A. Roldan and N. H. de Leeuw, "A density functional theory study of the structure of pure-silica and aluminium-substituted MFI nanosheets," *Journal of Solid State Chemistry,* vol. 237, pp. 192-203, 3 February 2016.

[15] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller and E. Teller, "Equation of State Calculation by Fast Computing Machines," *The Journal of Chemical Physics,* vol. 21, no. 6, p. 1087, June 1953.

[16] A. Ghorbanpour, J. D. Rimer and L. C. Grabow, "Periodic, vdW-corrected density functional theory investigation of the effect of Al siting in H-ZSM-5 on chemisorption properties and site-specific acidity," *Catalysis Communication,* vol. 52, pp. 98-102, 5 July 2014.

[17] J. D. Hunter, "Matplotlib: A 2D graphics environment," *Computing in Science & Engineering,* vol. 9, no. 3, pp. 90-95, 2007.

[18] I. Bull, W.-M. Xue, P. Burk, R. S. Boorse, W. M. Jaglowski, G. S. Koermer, A. Moini, J. A. Patchett, J. C. Dettling and M. T. Caudle, "Copper CHA zeolite catalysts," United States Patent US7601662B2, 13 October 2009.

[19] M. Štekrová, M. Kubů, M. Shamzhy, Z. Musilová and J. Čejka, "α-PINENE OXIDE ISOMERIZATION: Role of zeolite structure and acidity in selective synthesis of campholenic aldehyde," *RSC Catalysis Science & Technology,* vol. 8, no. 9, pp. 2488-2501, 2018.

[20] E. Xing, Y. Shi, W. Xie, F. Zhang, X. Mu and X. Shu, "Synthesis, characterization and application of MCM-22 zeolites via a conventional HMI route and temperature-controlled phase transfer hydrothermal synthesis," *Royal Society of Chemistry Advances*, vol. 5, no. 11, p. 8514, 2015.

[21] W. Chu, X. Li, S. Liu, X. Zhu, S. Xie, F. Chen, Y. Wang, W. Xin and L. Xu, "Direct synthesis of three-dimensional MWW zeolite with cyclohexylamine as an organic structure directing agent," *RCS Journal of Material Chemistry A*, vol. 6, no. 26, p. 12244, 2018.

APPENDIX

Algorithm 1: Calculate the numbers of Si atoms and O atoms in a zeolite framework

```
atoms = read('CONTCAR')
countO = 0
countSi= 0
for a in range(0,len(atoms)):
    if symbol[a] == 'Si':
       NcountSi = countSi + 1
       countSi = NcountSi
    if symbol[a] == 'O':
       NcountO = countO + 1
       countO = NcountO
#print 'The number of Si is:', countSi
#print 'The number of O is:', countO
```

Algorithm 2: Generate a list of random integers for Si atoms

```
RanNum = []
while len(RanNum)<countSi-1:
      c = random.randint(countO,countO+countSi-1)
      if c in RanNum:  continue
      else:  RanNum += [c]
      #print 'RanNum = ', RanNum
```

Algorithm 3: Find the four neighboring Si atoms of selected T-site

```
#expansion cell code
#make the unit cell bigger for corner atoms
def findatoms(dmin, dmax, target, Si):
      coord_new = []
      index = []
      x = cell [0,0]
      y0 = cell [0,1]
      z0 = cell [0,2]
      x1 = cell [1,0]
      y = cell [1,1]
      z1 = cell [1,2]
      x2 = cell [2,0]
      y2 = cell [2,1]
      z = cell [2,2]
      for k, i in enumerate(coord):
          if i[0]-Si[0]>x/2:
             i[0]=i[0]-x
             i[1]=i[1]-y0
             i[2]=i[2]-z0
          if Si[0]-i[0]>x/2:
             i[0]=i[0]+x
             i[1]=i[1]+y0
             i[2]=i[2]+z0
          if i[1]-Si[1]>y/2:
             i[1]=i[1]-y
```

```
            i[2]=i[2]-z1
            i[0]=i[0]-x1
         if Si[1]-i[1]>y/2:
            i[1]=i[1]+y
            i[2]=i[2]+z1
            i[0]=i[0]+x1
         if i[2]-Si[2]>z/2:
            i[2]=i[2]-z
            i[1]=i[1]-y2
            i[0]=i[0]-x2
         if Si[2]-i[2]>z/2:
            i[2]=i[2]+z
            i[1]=i[1]+y2
            i[0]=i[0]+x2
        coord_new.append(i)
    for k,at in enumerate(coord_new):
#double asterisk means exponentiation operator
         d = np.sqrt((at[0]-Si[0])**2+(at[1]-Si[1])**2+(at[2]-
                Si[2])**2)
         if d>dmin and d<dmax: index.append(k)
    return (index, coord_new)
```

Algorithm 4: Substituting Si atoms with Al atoms without consideration of energetics

```
Al1=RanNum[0]
positions = [Al1]
Si1 = coord[Al1]
(index1,coord1) = findatoms(dmin,dmax,'Si',Si1)
neighbors= neighbors + index1
Realneighbors=neighbors + positions
RestRanNum = [i for i in RanNum if i not in Realneighbors]

#always start with 0
for a in range(0,len(RestRanNum)-1):
    while len(positions)<number_Al:
         Al2 = RestRanNum[a]
         positions += [Al2]
         Si1 = coord[Al2]

#prevent neighbouring atoms from being substituted
         (index1,coord1) = findatoms(dmin,dmax,'Si',Si1)
         neighbors = neighbors + index1
         Realneighbors = neighbors + positions
         RestRanNum = [i for i in RestRanNum if i not in Realneighbors]
for d in positions:
    Si1 = coord[d] #coordinate of Substituted Si
    atoms.numbers[d] = 13  #substitute Si with Al atom
    (index1,coord1) = findatoms(dmin,dmax,'Si',Si1)
print 'The neighboring Si of atom #', c, 'they are', index1
print 'neighbors = ', neighbors
print 'The substituted Si are', positions
```

Algorithm 5: Considering energetic in Al/Si substitution step

```
for a in range(0,len(RestRanNum)-1):
    while len(positions)<number_Al:
          Al2 = RestRanNum[a]
          print Al2
          Tsite_Number=get_key(Al2)
          print Tsite_Number
          Energy=get_energy(Tsite_Number)
          print Energy
          Rconstant = 0.00008617333262 #eV/K #8.314 J/mol/K
          P = math.exp(-Energy/(Rconstant*Tnumber))
          print P

          #get a random number for Monte Carlo part:
          R = np.random.random(1)[0]
          print R
          if P>=R:
             print 'yes'
             positions += [Al2]
             Si1 = coord[Al2]
```

Algorithm 6: Finding the number tags for each distinct T-site

```
#take the identified T-sites files
#import ASE to able to view zeolites

from ase import Atoms,Atom
from ase.io import read,write
from ase.visualize import view

atoms = read('CONTCAR')

symbol = atoms.get_chemical_symbols()
coord = atoms.get_positions()
index = atoms.get_atomic_numbers()
for a in range(0,len(atoms)):
    if symbol[a] == 'Al':
       print a
```

Algorithm 7: Applying energetics into the whole algorithm

```
#import excel file that has relative stability
pathway = os.getcwd()
f = open('H-ZSM-5.txt', 'r')
line = f.readlines()[:]

T_subenergy={}
with open("H-ZSM-5.txt") as f:
     for line in f:
         (key,val) = line.split()
         T_subenergy[int(key)]=float(val)

#create a dictionary for energy of T-sites
def createList(r1, r2):
```

```
    return np.arange(r1, r2+1, 1)
r1 = 1
r2 = 13 #for 12 T-sites (+1 for dummy file)

#use an extra dummy tsite file
Tsite_lists ={}
numbers=createList(r1,r2)

for a in range(r1,r2):
    file='T%d.txt'%(a)
    txt=open(file,'r')
    t1=txt.read().split('\n')
    #remove extra spacing
    y=''
    t1.remove(y)
    #remove single quote from the list
    t1=map(int,t1)
    Tsite_lists[numbers[a-1]]=t1

# function to return key for any value
def get_key(val):
    for key, value in Tsite_lists.items():
         if val in value:
             return key

    return "key doesn't exist"

# function to return value for any key
def get_energy(key):
          for keys, value in T_subenergy.items():
        if key == keys:
             return value

    return "key doesn't exist"
```

Algorithm 8: Presenting the data in form of histogram

```
from matplotlib import pyplot as plt

#create histogram

bins = np.linspace(0, max(graphing_counter),max(graphing_counter)+1)
plt.hist(graphing_counter, bins=bins, histtype='bar', rwidth=0.8,
 edgecolor='black',linewidth=1.0, alpha=0.7)
plt.title('The paired acid site distribution with SAR = {}'.format(SAR))
plt.ylabel('Trial count')
plt.xlabel('Number of paired acid sites')
plt.show()
```

Algorithm 9: Running the substitution energy code for 200 iterations per number of Al atom

```
if len(initial_positions) == number_Al:
   iteration = 0
   while iteration < number_Al*200:
         Al_old = random.choice(positions)
         Al_new = random.choice(RestRanNum)
         CoO = coord[Al_old]
         Co = coord[Al_new]
         (indexO, coordO) = findatoms(dmin,dmax, 'Si',CoO)
         (index4,coord4) = findatoms(dmin,dmax, 'Si',Co)
         if Al_new in neighbors:
            continue
         if Al_new not in neighbors:
            Tsite_Number_old = get_key(Al_old)
            Energy_old = get_energy(Tsite_Number_old)
            Tsite_Number_new = get_key(Al_new)
            Energy_new = get_energy(Tsite_Number_new)
            Rconstant = 0.00008617333262 #eV/K #8.314 J/mol/K

   #if the relative stability of new positions is smaller than
    old positions, this part of the code applies
            if Energy_new <= Energy_old:
               positions.remove(Al_old)
               positions += [Al_new]
               forbidden = []
               RestRanNum.remove(Al_new)
               RestRanNum += [Al_old]
               for i in positions:
                   CoX = coord[i]
                   (indexx,coordx)=
                           findatoms(dmin,dmax,'Si',CoX)
                   indexx = [i for i in indexx if i not in
                            forbidden]
                   forbidden = forbidden + indexx
               neighbors = forbidden
               iteration = iteration + 1
               continue

            if Energy_new > Energy_old and Pro>=Rnumber:
               positions.remove(Al_old)
               positions += [Al_new]
               RestRanNum.remove(Al_new)
               RestRanNum += [Al_old]
               forbidden = []
               for i in positions:
                   CoX = coord[i]
                   (indexx,coordx) =
                           findatoms(dmin,dmax,'Si',CoX)
                   indexx = [i for i in indexx if i not in
                            forbidden]
                   forbidden = forbidden + indexx
               neighbors = forbidden
               iteration = iteration + 1
               continue
```

Algorithm 10: The complete code for MC2

```
#import ASE to view zeolites

from ase import Atoms,Atom
from ase.io import read,write
from ase.visualize import view
from collections import OrderedDict
import random
import numpy as np
import os
import math
from matplotlib import pyplot as plt
from collections import Counter
import sys

#get .cif or CONTCAR file
atoms = read('CONTCAR')
symbol = atoms.get_chemical_symbols()
pathway = os.getcwd()
center = atoms.get_cell().sum(axis=0) / 2
cell = atoms.get_cell()
coord = atoms.get_positions()

dmin = 2.8  # Minimum Si-Si bond length
dmax = 3.4  # Maximum Si-Si bond length

#calculate atoms number code
countO = 0
countSi = 0
countAl = 0
for a in range(0,len(atoms)):
    if symbol[a] == 'Si':
       NcountSi = countSi + 1
       countSi = NcountSi
    if symbol[a] == 'O':
       NcountO = countO + 1
       countO = NcountO
    if symbol[a] == 'Al':
       NcountAl = countAl + 1
       countAl = NcountAl
#print 'The number of Si is:', countSi
#print 'The number of O is:', countO
#print 'The number of Al is:', countAl

#expansion cell code #make the unit cell bigger for corner atoms
def findatoms(dmin, dmax, target, Si):
      coord_new = []
      index = []
      x = cell [0,0]
      y0 = cell [0,1]
      z0 = cell [0,2]
      x1 = cell [1,0]
      y = cell [1,1]
      z1 = cell [1,2]
```

```
      x2 = cell [2,0]
      y2 = cell [2,1]
      z = cell [2,2]
      for k, i in enumerate(coord):
          if i[0]-Si[0]>x/2:
             i[0]=i[0]-x
             i[1]=i[1]-y0
             i[2]=i[2]-z0
          if Si[0]-i[0]>x/2:
             i[0]=i[0]+x
             i[1]=i[1]+y0
             i[2]=i[2]+z0
          if i[1]-Si[1]>y/2:
             i[1]=i[1]-y
             i[2]=i[2]-z1
             i[0]=i[0]-x1
          if Si[1]-i[1]>y/2:
             i[1]=i[1]+y
             i[2]=i[2]+z1
             i[0]=i[0]+x1
          if i[2]-Si[2]>z/2:
             i[2]=i[2]-z
             i[1]=i[1]-y2
             i[0]=i[0]-x2
          if Si[2]-i[2]>z/2:
             i[2]=i[2]+z
             i[1]=i[1]+y2
             i[0]=i[0]+x2
          coord_new.append(i)
      for k,at in enumerate(coord_new):
          d = np.sqrt((at[0]-Si[0])**2+(at[1]-Si[1])**2+(at[2]-
Si[2])**2)
          if d>dmin and d<dmax: index.append(k)
      return (index, coord_new)

#import excel file that has relative stability
pathway = os.getcwd()
f = open('H-ZSM-5.txt', 'r')
line = f.readlines()[:]

T_subenergy={}
with open("H-ZSM-5.txt") as f:
     for line in f:
         (key,val) = line.split()
         T_subenergy[int(key)]=float(val)

#create a dictionary for energy of T-sites
def createList(r1, r2):
    return np.arange(r1, r2+1, 1)
r1 = 1
r2 = 13
#use an extra dummy tsite file
Tsite_lists ={}
numbers=createList(r1,r2)
```

```
for a in range(r1,r2):
    file='T%d.txt'%(a)
    txt=open(file,'r')
    t1=txt.read().split('\n')
    #remove extra spacing
    y=''
    t1.remove(y)
    #remove single quote from the list
    t1=[int(x) for x in t1]
    Tsite_lists[numbers[a-1]]=t1

# function to return key for any value
def get_key(val):
    for key, value in Tsite_lists.items():
         if val in value:
             return key
    return "key doesn't exist"

# function to return value for any key
def get_energy(key):
    for keys, value in T_subenergy.items():
        if key == keys:
             return value
    return "key doesn't exist"

#user input code
SAR=input("The current zeolite framework is ZSM-5. Range for SAR is
between 6 and 95. Please enter SAR: ")
FWnumber=input ("Please enter number of trials for this chosen
frameworks: ")
Tnumber=input ("Please enter the temperature in Kelvin for the Aluminum
Substitution Process (should be more or equal to 400K): ")
int_energy =input("Please enter a value for Al-Al interaction energy")
SAR=float(sys.argv[1])
FWnumber=int(sys.argv[2])
Tnumber=int(sys.argv[3])
int_energy=float(sys.argv[4])

#substitution code abides to Lowenstein rule

trials = 0
graphing_counter = []
graph = []
while trials <= FWnumber-1:
      atoms = read('CONTCAR')
      countO = 0
      countSi= 0
      for a in range(0,len(atoms)):
          if symbol[a] == 'Si':
             NcountSi = countSi + 1
             countSi = NcountSi
          if symbol[a] == 'O':
             NcountO = countO + 1
             countO = NcountO
      number_Al=int(countSi/(SAR+1))
```

```
        coord=atoms.get_positions()

        positions =[]
        neighbors = []
        RanNum = []
        Assigned = []
        while len(RanNum)<countSi:
              c = random.randint(countO,countO+countSi-1)
              if c in RanNum:  continue
              else:  RanNum += [c]
        RestRanNum = RanNum

        Al0 = RestRanNum[0]
        Tsite_Number=get_key(Al0)
        positions += [Al0]
        Co0 = coord[Al0]
        #prevent neighbouring atoms from being substituted
        (index1,coord1) = findatoms(dmin,dmax,'Si',Co0)
        #neighbors:silicons due to lowenstein rule
        neighbors = neighbors + index1

        #Assigned:  all elements that are already assigned to silicon or
        aluminum
        Assigned = neighbors + positions
        #RestRanNum = [i for i in RestRanNum if i not in Assigned]
        a=a+1

        #substitute the number of Al until SAR is reached
        potentialAl = []
        oldpotentialAl = []

        run = 0
        while len(positions)<number_Al and run<200:
           for i in range(0, len(neighbors)):
               print (number_Al)
               Si1 = neighbors[i]
               Co1 = coord[Si1]
               (index2,coord2) = findatoms(dmin,dmax,'Si',Co1)
               index2 = [i for i in index2 if i not in Assigned]
               potentialAl = potentialAl + index2
               run = run + 1
#put in code for checking neighbor if they are aluminum
               for s in range(0, len(index2)):
                   Co3 = coord[index2[s]]
                   (index3,coord3) = findatoms(dmin,dmax,'Si',Co3)
                    index3 = [i for i in index3 if i not in neighbors]
                    neighbors = neighbors + index3
                    index3 = [i for i in index3 if i not in Assigned]
                    potentialAl = [i for i in potentialAl if i not in
                   neighbors]
                    potentialAl = [i for i in potentialAl if i not in
                   positions]
                    NewAl = list(OrderedDict.fromkeys(potentialAl))
                    potentialAl = NewAl
            NewAl=list(OrderedDict.fromkeys(potentialAl))
```

```
            positions = positions + NewAl
            NoRepeat = list(OrderedDict.fromkeys(positions))
            positions = NoRepeat
            RestRanNum = [i for i in RestRanNum if i not in positions]
            Assigned = neighbors + positions
            while len(positions)>number_Al:
                  RestRanNum += positions[-1:]
                  positions = positions [:-1]

#this code considers the energetics but the substitution trials will
only run for 100 times
      if len(positions)<number_Al:
         print ("The current trial does not meet SAR. Please try
         again.")
      initial_positions = positions

      for x in initial_positions:
          Si1 = coord[x] #coordinate of Substituted Si
          atoms.numbers[x] = 13  #substitute Si with Al atom

      #create POSCAR
      if len(initial_positions) == number_Al:
         trials = trials + 1
         file = 'POSCAR_INI%d'%(trials)
         pathway = os.getcwd()
         write(os.path.join(pathway,file),atoms)

      #make Al atoms back to Si atoms
      for x in initial_positions:
          Si1 = coord[x] #coordinate of Substituted Si
          atoms.numbers[x] = 14  #substitute Si with Al atom

      if len(initial_positions) == number_Al:
         iteration = 0

         if SAR > 12: totaliterate = 200
         if SAR<= 12: totaliterate = 400

         realiteration = 0
         while iteration < number_Al*totaliterate:
               Al_old = random.choice(positions)
               Al_new = random.choice(RestRanNum)
               Co = coord[Al_new]
               CoO = coord[Al_old]

               (index4, coord4) = findatoms(dmin,dmax, 'Si',Co)
               (indexO, coordO) = findatoms(dmin,dmax, 'Si',CoO)

               if Al_new in neighbors:
                  iteration += 1
                  continue

               if Al_new not in neighbors:
                  Tsite_Number_old = get_key(Al_old)
                  Energy_old = get_energy(Tsite_Number_old)
```

```
                    Tsite_Number_new = get_key(Al_new)
                    Energy_new = get_energy(Tsite_Number_new)
                    Rconstant = 0.00008617333262 #eV/K #8.314 J/mol/K

# this part of the code is to check if they are paired acid sites
# optional user input for attractive or repulsive force
                    #negative - attractive; positive - repulsive
                    #count number of neighbors for old
                    o_neighbors = []
                    for e in indexO:
                        Coo = coord[e]
                        (ind_old,coo_old)=findatoms(dmin, dmax, 'Si', Coo)
                        for i in ind_old:
                            o_neighbors += [i]
                    #take out repeats
                    old_neigh = []
                    [old_neigh.append(x) for x in o_neighbors if x not in
                    old_neigh]
                    dupl = [x for x in old_neigh if x in positions]
                    num_neigh = len(dupl)
                    Energy_old += num_neigh * int_energy

                #count number of neighbors for new

                    n_neighbors = []
                    for g in index4:
                        Cooo = coord[g]
                        (ind_new,coo_new)=findatoms(dmin, dmax, 'Si',
                        Cooo)
                        n_neighbors += [ind_new]
                        for i in ind_new:
                            n_neighbors += [i]
                #take out repeats
                    new_neigh = []
                    [new_neigh.append(x) for x in n_neighbors if x not in
                    new_neigh]
                    duplicate = [x for x in new_neigh if x in positions]
                    num_neigh2 = len(duplicate)
                    Energy_new += num_neigh2 * int_energy
                    #print num_neigh

    ###### start of the interaction energy part of the code ######

                    Pro = math.exp(-((float(Energy_new)-
                    float(Energy_old))/(float(Rconstant)*float(Tnumber))))
                    Rnumber = np.random.random()

#if the relative stability of new posiions is smaller than old postions,
this part of the code applies.

                    if Energy_new <= Energy_old:
                       positions.remove(Al_old)
                       positions += [Al_new]
                       forbidden = []
                       RestRanNum.remove(Al_new)
```

```
                        RestRanNum += [Al_old]
                        for i in positions:
                            CoX = coord[i]
                            CoX = coord[i]
                            (indexx,coordx) = findatoms(dmin, dmax, 'Si',
                            CoX)
                            indexx = [i for i in indexx if i not in
                            forbidden]
                            forbidden = forbidden + indexx

                        neighbors = forbidden
                        iteration = iteration + 1
                        realiteration +=1
                        print (iteration)
                        continue

                    if Energy_new > Energy_old and Pro>=Rnumber:
                        positions.remove(Al_old)
                        positions += [Al_new]
                        RestRanNum.remove(Al_new)
                        RestRanNum += [Al_old]
                        forbidden = []
                        for i in positions:
                            CoX = coord[i]
                            (indexx,coordx) = findatoms(dmin, dmax, 'Si',
                            CoX)
                            indexx = [i for i in indexx if i not in
                            forbidden]
                            forbidden = forbidden + indexx
                        neighbors = forbidden
                        iteration = iteration + 1
                        realiteration += 1
                        print (iteration)
                        continue

            dummy = positions
            count_pair = 0
            #calculate the amount of paired acid site in a trial
            for z in positions:
                for y in dummy:
                    corz = coord[z]
                    (ind,cor) = findatoms(dmin,dmax, 'Si', corz)
                    cory = coord[y]
                    (IND,COR) = findatoms(dmin,dmax, 'Si', cory)

                    if any(i in ind for i in IND) == True and y != z:
                       count_pair = count_pair + 1
            real_pairs = int(count_pair/2)

            for d in positions:
                Si1 = coord[d] #coordinate of Substituted Si
                atoms.numbers[d] = 13  #substitute Si with Al atom
            print ('The substituted Si are', positions)

    #check if the sustitution energy part of the code is complete or not
```

```
#if realiteration is equal to zero, that means that the initial POSCAR
Al atoms has not been moving around

      #create POSCAR
      if len(positions) == number_Al and realiteration >=1:
         #count how many paired acid sites
         graphing_counter += [real_pairs]
         file = 'POSCAR_FIN%d'%(trials)
         pathway = os.getcwd()
         write(os.path.join(pathway,file),atoms)

######################################################################

#create graph for average number of Al atoms against T-site numbers
#this code can be modified to create graph of Trial counts against #
paired acid sites

         for i in positions:
             Tsite_Number_graph = get_key(i)
             graph += [Tsite_Number_graph]

c = Counter(graph)
plt.bar(c.keys(), c.values())
plt.xlim(0,13)
plt.title('T-site distribution with SAR = {}'.format(SAR))
plt.ylabel('count')
plt.xlabel('T-site')
plt.show()
plt.savefig('graph-count.png',format='png')

######################################################################

def Average(List):
    return sum(List) / len(List)
print (graphing_counter)
average = Average(graphing_counter)
print('Average of the paired acid sites =', round(average,3))
Ave=str(round(average,3))
f = open("Average.txt",'w')
f.write(str(graphing_counter))
f.write('\n')
f.write(Ave)
f.close()

######################################################################
```

Table 3: Data of average paired acid sites for ZSM-5 zeolite at different SARs and temperatures (based on MC2 method) – Results are plotted in Figure 17.

		Simulation Temperature (K)			
SAR	Number of Al atoms	400	850	2500	Random
6	13	11.6	10.7	11.8	11.0
7	12	9.9	9.6	9.6	9.8
7.5	11	8.8	7.3	8.1	7.5
8	10	6.7	6.5	6.0	6.4
9	9	5.5	5.3	5.7	5.4
11	8	3.8	4.3	3.8	4.0
12	7	2.8	2.8	3.0	3.1
15	6	1.9	1.8	1.7	2.2
18	5	0.8	1.4	1.2	1.0
23	4	0.3	0.8	1.2	0.8
31	3	0.2	0.4	0.5	0.4
47	2	0.0	0.2	0.1	0.2
95	1	0.0	0.0	0.0	0.0