



Molecular Geometry Optimisation Using Evolutionary Computation

Dissertation Project

2020/2021

Developer: Sophie Turner

110599533

BSc (Hons) Computer Science

Supervisor: Dr. David Walker

Contents

| | |
|--|----|
| Abstract..... | 3 |
| Introduction | 3 |
| Links to Work | 3 |
| Background | 3 |
| Aim | 4 |
| Literature and Software Review | 5 |
| Project Plan | 9 |
| Roadmap & Sprint Plans | 9 |
| Risk Assessment | 10 |
| Legal, Social, Ethical and Professional Matters..... | 11 |
| Implementation | 12 |
| Technologies Used | 12 |
| Sprint Zero – Planning..... | 13 |
| Sprint One – Building a Molecule..... | 13 |
| Sprint Two – Predicting a Shape | 16 |
| Sprint Three – Other Algorithm | 17 |
| Sprint Four – Viewing Results | 19 |
| Sprint Five – Testing..... | 23 |
| Sprint Six – Interactivity | 24 |
| Sprint Seven – Testing..... | 25 |
| Sprint Eight – Final Touches..... | 25 |
| Sprint Nine – Showcase Materials and Report | 26 |
| Discussion & Analysis | 27 |
| Analysis of Many-Molecule Evolutionary Algorithm | 27 |
| Analysis of Per-Atom Exhaustive Test..... | 30 |
| Measurements and Comparisons | 31 |
| Reflection on the Project | 35 |
| Conclusions | 36 |
| Suggestions for Improvement..... | 36 |
| References | 38 |
| Appendices..... | 40 |
| Appendix A – Roadmap..... | 40 |
| Appendix B – Questionnaire Information Sheet..... | 43 |
| Appendix C – Target User Questionnaire..... | 45 |

| | |
|--|----|
| Appendix D – Participant A’s Responses to Target User Questionnaire..... | 49 |
| Appendix E – Participant B’s Responses to Target User Questionnaire | 52 |
| Appendix F – Project Initiation Document..... | 54 |
| Appendix G – Performance Comparison..... | 56 |
| Appendix H – First Usability Test Questionnaire | 58 |
| Appendix I – Participant C’s Responses to First Usability Test Questionnaire | 60 |
| Appendix J – Participant D’s Responses to First Usability Test Questionnaire..... | 62 |
| Appendix K – Participant E’s Responses to First Usability Test Questionnaire | 63 |
| Appendix L – User Guide..... | 64 |
| Appendix M – Poster..... | 67 |

Word count: 10,991.

Abstract

The aim of this project was to create an evolutionary algorithm to find energetically optimal arrangements of atoms in space. The evolutionary algorithm was compared to an exhaustive algorithm. Both the algorithms were able to reduce the energies of systems. The evolutionary algorithm was faster to execute than the exhaustive algorithm for 92 % of molecules and it produced more optimal structures for 70 % of systems, although the exhaustive algorithm also optimised structures to varying extents. Execution times of both the algorithms were improved without worsening the results obtained, using a combination of code optimisations, multiprocessing and algorithm re-design. Although the energy values were improved, the program was usually unable to accurately predict molecular geometry to provide chemically reasonable shapes due to over-simplifications and a lack of data used in energy calculations.

Introduction

Links to Work

GitHub repository: <https://github.com/Squidgeypea/SophieCOMP3000/tree/main>

Demonstration of the program at sprint five: <https://youtu.be/ylv4J85m95Y>

Demonstration of the program at sprint seven: <https://youtu.be/llvHbYyEO6Q>

Demonstration of the program later in sprint seven: <https://youtu.be/irX853Cr7zk>

Demonstration of the final product: <https://youtu.be/6q2dhAwMcps>

Microsoft Planner: <https://tasks.office.com/live.plymouth.ac.uk/en-GB/Home/Planner/#/plantaskboard?groupId=94cc8cbf-c90e-473e-a3f2-7d7d5dee52d9&planId=VmtMkciqc0GG6F--BwFrqpYAESf1>

Background

The purpose of energy optimisation in molecular geometry is to find the most probable configuration of a molecule or other system as it occurs in the real world. This allows scientists to make more accurate calculations and design more efficient chemical processes for industrial uses. The process can be for intermolecular as well as intramolecular bonds or forces. This project focussed on intramolecular situations in a single molecule.

By considering the wave-like properties of electrons, the Schrödinger equation could theoretically give accurate results for the energy of a system, but it is not possible to obtain a precise value for a system of many particles from the Schrödinger equation partly due to the movement and repulsion of electrons. Furthermore, the computational expense required to calculate all possible solutions using this method would be unreasonable as it has nondeterministic polynomial (NP) time complexity and, therefore, it is believed that not even a supercomputer would perform this task in a reasonable amount of time, as explained by Matthews et al. (2020). For this reason, computer programs must attempt to estimate the correct values by making assumptions about the system in order

to simplify the calculation methods and by finding alternative formulae to approximate values of energy, force and other properties. The output of the algorithm can be compared to the known shape, structure and total potential energy of the molecule to determine its accuracy.

Approximations of a system's energy can be gained using various computational techniques, the most popular and successful currently being density functional theory (DFT). A detailed explanation of this is beyond the scope of this project. The main advantages of DFT from a practical point of view are that it can be executed on an ordinary desktop computer and can be used for many different types of molecules, unlike most other methods, such as the effective medium theory (EMT) calculations used in Geopt, which are only accurate for certain metal structures. It was suspected that Geopt could have been improved if DFT had been used, as shown by Kitchen (2012), a developer of DFT software who demonstrated its use in a similar project. Details of this situation are provided in the discussion and analysis section of this report.

Evolutionary algorithms (EAs) are nature-inspired, heuristic techniques to solve problems. The underlying concept is that a population of potential solutions undergoes a reproduction process based on those seen in the natural world. Randomness is important for EAs because it is needed to create potential solutions with unique traits or variables, according to Soni (2018). The type of EA used for Geopt was a genetic algorithm (GA). GAs are EAs which are designed to suggest approximate solutions to optimisation problems which are difficult or impossible to solve using classical computation, such as those with NP complexity. During the progression of the algorithm, the population of potential solutions is subjected to a process akin to natural selection which encourages certain individuals to reproduce based on a measurement of fitness to solve the problem in question. The aim is that, over time, the population will evolve and create better, more optimal solutions. The amount of randomness applied to each new individual via mutation of its defining variables, crossover between parents' variables, or other methods must be carefully chosen. Difficulty with this was experienced during this project, as too much randomness offset any evolutionary advantages, and too little randomness led the EA to become trapped in a local minimum and not find a satisfactory solution. GAs generally take the following form, after the generation of an initial population:

1. Test the fitness of each solution.
2. Select the fittest solution(s) to reproduce.
3. Create offspring with random mutations from the selected parent(s).
4. Place the offspring in the population and, optionally, remove others.
5. Repeat the steps until convergence, or until some stopping criteria are met.

Researchers, including Grumbling and Horowitz (2019), anticipate that quantum computing may hold the key to precisely solving scientific problems of the kind that EAs and supercomputers are currently used to approximate, although the quantum technology required to do this is not yet available as it is still being developed from first principles.

This report refers to distances between atoms rather than bond lengths because the calculation methods used in this project do not take into account information about the existence, types or locations of bonds present in the molecule. This report refers to the ground state total potential energy of configurations as their 'energy', and pseudo-randomness as 'randomness' for the purpose of concision.

Aim

The aim of this project was to create an interactive EA to predict the geometric structure of a molecule or system of atoms, based only on an estimate of the system's total potential energy. This was a one-objective problem. The

objective was to find the arrangement of atoms, and therefore their bonds, where the net force on each atom was as close to zero as possible.

Literature and Software Review

Software which performed similar tasks included PubChem, Avogadro, Biovia, ChemDraw and GaussView, programs designed for modelling and visualising molecules. This software revealed some requirements, such as visualisation of the structure as an image and the option for the user to specify the desired chemical formula. Additionally, some undesirable features were presented by the software; some of the user interfaces (UIs) had a large number of unexplained options on the screen which could be overwhelming or confusing to the user. Not all of these programs offered the user the option of geometry optimisation. GaussView and Avogadro provided their own closed-source interactive EAs for geometry optimisation. GaussView allowed users to build an initial molecule by selecting templates and joining functional groups. Avogadro allowed users to build any molecule whether chemically realistic or not. It then allowed users to optimise the geometry of the molecule but did not display a view of the optimised molecule. Instead of treating geometry as an optimisation problem, PubChem kept a database of structures and attempted to match the selected molecule to the most probable geometric structure when searching the database. Like GaussView, PubChem provided a list of templates for a molecule to be built from functional groups. Unlike the other software, PubChem ran in the browser rather than as a downloadable application. Examples of the UIs are given in figures one and two.

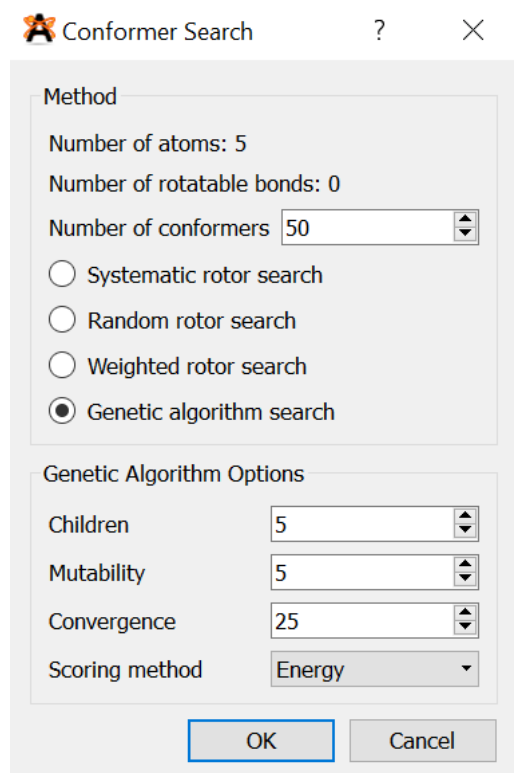


Figure one – Avogadro's geometry optimisation UI.

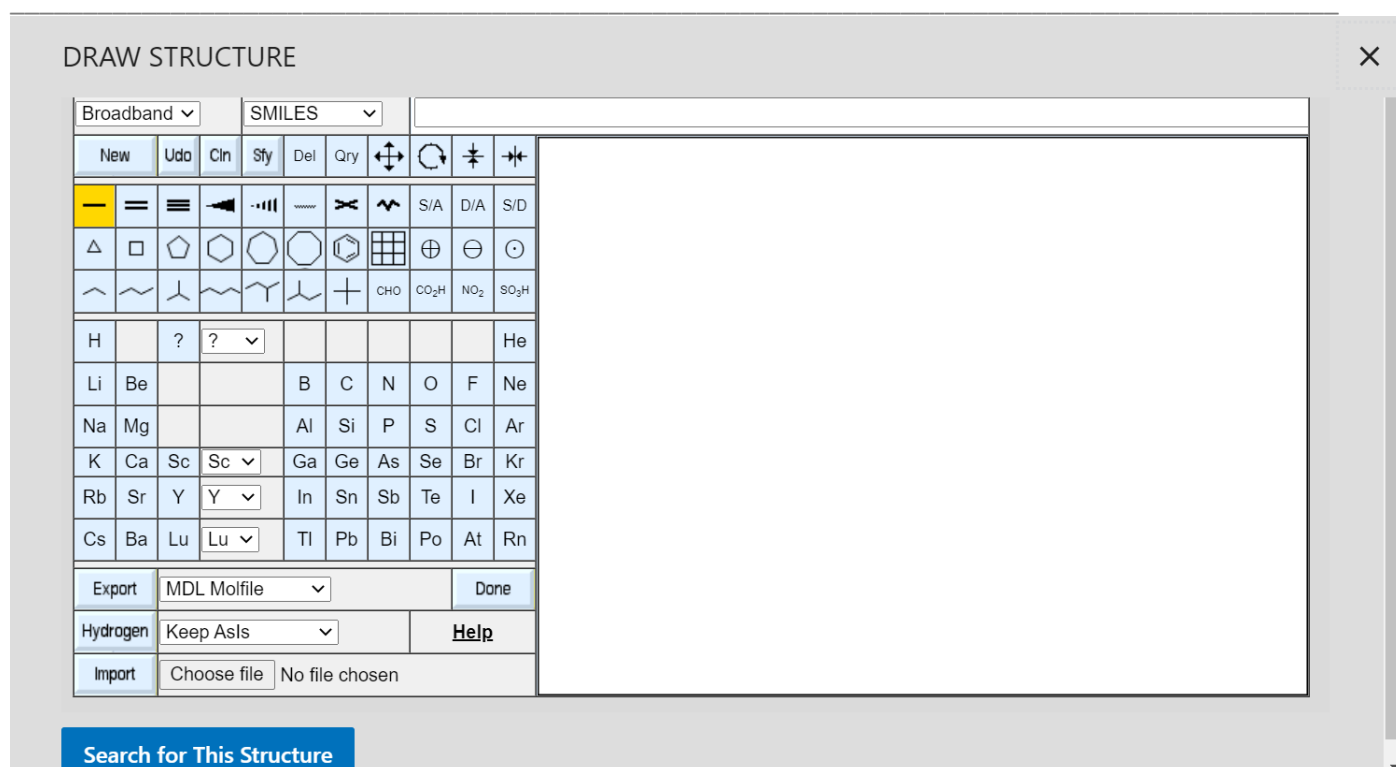


Figure two – Building a molecule from common groups in PubChem.

Scientists have had greater success designing algorithms to optimise a single type of molecule than generalising a particular model to all chemicals. Johnston et al. (2002) created a GA to find aluminium crystal structures. The flow chart for this algorithm is shown in figure three. Figure four displays a similar type of algorithm, created by Chen et al. (2007) to work with DFT to optimise clusters of a specific theoretical molecule. Figure five presents another such algorithm, from Mohn (2016), whose goal was to predict cation ordering in another specific target molecule and was also aided by DFT. These similar algorithm designs reveal that they all used mutation operations for new generations and Mohn's (2016) algorithm performed crossover on a population of only two solutions. Although it was not expected that Geopt would be able to generalise well to a greater range of molecules than those studied in the literature, the option was provided as a proof of concept.

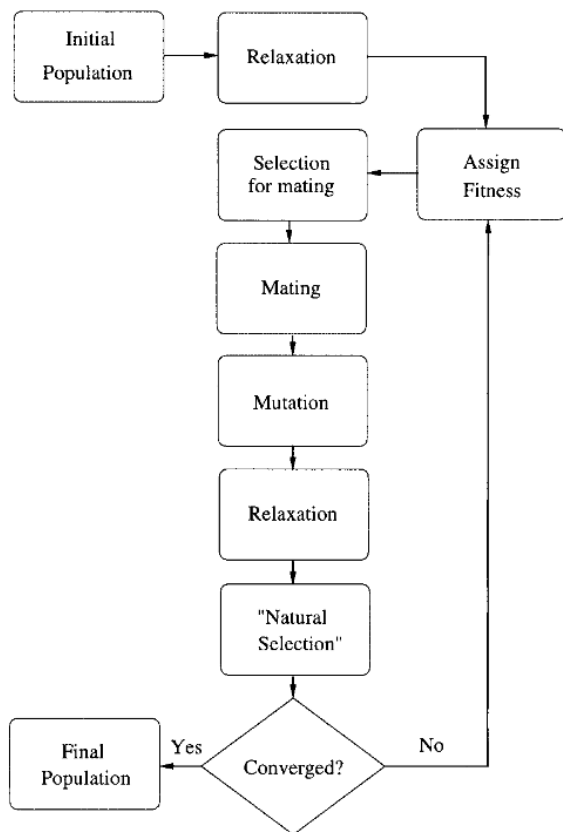


Figure three – Flow chart of GA by Johnston et al. (2002).

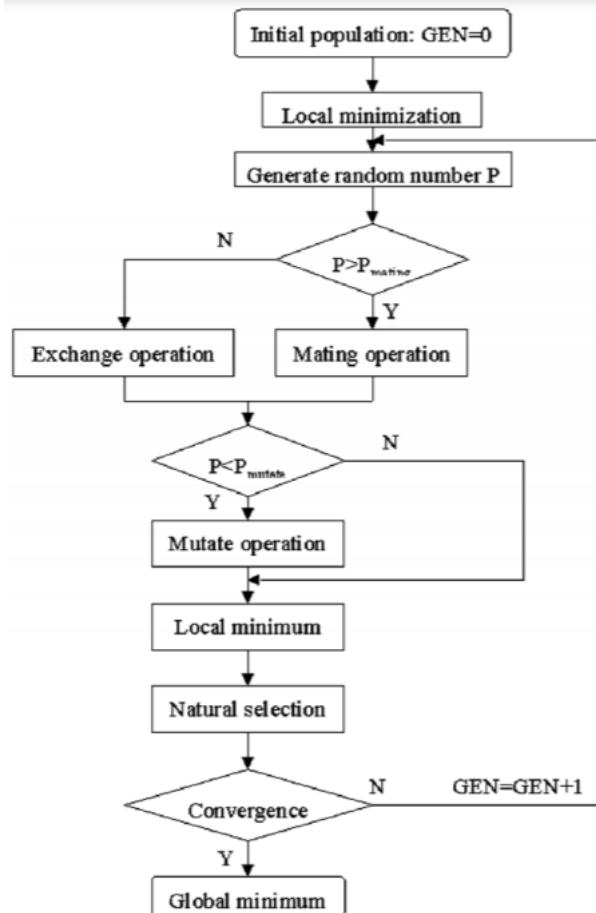


Figure four – Flow chart of GA by Chen et al. (2007).

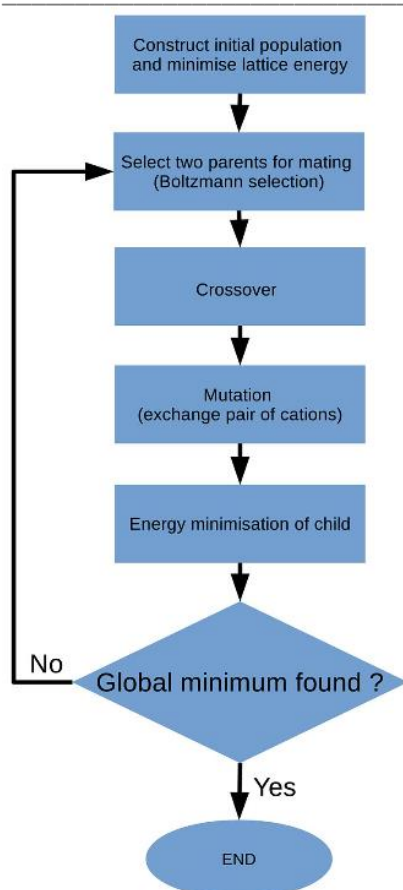


Figure five – Flow chart of GA by Mohn (2016).

Hashimoto et al. (2015) reviewed the choice of programming languages for use in high-performance quantum chemistry programs and chose Python as their first choice. The researchers concluded that dynamic scripting languages may have been preferable to compiled languages such as C for this task, mainly due to the wide array of libraries available and the simplicity for the programmer when using a popular implementation such as Python. This led to the choice of Python for Geopt.

Addicoat and Brain (2010) created a GA, using Python, to optimise small gaseous structures, due to the fact that exhaustively testing every possible configuration quickly became infeasible as the number of atoms present increased. The researchers had attempted to use an exhaustive algorithm and found the computational complexity to be $O(3^n)$ where n was the number of atoms. This implies that the complexity trebled with each newly added atom. Theoretically, the complexity of an exhaustive configuration search could have been factorial, as to test every possible permutation could produce a computational complexity of $O(n!)$, as seen with the travelling salesman problem (Shahab, 2019). Whether the complexity was $O(n!)$ or $O(3^n)$, it would not be sensible to rely on such an algorithm. This complexity issue was in addition to the problem that calculating the energy of each individual solution may also have been computationally infeasible, as discussed previously.

Later, Gueorguiev and Kuttel (2016) similarly used a GA for molecular conformational optimisation. Their aim was to discover three-dimensional structures of proteins that were formed from combining different amino acids. The greatest problem previously encountered by Addicoat and Brain (2010) was still present; the task was made especially difficult by the size of the molecules. Protein molecules contain so many atoms that the researchers were unable to perform precise optimisation using any quantum mechanical methods due to the huge amount of computational power and time that would have been required, even by the simplest model they could find. Their GA

was able to suggest some reasonable structures and was executed quickly. Due to the complexity of these algorithms, many of the studies in this field have been performed using supercomputers, such as the work of Dawson and Gygi (2014) who used randomised search algorithms for optimisation in DFT using a supercomputer.

Ishimoto et al. (2011) investigated the use of a Monte Carlo simulation for optimisation of butane and used a potential energy surface (PES) to analyse the results, which is a plot depicting the energies of structures according to bond lengths and angles and can be used to spot circumstances that lead to high and low energies. The Monte Carlo method was chosen because of its ability to quickly overcome local minima, but the researchers concluded that too much time was spent calculating unfavourable structures. Geopt was planned to create a PES so that the user could view the points of lowest energy on the plot.

The literature had frequently mentioned the problems associated with an exhaustive algorithm and used this to justify the need for an EA. Because of this, an algorithm based on an exhaustive search was planned to be added to Geopt so that the EA could be compared to a more traditional algorithm. The algorithm would not be completely exhaustive as that would not be possible to run on a desktop computer. It was planned to move every atom around every other atom in the molecule, but only test a sample of positions at each of these stages rather than every possible spatial point in the cell with respect to each atom.

The EMT technique used in Geopt was derived from the work of Jacobsen et al. (1996), who created an EMT method for certain metallic structural geometries. This method is simpler than others, including DFT, and executes quickly on a typical computer, although it lacks accuracy as it makes many generalisations and assumptions. Kim and Moon (2006) reported that, although it was sometimes useful, EMT often failed to describe structures correctly. This EMT method was designed specifically for metallic structures consisting of some of only seven elements. Its use for all the elements in Geopt is given as a proof of concept and is unlikely to yield practical results in most cases. EMT was chosen for Geopt because of its simplicity and fast execution time. Furthermore, a pre-existing EMT Python package was freely available from Blomqvist et al. (2017b), which meant that the project's focus could remain in the computing domain.

Project Plan

Roadmap & Sprint Plans

Appendix A contains the roadmap for the sprints. The plan was that each sprint would last a fortnight and would be based around a user story, apart from the sprints which were reserved for the planning and testing stages of the project. The following list summarises the plan for each sprint, starting with sprint zero:

0. Planning the project.
1. User story: a user wishes to create a molecule.
2. User story: a user wishes to predict a shape using an EA.
3. User story: a user wishes to predict a shape using a different algorithm.
4. User story: a user wishes to view a molecule and its analytical information, including a PES.
5. Usability testing and making changes to the program.
6. User story: a user wishes to interact with the algorithm.
7. Usability testing and making changes to the program.
8. Making finishing touches and bring the programming part of the project to an end.
9. Creating showcase materials and making the transition from the coding stage to the report writing stage.

The time after sprint nine was planned to be for writing the report, although continued code development during this time was not explicitly ruled out. The agile methodology allowed flexibility for the plan to change if necessary.

Risk Assessment

Table one shows the risk assessment for the project, which was created during sprint zero, the planning phase. The 'exposure' measure is the 'likelihood' rating multiplied by the 'impact' rating and can be used as an indication of the severity of the risk. Eventually, all six of the identified risks were realised in minor forms. This suggested that the likelihoods of the risks had been underestimated, a mistake to bear in mind in future projects, along with the discovery that the one risk often led to another. Risks R1 (calculations too computationally expensive) and R2 (evolutionary algorithms take too long) were linked because they were affected by the same code, so they occurred simultaneously. The planned measures for these risks were to use optimisations such as multiprocessing, which is a parallelism tool provided by Python and based on multithreading, and try different techniques.

Risk R3 (programming difficulties/lack of knowledge) occurred at the same time as R5 (unable to get desired results from algorithms), for the same reason. The algorithm often produced undesirable and inaccurate results because it used an energy calculation which was not suitable for the molecule chosen. More knowledge of the area could have enabled the programmer to improve the way in which the calculations were performed, leading to better results. The planned measures for these risks were to research the area and discuss problems with the supervisor. The depth of research into the area was limited by the fact that the priority for the project was computer science, not chemistry, and the supervisor's advice was that the algorithms' success should be measured, primarily, from a computational point of view. It was important to bear this in mind to avoid risk R4 (too much to do/not finish on time), which had the maximum exposure rating (nine out of nine). Risks R4 and R6 (coronavirus or illness) were also linked because the developer was unwell for some time in March, which meant that the progression of the project was delayed by up to two weeks. The planned responses to these risks were to prioritise the workload and use Agile, as well as knowing the policy for deadline extension. An extension was not sought but the prior planning and Agile methodology helped the project to get back on track after the set-back and it was possible to alter the plan in order to catch up.

| Reference | Risk event | Likelihood 1=low 3=high | Impact 1=low 3=high | Exposure 1=min 9=max | Plan |
|-----------|---|-------------------------------|---------------------------|----------------------------|--|
| R1 | Calculations too computationally expensive | 2 | 2 | 4 | Use Python packages for calculations. Choose appropriate methods. Simplify calculations. Try multithreading/GPU options etc. |
| R2 | Evolutionary algorithms take too long | 3 | 2 | 6 | Use fewer iterations. Use different technique, selection criteria etc. Make code simpler. |
| R3 | Programming difficulties/lack of knowledge | 3 | 2 | 6 | Have regular meetings with David. Do lots of research and practice. |
| R4 | Too much to do/not finish on time | 3 | 3 | 9 | Keep working throughout the year. Adjust plans if necessary. Focus on most important things first. Use Agile. |
| R5 | Unable to get desired results from algorithms | 2 | 1 | 2 | Alter mutations etc. Analyse and test algorithms. Do plenty of research. |
| R6 | Coronavirus or illness | 2 | 3 | 6 | Know the University's EC policies. Work from home where possible. |

Table one – Risk assessment.

Legal, Social, Ethical and Professional Matters

Ethical matters for artificial intelligence (AI) projects needed to be considered. According to Müller (2020), a widespread concern is that the use of AI and computerised systems in science risks making human jobs redundant. A human chemist may use their knowledge to suggest the most probable geometries for systems without the use of a computer at all. Chemical geometry optimisation may assist scientists but relies heavily on human expertise and does not seek to replace humans and is not currently able to do so. Another concern is that AI may produce solutions to problems without the method being understood by the humans who use it. This may result in seemingly correct solutions being formulated from irrelevant data, making the process inapplicable in other situations, and it may also prevent the method from being repeated when needed. EAs are relatively transparent due to their trial-

and-error nature using random numbers, but one way to avoid the situation is to view the progression of the algorithm step-by-step.

For all three stages of user feedback, the generic ethical approval documentation was used because all the participants were students of the University of Plymouth. As required by the University, consent forms and information sheets were written for the user feedback tests and questionnaires. This documentation can be viewed in appendices B and C. The questionnaires were anonymous.

All the images shown within Geopt were created by the developer. The background image of the poster was created by Filipe (2017).

Figure six displays a section of the original source code of the EMT package used in Geopt. The code contains realistic data for seven elements and sample data for four more. The developer of Geopt extended a copy of this code to include sample data for some other elements, which was necessary because it allowed the user to create many more molecules. The original source code for EMT was created by Blomqvist et al. (2017a) and is freely available for modification under the GNU Lesser General Public License.

```
parameters = {  
    #      E0      s0      V0      eta2      kappa      lambda      n0  
    #      eV      bohr    eV      bohr^-1    bohr^-1    bohr^-1    bohr^-3  
    'Al': (-3.28, 3.00, 1.493, 1.240, 2.000, 1.169, 0.00700),  
    'Cu': (-3.51, 2.67, 2.476, 1.652, 2.740, 1.906, 0.00910),  
    'Ag': (-2.96, 3.01, 2.132, 1.652, 2.790, 1.892, 0.00547),  
    'Au': (-3.80, 3.00, 2.321, 1.674, 2.873, 2.182, 0.00703),  
    'Ni': (-4.44, 2.60, 3.673, 1.669, 2.757, 1.948, 0.01030),  
    'Pd': (-3.90, 2.87, 2.773, 1.818, 3.107, 2.155, 0.00688),  
    'Pt': (-5.85, 2.90, 4.067, 1.812, 3.145, 2.192, 0.00802),  
    # extra parameters - just for fun ...  
    'H': (-3.21, 1.31, 0.132, 2.652, 2.790, 3.892, 0.00547),  
    'C': (-3.50, 1.81, 0.332, 1.652, 2.790, 1.892, 0.01322),  
    'N': (-5.10, 1.88, 0.132, 1.652, 2.790, 1.892, 0.01222),  
    'O': (-4.60, 1.95, 0.332, 1.652, 2.790, 1.892, 0.00850)}
```

Figure six – Elements included in the original EMT source code (Blomqvist et al., 2017a).

Implementation

Technologies Used

The project used an Agile, cyclic approach rather than the waterfall method because it was important not to leave the analysis to the end of the project, as doing it incrementally throughout the course of the project helped the developer to choose better designs and parameters for the algorithms and improve the UI in line with regular user feedback. The model-view-controller (MVC) design pattern was used as it enabled clear separation between the UI, the data and the functionality in the code.

Python was used as the programming language for the project because there were many scientific tools available in Python which could be used, allowing the developer to spend more time on core features of the project. The use of C# or C++ was also considered, as these are fast languages which also have a range of scientific tools available, but

Python was preferred due to the availability of the atomic simulation environment (ASE) package and the EMT energy calculator. Furthermore, graphs, plots and visualisations are easy to show in Python.

Tkinter was chosen for the UI as its 'widgets' are useful for creating desktop-based forms and executable programs. Extensible markup language (XML) was used for the data storage of the chemical elements because this was a small amount of data which would not require a relational database, and XML files required a small amount of space in the drive on which the program was kept. Accessing the XML file as a tree with a document object model in Python was a fast and simple process.

Sprint Zero – Planning

Geopt was planned to suggest multiple structures which could potentially be isomers in some circumstances, and not restrict the molecule creation to realistic formulae so that the user could study theoretical systems that would be unlikely to occur in reality. The study of entities which do not appear naturally on Earth has often been useful to chemists, such as the synthesis of medicines and the discovery of the heaviest elements (Alvarez et al, 2019). Geopt would use random structures rather than estimating initial starting positions based on chemical properties. It was not known whether any geometric structure could successfully be estimated purely through randomisation, with no a priori knowledge of the system. It was not anticipated that all types of molecule could be modelled in this way, but perhaps it could work for some simple molecules. It was expected that the number of atoms in the molecule would be restricted to a small value by the computational expense.

A questionnaire was created and sent to science students so that the project could be designed to meet their requirements. The full version of the questionnaire can be found in appendices B and C. The questionnaire received two responses which can be seen in appendices D and E. Participants wanted the option to interact with the EA. Next, the backlog began. The GitHub repository, Microsoft Planner and ReadMe file were created.

The project initiation document was compiled and can be viewed in appendix F. The supervisor suggested that the project may have been at risk of focussing too much on chemistry and not enough on computer science. In response to this, it was planned that pre-existing Python modules would be used for energy calculations and other chemical properties so that the majority of time and effort could be directed towards computing-related tasks.

Sprint One – Building a Molecule

Sprint one was based on the user story, 'a user wishes to create a molecule'. Firstly, a page layout plan was drawn, and is shown in figure seven. Secondly, an activity diagram was drawn and is shown in figure eight. It was later decided that the user should not have to click 'clear' as per the activity diagram, so this was removed. It was planned that the user would be able to select elements from a periodic table or type a molecular formula to build a molecule by instantiating a 'molecule' class. The entry point of the application and UI was created in Python using Tkinter. The XML file was constructed to hold information about each element and was used to populate a clickable periodic table. Figure nine displays the XML for the first two elements.

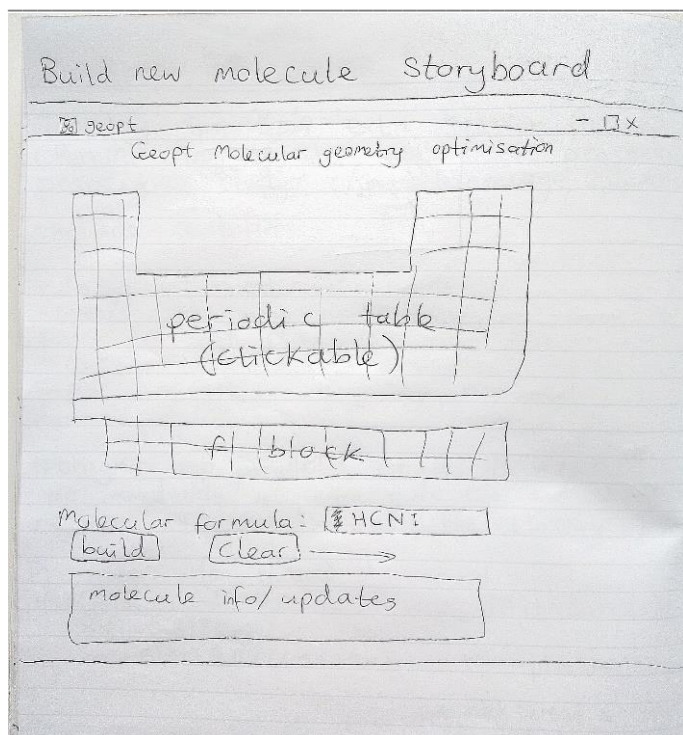


Figure seven – Initial layout design for sprint one.

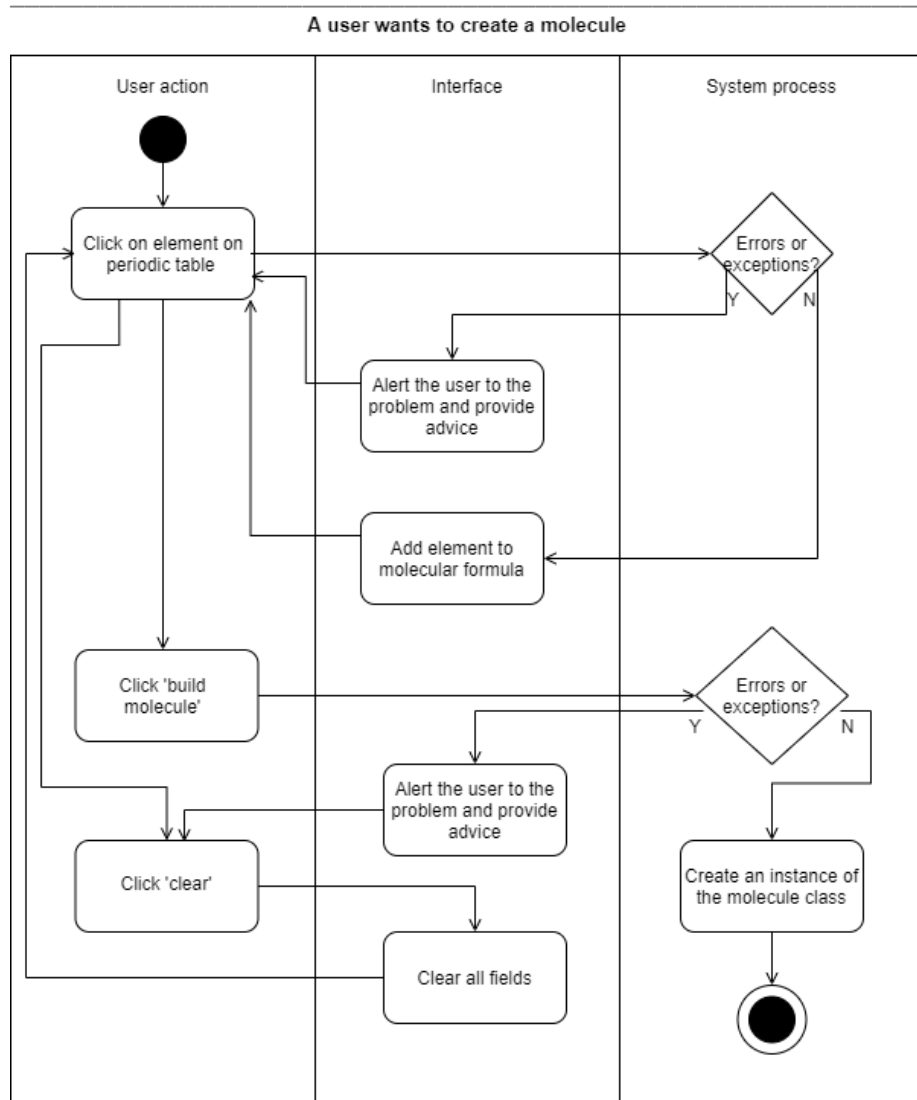


Figure eight – An activity diagram for sprint one.

```

<?xml version="1.0" encoding="UTF-8"?>
<elements>
  <period periodNum = "1">

    <element atomicNumber="1">
      <symbol>H</symbol>
      <name>Hydrogen</name>
      <mass>1.0078</mass>
      <charge>0</charge>
      <group>1</group>
    </element>

    <element atomicNumber="2">
      <symbol>He</symbol>
      <name>Helium</name>
      <mass>4.0026</mass>
      <charge>0</charge>
      <group>18</group>
    </element>
  </period>
</elements>
  
```

Figure nine – Part of the XML file created for the elements.

The idea for the UI was then changed to improve the layout and make the options clearer and simpler. The string containing the molecular formula which the user typed required several steps of processing to convert it to a list of atoms. The algorithm iterated through the string, picking out capital letters, lowercase letters and digits to find the constituent atoms. A loop iteration was skipped if the previous symbol was two digits long, e.g. 'Na', to prevent overwriting the second digit. Multiples of the same element were also checked, e.g. 'CHOCH₃'.

Sprint Two – Predicting a Shape

Sprint two was for the user story, 'a user wants to predict a geometric structure'. The goal was to create an EA that could predict the shapes of molecules selected by the user.

Calculations were created to place the atoms in a grid so that they were evenly distributed within the grid. If the cell was treated as a cube it could be divided into segments by splitting each of its three dimensions into the number which was the ceiling of the cube root of the number of atoms. The atoms could then be placed in the centre of each segment. Atoms were different sizes which meant that extra calculations needed to be done to work out the atoms' sizes and adjust their positions and the grid size accordingly. The atomic radii followed a pattern on the periodic table so their places on the table were used to estimate relative size values. The atoms were placed into the unit cell with relative co-ordinates which could be used in functions which approximated the overall energy of the molecule. The unit cell's size and shape mattered because the theoretical space around the molecule could have affected the distances and interactions between multiple molecules, so the cell was designed to scale to the largest atom. Tests were performed in an isolated directory in the project before the code was moved to its intended directory and connected to other parts of the program.

Atom and ion classes were created to match the unified modelling language (UML) diagrams shown in figures 10 and 11, but although it had originally been planned that Geopt would have classes for molecules, atoms, etc, it transpired that the ASE package provided these when setting up a molecule or atom. It seemed contrived to have two classes for one object, so the classes created in Geopt were removed. Moreover, a molecule class was unnecessary as only one molecule would be studied at a time.

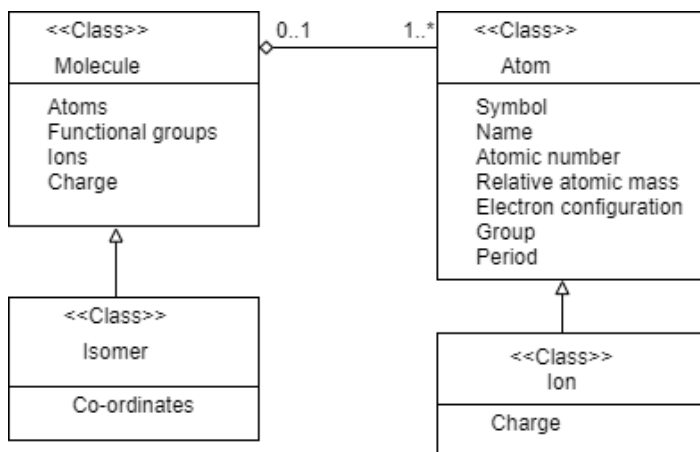


Figure 10 – UML diagram for the first plan for Geopt's classes.

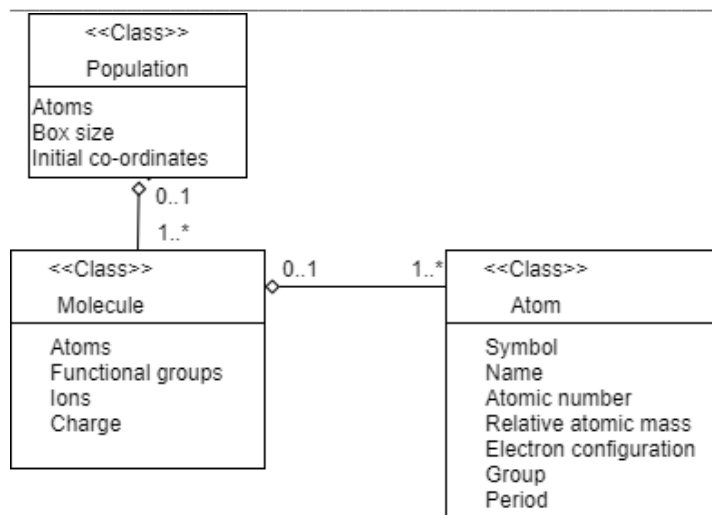


Figure 11 – UML diagram for the second plan for Geopt's classes.

The EA was created. Functions for selection, random and structured generation, permutation, mutation and crossover were created. Various different methods were trialled, some using all these functions and some only using one or two of them. A variety of parameters for these methods were also tested. This was performed in the test directory of the application to find out which methods and parameters worked best and worst. The EAs were unable to predict geometries well although the energy values were improved considerably which was the main objective. After adjustments, the energy predictions were improved by three times those achieved by the first EA. Eventually, an EA was chosen which created a population of many permutations of a molecule, chose the best permutation, created many mutated copies of it, introduced some completely random structures, selected the best of these, and continued like this until the lowest energy remained within a certain range for a specified number of iterations. The problem of getting trapped in local minima was hard to overcome. The algorithm was named the many-molecule EA (MMEA) because it was an EA which used a population of many versions of a molecule and there were several other EA attempts at that time.

A function was created to correct a tendency for Hydrogen atoms to 'fall off' the molecule, which pushed them towards carbon atoms. Generally, this function was not helpful and made the permutations less favourable in some cases, so it was omitted from the final EA. Its code was moved to a shared Python file which contained EA functions so that if more EAs or parameter options were later added, these functions could be easily accessed by others.

Sprint Three – Other Algorithm

Before the programming began, it was expected that the EAs would require extra time, so another sprint had been set aside for creating new algorithms. After the literature review, however, this sprint had been allocated to creating an algorithm based on an exhaustive method so that it could be compared with the EA. The algorithm was named the per-atom exhaustive test (PAET) because it worked with one atom at a time.

The PAET started with an empty cell and introduced each atom at a time, moving all other atoms systematically, testing a set number of random locations centred around one another and testing for the best energy. Since every atom moved around every other atom each time, the time taken to complete the algorithm increased exponentially with the input size. For this reason, it was necessary to use a small population size and a low number of iterations, which meant that not enough variations or comparisons were made, leading to inaccurate results. In some cases, this meant the best energy achieved by this algorithm was ten times worse than that of the EA.

Multiprocessing was applied to speed the algorithm up. Experiments were performed to choose the number of processes to run in parallel and the number of inner loop iterations to be run at each stage of each of these processes. Table two presents the results. Six threads and six inner loop iterations were chosen.

The algorithm was further improved by scaling atom movements relative to their size and their neighbours' sizes and positions. An example of a successful output with its optimised energy value in electron-volts is given in figure 12. Although table two implies that half the results were chemically reasonable, they are the best results taken from all the threads in the test. The PAET produced acceptable structures approximately 25 % of the time although the energy was reduced by some amount in almost all of the tests.

| Molecule tested | 4 threads, 4 iterations | | 4 threads, 8 iterations | | 8 threads, 4 iterations | | 6 threads, 6 iterations | |
|---------------------------------|-------------------------|----------|-------------------------|----------|-------------------------|----------|-------------------------|----------|
| | Energy / eV | Time / s | Energy / eV | Time / s | Energy / eV | Time / s | Energy / eV | Time / s |
| C ₁₂ | 4.5 | 21 | 3.9 | 75 | 4.5 | 47 | 3.2 | 40 |
| C ₆ H ₆ | 14 | 19 | 23 | 42 | 26 | 33 | 20 | 40 |
| C ₃ H ₆ O | 16 | 10 | 27 | 22 | 7.0 | 15 | 14 | 17 |
| C ₂ H ₆ | 4.4 | 12 | 8.5 | 62 | 3.8 | 36 | 5.4 | 39 |
| N ₄ O ₄ | 1.3 | 21 | 1.7 | 58 | 1.3 | 21 | 1.2 | 25 |
| Al ₂ O ₃ | 3.1 | 17 | 3.2 | 33 | 3.2 | 16 | 3.3 | 16 |

Table two - Energies and times taken for different methods, with chemically reasonably shaped results highlighted in green and unreasonably shaped results highlighted in orange.

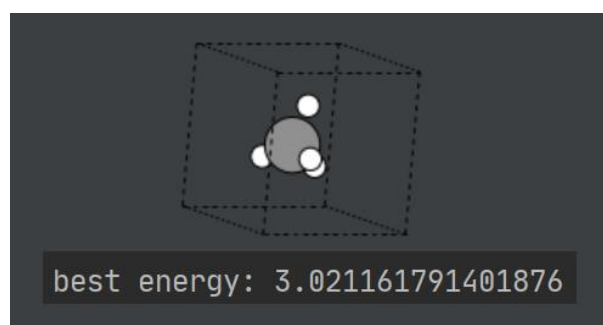


Figure 12 – Good prediction for methane.

When the PAET ran for too many iterations, the molecule started to break apart, as the energy calculator did not anticipate the energy required to break bonds and just analysed the ground state. This meant that the energy was lowest overall when no atom was interacting with any other atom and caused atoms to move apart, as shown in figure 13. Figure 14 displays a phenomenon which caused atoms, usually all the hydrogen atoms, of some structures to group together, while pushing some other atoms away from this cluster. Although this initially seemed like the effect of a particularly electronegative atom, tests demonstrated that it was actually caused by the way in which the algorithm accessed the list data structure in which the molecules' components were stored. The list of atoms was rearranged to place all hydrogen atoms at the end.

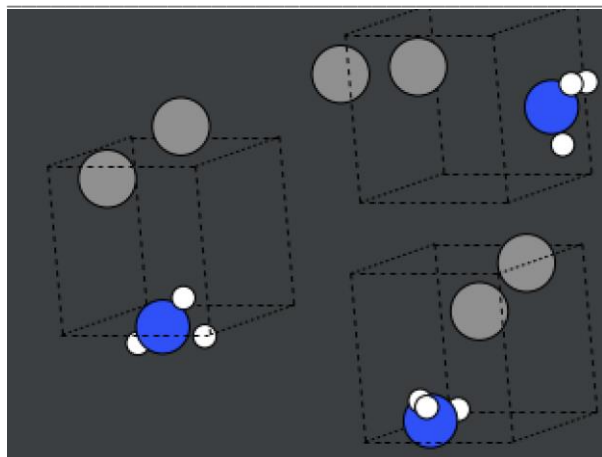


Figure 13 – Recurring separation problem with acetonitrile.

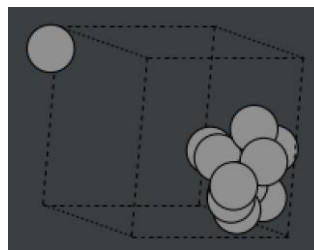


Figure 14 – A group of atoms clustered together.

The algorithm was altered so that after each atom had circulated the group, the structure reverted to the most energetically favourable state found during that loop. Table three shows the effects of this. Although it did not cause a large improvement, the feature was kept because the amount of processing required was small compared to amount used by the energy testing. The PAET was compared to the MMEA as detailed in appendix G. The MMEA was found to perform better, overall.

| Molecule | Energy achieved / eV | |
|-------------------------------|----------------------|-----------|
| | Update | No update |
| C ₁₂ | 3.2 | 3.9 |
| Benzene | 8.0 | 9.0 |
| Acetone | 5.5 | 8.6 |
| Ethane | 2.7 | 2.6 |
| N ₂ O ₄ | 3.6 | 1.2 |

Table three – Results of updating and not updating the structure after each atom addition loop.

Sprint Four – Viewing Results

Sprint four had originally been dedicated to testing, user feedback and amendments, and sprint five had been allocated to displaying visualisations and analytical information. It became apparent that it would be difficult to conduct questionnaires with participants without the presence of more visual displays in the application. Sprints four and five were swapped so that sprint four was dedicated to displaying visualisations and analytical information and sprint five was allocated to testing, user feedback and improvements.

The optimised geometric structure was depicted on the screen and a window showed analytical information. A plot displayed the different positions tested during the evolution. Datasets were created to hold information needed for the plots, and gradually grew while the algorithm ran. This was done to avoid needing to iterate back through all the previous solutions to create the plot, which caused a delay on loading the view. The data needed to be recreated during each execution. It would not be appropriate to store a table of values because none of the values were definite; they were guesses and could not be relied upon.

The energies of molecules were scaled to fit within a range so that the lowest energies found were close to zero and the highest were close to one. This was done by dividing each energy point by the highest energy value produced by the algorithm. The resulting number was then subtracted from the green value and added to the red value to colour

each data point, representing an atom's location in terms of the molecule's energy at that position. Each point was also labelled according to the atom's symbol. The original design is shown in figure 15.

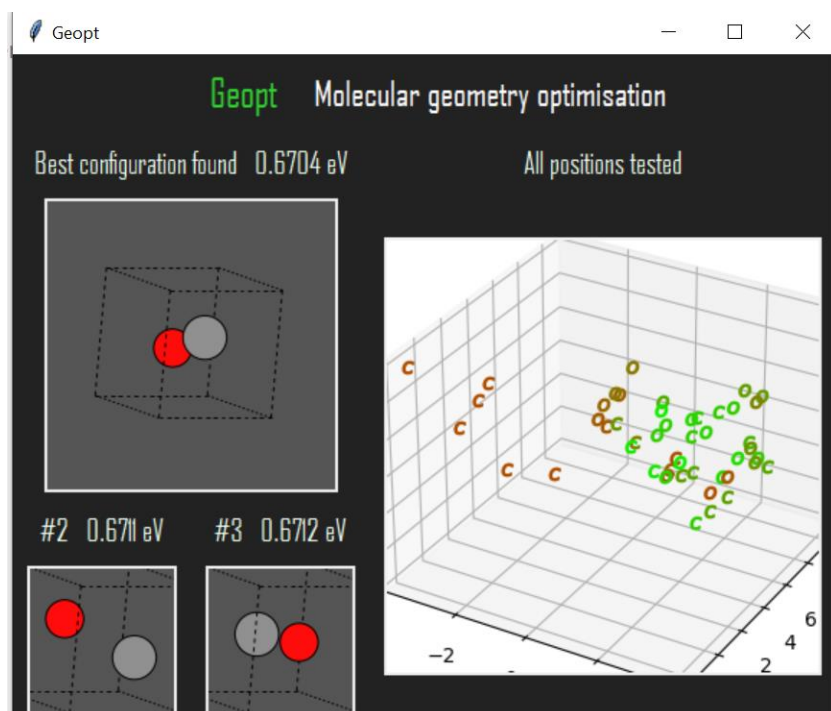


Figure 15 – The first design.

The analysis view was expanded to show the three most optimal versions of the molecule so that the user could view the differences between them and decide which structure was the most favourable. It could also potentially propose isomers of some molecules. The plots were initially displayed on the screen using a grid layout, as shown in figure 16, but a PES plot was later included, which meant that the plots would not all fit on the screen at once without becoming too shrunken to be easily read. Instead of creating multiple plots each with a PES for a part of the molecule, all the PES were placed onto the same plot and distinguished by colour. It was possible to specify the axis rotation so that the energy scale was displayed vertically but this pushed the axis labels off the scale. To solve this problem, the layout was redesigned to allow the plots to fit comfortably on the screen. The new design included an extra information window for each version of the molecule and it utilised the datasets created earlier to display more information to the user, shown in figures 17 and 18.

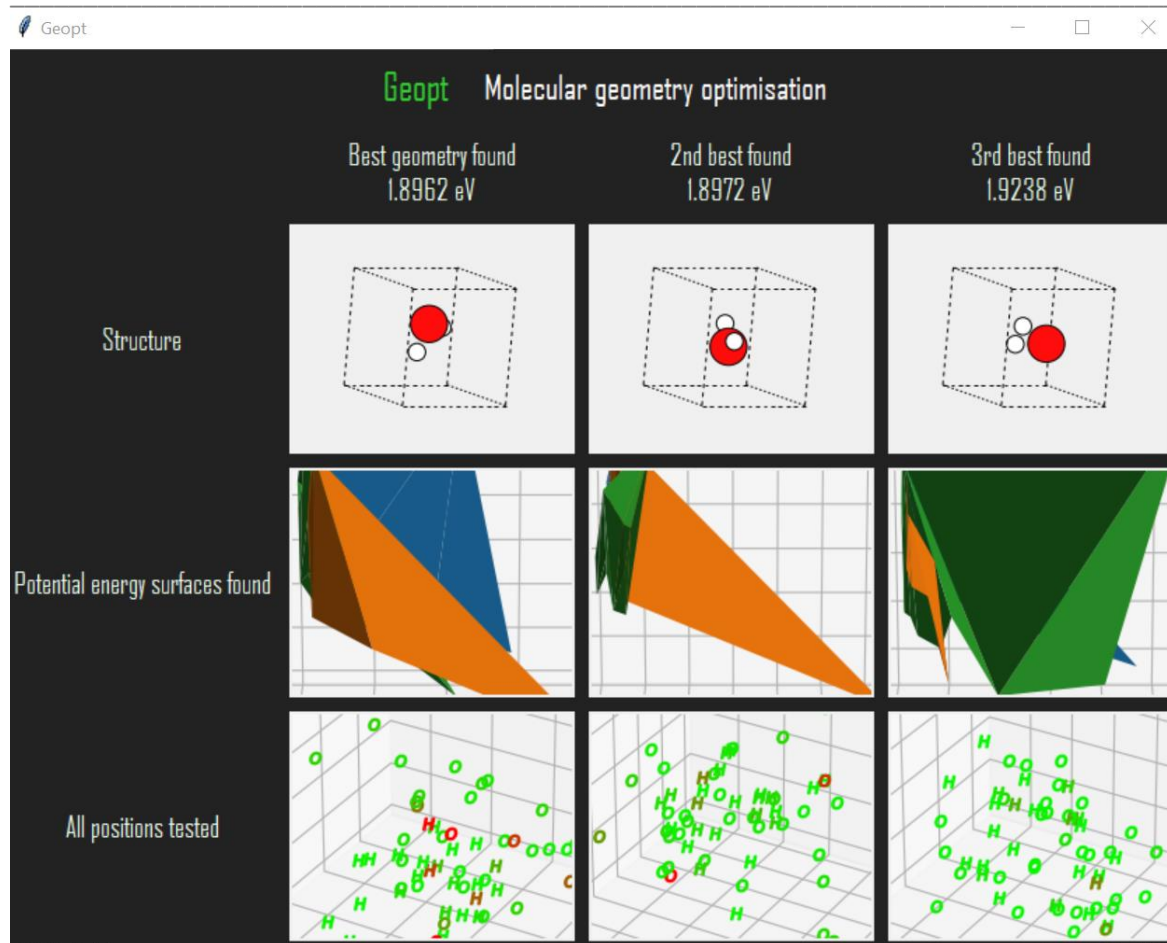


Figure 16 – The second design.

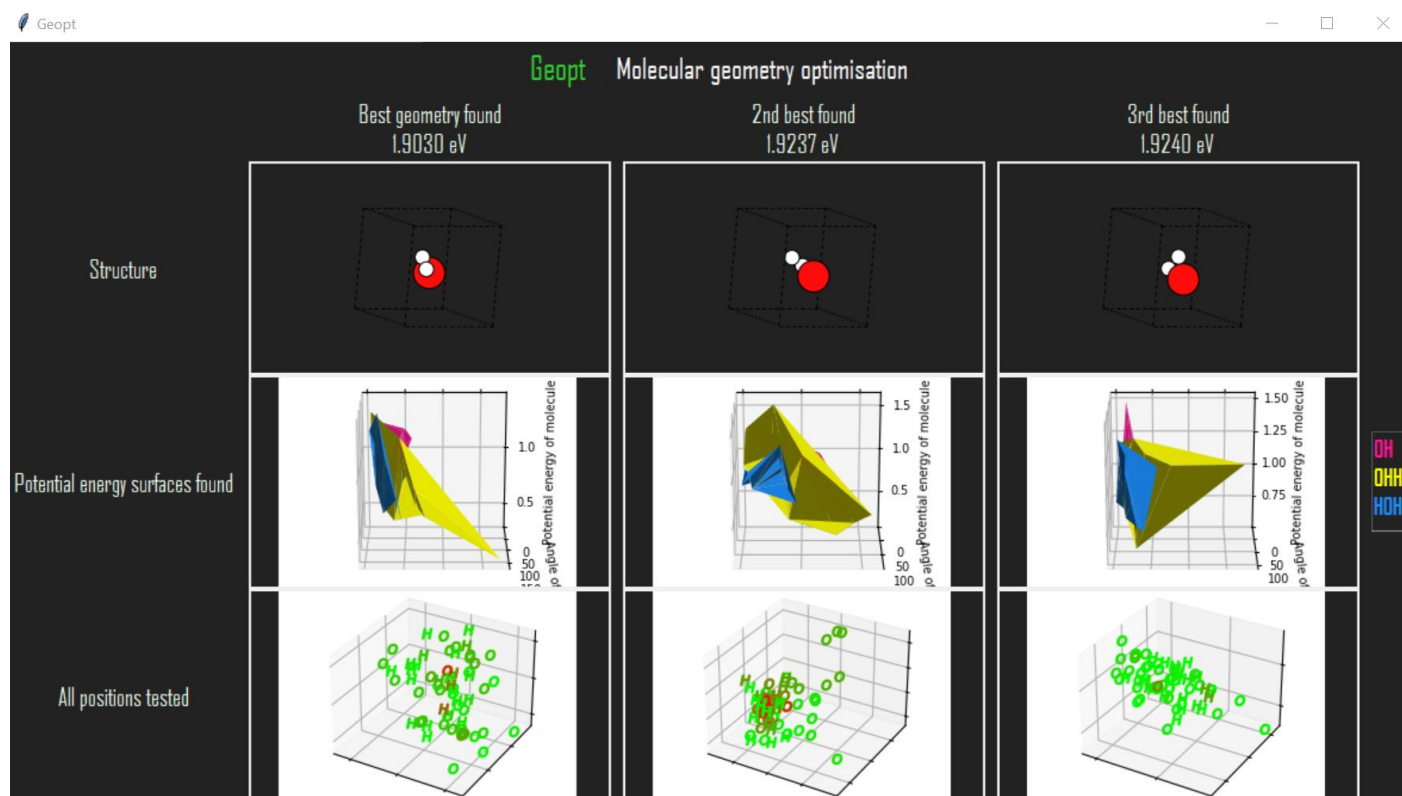


Figure 17 – The analysis window of the third design.

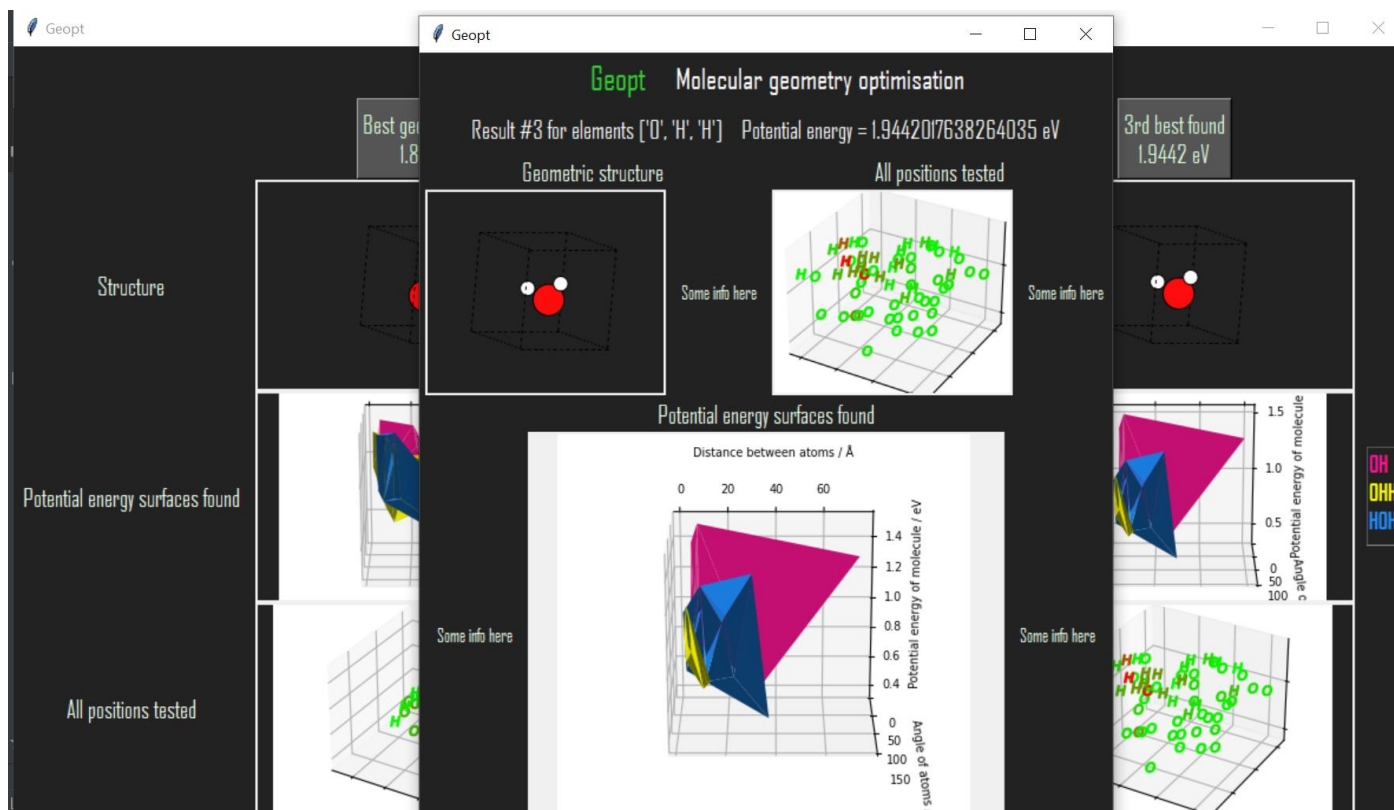


Figure 18 – The information window of the third design.

The PES compared the energy of the molecule to the distance between two atoms and the angle over three atoms. Programming and visualising it was difficult because each axis had a different number of variables. Tri-surface plots were found to be preferable to surface plots with mesh-grids or wire-frames because of the vast variation in data points. A lack of data often meant that the PES was not detailed enough to create a proper surface but it was not worth spending extra execution time gathering more data which may not have been relevant. If the molecule was diatomic the PES graph would be two-dimensional as the only factor to consider would be the bond length, or the distance between the two atoms in the case of Geopt. As the number of atoms in the molecule exceeded two and increased, the degrees of freedom and the dimensionality increased to include all distances between atoms and angles over groups of three atoms. The fact that Geopt did not utilise any specific bonding information meant that the number of distances and angles to consider was greater than it otherwise would have been, as it was assumed that every combination of atoms could contain a bond. For example, a human would know that only one angle would need to be considered in a water molecule, but without knowledge of bonding, a computer may assess all possible angles and distances between atoms in the molecule. Additionally, a human would expect the distances between both hydrogen atoms and the oxygen atom to be the same, but the computer would not.

Several of the functions created in this sprint were amalgamated into one, larger function. This was done to speed up the algorithm because it meant that multiple things could be computed inside the same loop, reducing the number of loops and function calls. A drawback of this was that this made the code less separable and defied principles of good programming practice such as SOLID, making the code harder to read, harder to debug and more confusing to work on later.

Sprint Five – Testing

The plan for sprint five was to gain feedback from participants and possibly make changes to the application based on this feedback. One test was conducted with a chemist and two tests were completed by computer scientists. The questionnaires can be viewed in appendices H to K. Although the computer scientists were unsure about the chemistry content, they found the application easy to use and were able to construct molecules in the way intended. To assist with the usability test, a video of Geopt was made available. It can be viewed here:

<https://youtu.be/ylv4J85m95Y>

A participant attempted to create a molecule containing two nitrogen atoms and two oxygen atoms but Geopt's exception handling prevented this molecule from being built and told the user that it was invalid. Geopt should have allowed the user to build any combination of up to 12 atoms and should have only prevented the formula from being used if the input string contained characters which were not recognised as atomic symbols, such as 'hello'. It was found that this problem had not been caused by a logical error when interpreting the string, but by an error in another function further into the process. It was hard to find the error and this was an indicator that exception handling needed to be more specific and that initial testing should have been applied to smaller, separate pieces of code. The error was fixed and exception handling was improved to prevent future difficulties.

It was suggested that the information buttons should have been made clearer, as a participant was unaware of the buttons' functionality. It was recommended that the test positions plot should be easier to read and understand. The layout of the UI was changed according to these comments. It was also advised that the user should be reassured that the program was calculating and had not frozen, because the calculation process could take a long time. Firstly, a text box was placed underneath the button to start the calculation process, which alerted the user to the fact that the calculation could take a long time. This was later changed to a pop-up window which showed the user a generous estimate of the time to be taken for the chosen molecule and asked the user if they wished to proceed, as presented in figure 19. The time was calculated by extrapolating a trendline of best fit which was exponential for both algorithms but steeper for the PAET. Finally, it was suggested that the number of iterations could be user-defined, which was already the plan for later in the project, as per the roadmap.

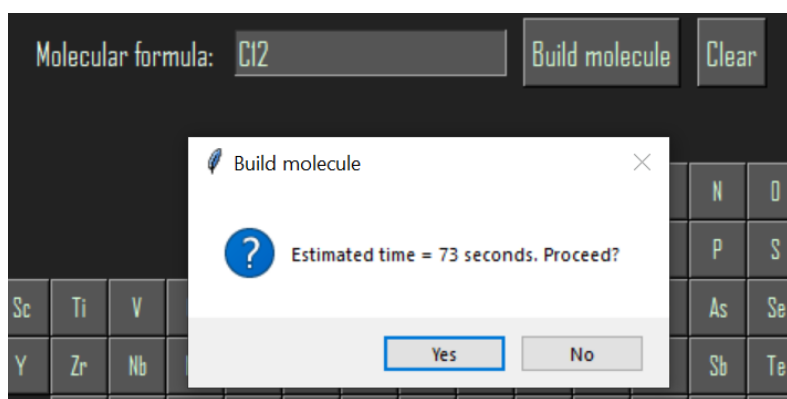


Figure 19 – Time estimate of the algorithm.

The MMEA was adapted to use the data format that the PAET used so that it could pass data to the UI without any more changes needing to be made to the PAET or the UI. In order to reduce the time taken for the user to receive the output, some code optimisations were performed, including moving as much code as possible out of loop bodies and using scalar replacement. The datasets used for plots were limited so that they would not expand beyond a certain size. This reduced the time taken to display the plots after the algorithms had finished.

Sprint Six – Interactivity

Interactive features were added to the UI to allow the user to select the algorithm and tune its parameters. The following options were made available to the user:

- The algorithm to use.
- The presence of periodic boundary conditions for the cell.
- The probability distribution shape of the random mutations.
- The range of the random mutations.
- The size of the population.
- The number of processes to run in parallel.
- The application of permutation to new generations (for the MMEA).
- The application of crossover to new generations (for the MMEA).
- The display of plots.
- The limit of the number of data points in plots.

A participant had suggested allowing the user to choose the number of iterations but it was thought that this could hinder the progress of the EA because the number of iterations needed to be variable for the EA to run properly and the stopping criterion to be met, although it could have been possible to allow the user to choose the number of consecutive iterations of similar results reached before the process was terminated. A cap on the maximum number of iterations was applied. The user was advised by text boxes to turn off the display of plots and reduce the data points limit if they were building a large molecule as this would speed up the process. This was tested and found to be true. For a test molecule, when 5,000 data points were collected, the execution time was 60 seconds. When the data points were capped at 3,000 but the plots were still shown, the execution time was 49 seconds. When the plots were disabled, the execution time was 46 seconds.

The distribution sizes affected the PAET more than the MMEA. The MMEA tended to produce similar results regardless of the distribution size but the PAET produced high energies when the range was too small, and broken structures when the range was too large, as described by figure 20 of xenon hexafluoride. Tests were performed to find the relative ranges that worked best for the algorithms and these were set as the 'medium', default option, so in many cases, increasing or decreasing the range would produce higher energies than those produced by leaving it at the default value.

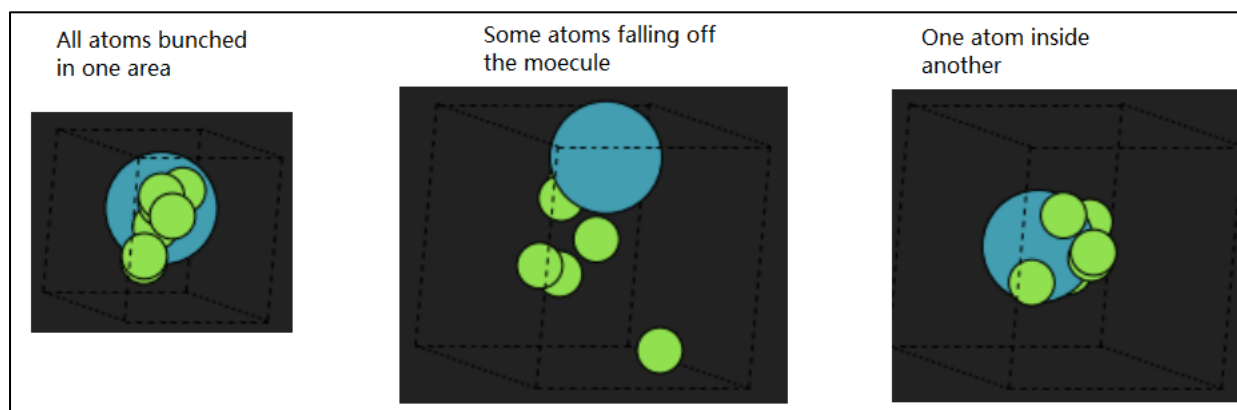


Figure 20 – Problems with the distribution sizes in the PAET.

Sprint Seven – Testing

Sprint seven was allocated to acting on user feedback and making final improvements to the project. Students provided more feedback on the program. Based on this feedback, some changes were made. It was noted that the many information buttons on the configuration window made the UI look cluttered and confusing, so the buttons were temporarily replaced by text boxes. The view of the molecule sometimes hid some atoms behind other atoms. An additional view of the molecule, rotated by 180 degrees, was included in the molecule's information window so that all the atoms could be seen. Distances between atoms were explicitly shown to the user in a text panel so that the user was not required to deduce distances by looking at the PES or other information. To assist with the usability tests, a video of Geopt was made, which can be viewed here: <https://youtu.be/llvHbYyEO6Q>. Another video was made after the feedback was acted on to ascertain whether the participants approved of the changes. This video can be viewed here: <https://youtu.be/irX853Cr7zk>. Subsequent feedback was that the text boxes made the settings window look cluttered and confusing, and tooltips were instead advised. Tooltips were implemented and the final version of the UI is shown in figure 21.

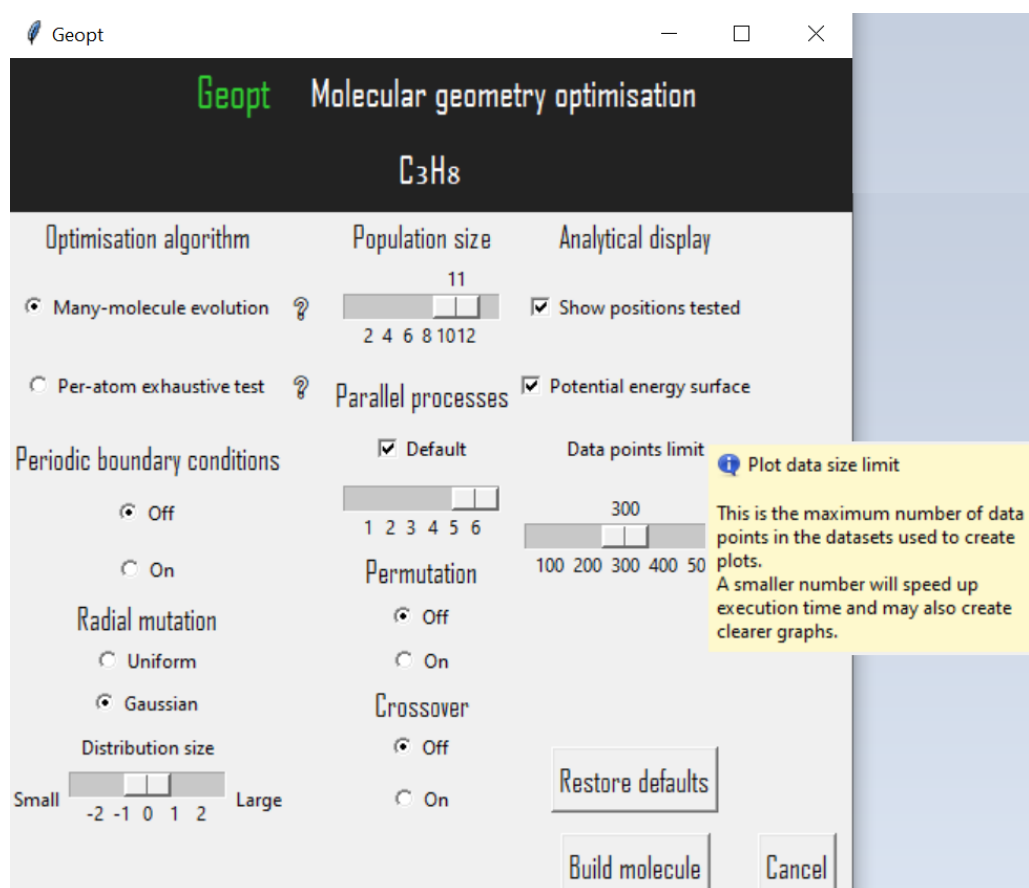


Figure 21 – The final UI for the settings window.

Sprint Eight – Final Touches

Sprint eight had been dedicated to making final changes or additions in order to finish the development of Geopt. In consideration of the problems encountered with atoms moving away from others or bunching together, development of an additional fitness function which ranked solutions by their distances began, but there was not

enough time to finish it. It would not be useful to rely solely on this measure of fitness but it could have been used in conjunction with the existing fitness function to create a two-objective situation. This would require extra work and analysis and could have complicated the project which was a risk not worth taking due to the fact that it was the final programming sprint.

A previously undetected error was discovered which rarely presented itself so was difficult to find. If the number of parallel process was reduced to two or fewer, the three best solutions could not be retrieved from the PAET because it did not use populations of versions. This was fixed by omitting the display of extra molecules in this case.

Because optimisations had been applied the code, methods and data throughout the project, the execution time had been reduced. The estimate provided to the user was recalculated using the latest version of Geopt and it was possible to increase the maximum molecule size allowed from 12 atoms to 14 atoms.

Figure 22 shows that Geopt was run on Linux Ubuntu to ensure that it worked as it had usually been run on Windows 10.

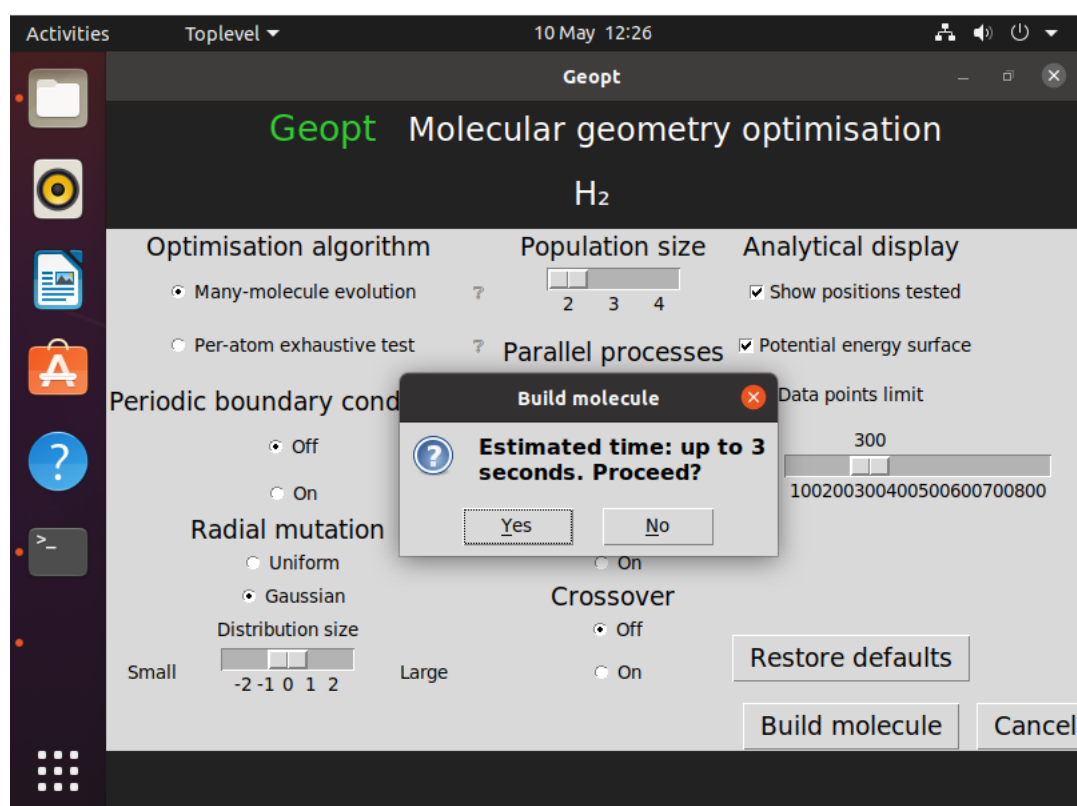


Figure 22 – Geopt running on Linux.

Sprint Nine – Showcase Materials and Report

Sprint nine was reserved for the creation of the poster, the showcase description and planning the dissertation report. The user guide is provided in appendix L and the poster is supplied in appendix M.

Discussion & Analysis

Analysis of Many-Molecule Evolutionary Algorithm

The first plan for the EA is displayed in figure 23. The first execution immediately found a solution with the minimum amount of evolution possible, as the stopping condition was met at its first evaluation. The molecule had not been optimised and had retained its initial structure. This suggested that all the variations which had been applied to the molecule, including the mutation, crossover, permutation and generation of other structures, had produced energies that were not lower than that of the initial configuration. It was suspected that this had been caused by the following problems: the population was too small; the mutations were too large or too small; the stopping condition was not strict enough; and EMT was not an accurate calculation method. It was also found that the majority of atom movements were energetically unfavourable, and so the energy would often need to increase before it could begin to decrease. Because the atoms were evenly spaced apart before the algorithm began, the total potential energy would usually increase when an atom was moved further from one atom and closer to another, as the force applied to some of the atoms would increase. This was understandable from a scientific point of view because there would be few ways in which atoms could reasonably be arranged at the ground state of a system, particularly in such small molecules.

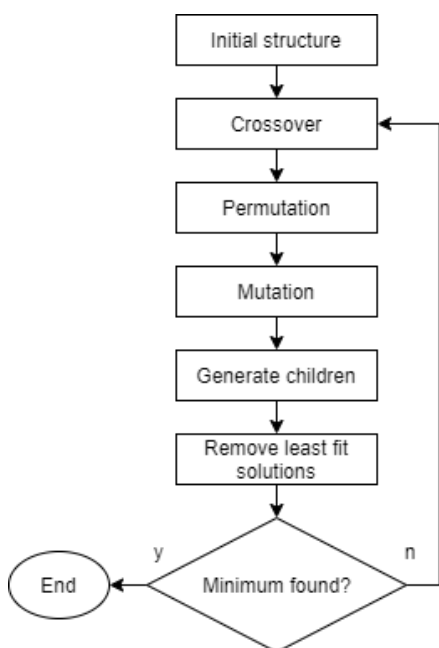


Figure 23 – Flow chart of the original plan for the EA.

Several techniques were applied to improve the MMEA's performance. The goal was to find a set of parameters that generalised to all molecules, rather than working well for some but poorly for others. The algorithm was repeated using each evolutionary operation exclusively to assess how they contributed to the solutions formed.

The crossover function was designed so that it could not combine the wrong atoms by giving each new atom's position a 50 % chance of being selected from one parent or the other. Crossover was discovered to produce more unfavourable than favourable solutions, so it was decided that the crossover function would not be called in the default MMEA, although it could be activated by the user. It was suspected that the reason for its desultoriness was that combining atoms' co-ordinates from different versions of the system was too destructive for such a small population and did not take into account the evolutionary process that had led to those arrangements being formed.

Permutations were shown to offer the largest energy reductions in many cases, when a reasonable shape was created and permuting some atoms within this shape produced a more favourable arrangement. The frequency of permutation was increased.

The ranges of random mutations were adjusted so that they were determined in proportion to the size of the atoms and the unit cell. Because mutations were implemented as selections of points in three-dimensional space, a Gaussian distribution and a radial distribution of mutation ranges were implemented, as well as the uniform distribution. These functions were further utilised by the PAET and included in the interactive settings after the discovery that their effectiveness depended on the molecule chosen.

The population size was chosen to scale with the size of the molecule, as larger systems required more exploration. A drawback of this was that it caused the algorithm to take significantly more time and power for larger molecules, which was the original problem that evolutionary approaches to optimisation sought to overcome.

A corrective function was created to push hydrogen atoms towards carbon atoms because of their tendency to move further away from the molecule with each iteration. It was later determined that this method was not desirable because it did not address the problem, which was partly that the EMT calculator was unable to appreciate specific bonding interactions, and partly that the total potential energy of the system was lowest when the atoms moved so far apart that none of them interacted with one another. Instead of forcing hydrogen atoms to stay near carbon atoms, a softer approach was taken, which consisted of adjusting the stopping criteria and restricting the range within which atoms could travel in one step.

Random structures were introduced at each iteration to diversify the population in an attempt to avoid becoming trapped at local minima. A smaller random mutation was used for parents' descendents and a larger random movement was used for the new, unrelated versions.

An extra energy test was included in the algorithm so that no version would be created if its energy was more than a hundred times higher than that of its most direct ancestor. In this case, the relevant function would start again. Repeating generations of new molecules in this way did not add a noticeable amount of time to the process, but testing the energy so many times did noticeably increase the time taken. After all the changes were made, the algorithm was structured as described by figure 24, which may appear more complicated than it really is due to the fact that functions removed the need for the repetition it displays. A readout of the process is given in figure 25.

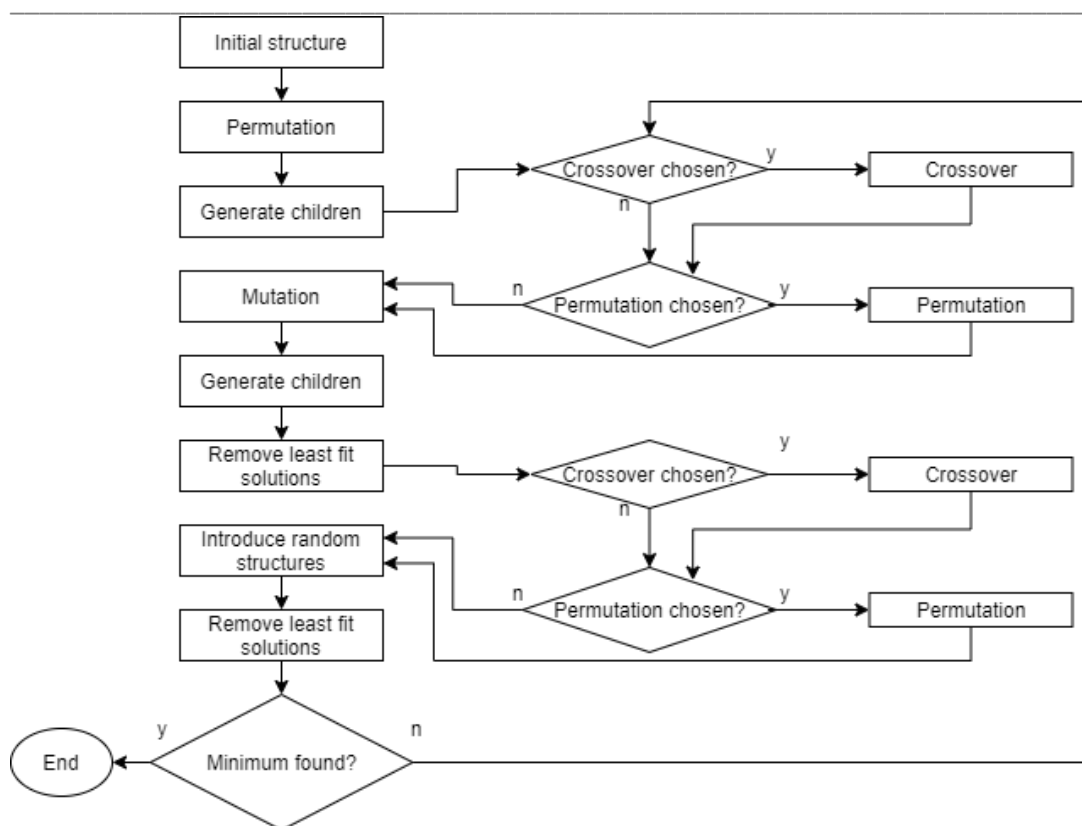


Figure 24 – Flow chart of the final EA.

```

main x
setting up calculator and cell
box size [4.25, 4.25, 4.25]
Starting energy = 2.9856690174377176
Evolving the populations
Evolving the populations
Evolving the populations
Evolving the populations
Evolving the populations
Evolving the populations
Evolving the populations
5 iterations
5 iterations
7 iterations
14 iterations
18 iterations
20 iterations
processing results from multiprocessing
EA has finished
Optimised energy = 1.881462155384054
Time taken = 1 seconds
  
```

threads starting

threads finishing

Figure 25 – Stages in the optimisation of H₂O.

As discussed previously, this was a one-objective problem in which the fitness function was a measure of the total potential energy of the molecule at each step. The possibility of adding a second objective was explored. The second objective could have been that the distances between atoms should not be excessively large or small. This would require the nearest neighbours of each atom to be identified in order to measure the distances between the relevant atoms. There were several ways in which this could have been performed, such as keeping a table of distances between pairs of atoms using their co-ordinates or by sampling the space around an atom and moving

outwards. Although this could have improved the performance of the algorithms, it was not implemented because it could have taken much more time than was available. Additionally, the original plan was to find out whether the structure could be determined without a priori knowledge of the system, and introducing a fitness function based on such reasoning could restrict the output from the EA too much. This restriction would not necessarily improve the results as the energy itself may not have been optimised any more than it would have been without the second objective. With more time available to study this topic, a second objective could be introduced.

Analysis of Per-Atom Exhaustive Test

During the EA tests, most of the atom movements were unfavourable apart from the permutations. Another problem which had arisen then was that the atoms tended to move too far apart. In an attempt to avoid these problems occurring in the PAET, one atom at a time was moved instead of moving all atoms at once. The plan was to start with an empty cell and add the atoms to it one at a time. Each time an atom was added, all the other atoms in the cell would move around one another, one at a time, and the energy would be tested at each step. Different distribution patterns were created to determine the atom's new position, including a Gaussian distribution, a uniform distribution and a radial distribution which was designed to avoid the central space in the range which was occupied by another atom. To create more favourable atom movements, standard deviations were defined using each atom's relative size.

The PAET was not completely exhaustive because it tested six relative positions on each inner loop atom movement rather than every possible position, but it was exhaustive in that it tested every combination of every atom relative to every other atom. A flow chart of the algorithm is given in figure 26.

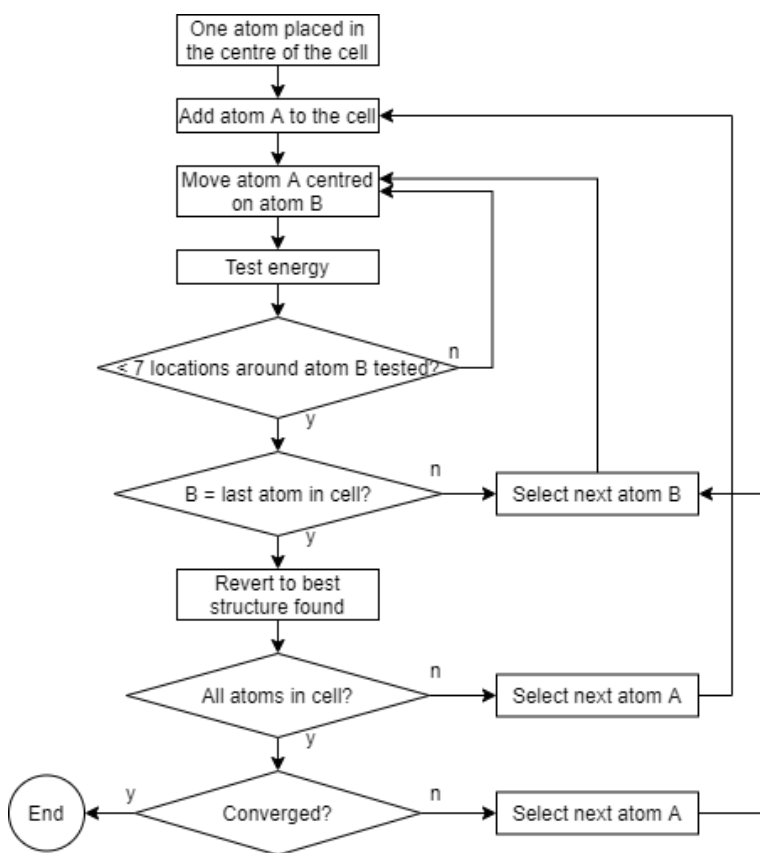


Figure 26 – Flow chart of the PAET algorithm.

New problems were introduced by the PAET. The execution time increased exponentially as the number of atoms (the input size) increased due to the nested loops. The atom movements were still mostly unfavourable. The thousands of calculations, array changes and function calls per iteration appeared to put a strain on the computer's hardware.

Multiprocessing was used to reduce the time spent in loops, which made up the majority of the execution time. Multiple, isolated versions of the PAET were run with multiprocessing, allowing a comparison to be made between the best results, as in the MMEA. This meant that fewer inner loop iterations needed to be performed as having a greater number of tests was believed to be more effective than having one test with a greater number of iterations (refer back to table two, in the sprint three section of this report).

Code optimisations such as scalar replacement and strength reduction were applied. The precision of some variables used in floating point calculations was reduced. Functionality for different parts of the program was moved into the same loop and information pertaining to different areas were stored in shared data structures in order to further reduce the number of loop iterations and array accesses required. This meant that good programming practices such as the single responsibility principle were disobeyed so it may not have been appropriate in a team of programmers who all needed to work on the same code. Concurrency was not explicitly implemented but it was anticipated that the compiler may apply it in some places.

An idea was tested which involved specifically moving atoms closer together or further apart as the algorithm progressed, rather than moving them randomly, but it was decided that this was too restrictive to the algorithm's 'learning' process, which was an important concept behind the optimisation. Evolutionary features from the MMEA were introduced into the PAET but were later removed as it was found to perform better without them and this was more true to the original concept of comparing an EA with an exhaustive algorithm. Unused evolutionary functions were moved to a shared 'algorithms' directory where algorithms could access them later if desired.

Measurements and Comparisons

Table four shows the energy values obtained by both algorithms for molecules of increasing sizes. The results show that the algorithms produced similar results for smaller molecules but the MMEA became better than the PAET as the number of atoms increased. The MMEA was able to reduce the energies of all the molecules and the PAET produced energies lower than the MMEA's starting energies for seven (58 %) of the molecules.

| Structural Formula | MMEA starting energy / eV | MMEA optimised energy / eV | PAET optimised energy / eV |
|-------------------------------------|---------------------------|----------------------------|----------------------------|
| H ₂ | 2.29 | 1.07 | 1.07 |
| H ₂ O | 2.98 | 1.90 | 2.01 |
| NH ₃ | 3.27 | 2.70 | 2.39 |
| CH ₄ | 2.99 | 2.23 | 2.46 |
| PCl ₅ | 3.24 | 2.95 | 2.85 |
| SF ₆ | 3.80 | 1.85 | 1.79 |
| C ₂ H ₆ | 2.70 | 2.00 | 3.84 |
| CH ₃ CH ₂ CN | 5.19 | 2.82 | 4.20 |
| CH ₃ CH ₂ CHO | 4.89 | 4.01 | 8.06 |
| CH ₃ COOCH ₃ | 5.63 | 4.50 | 7.36 |
| C ₆ H ₆ | 2.46 | 2.32 | 5.55 |

Table four – Optimised energies from the algorithms for molecules with increasing numbers of atoms. The best algorithm for each molecule is shaded in green. Note that the PAET did not have starting energies as it began with an empty cell.

Figure 27 displays the time measurements of the algorithms' execution times during sprint three. Being close to one, the R^2 values indicate that the exponential trendlines are a good fit. The exponential curve of the PAET was so steep at $0.4542e^{0.7766x}$ that systems with more than seven atoms were not tested due to concerns about the increase in temperature in the computer's hardware.

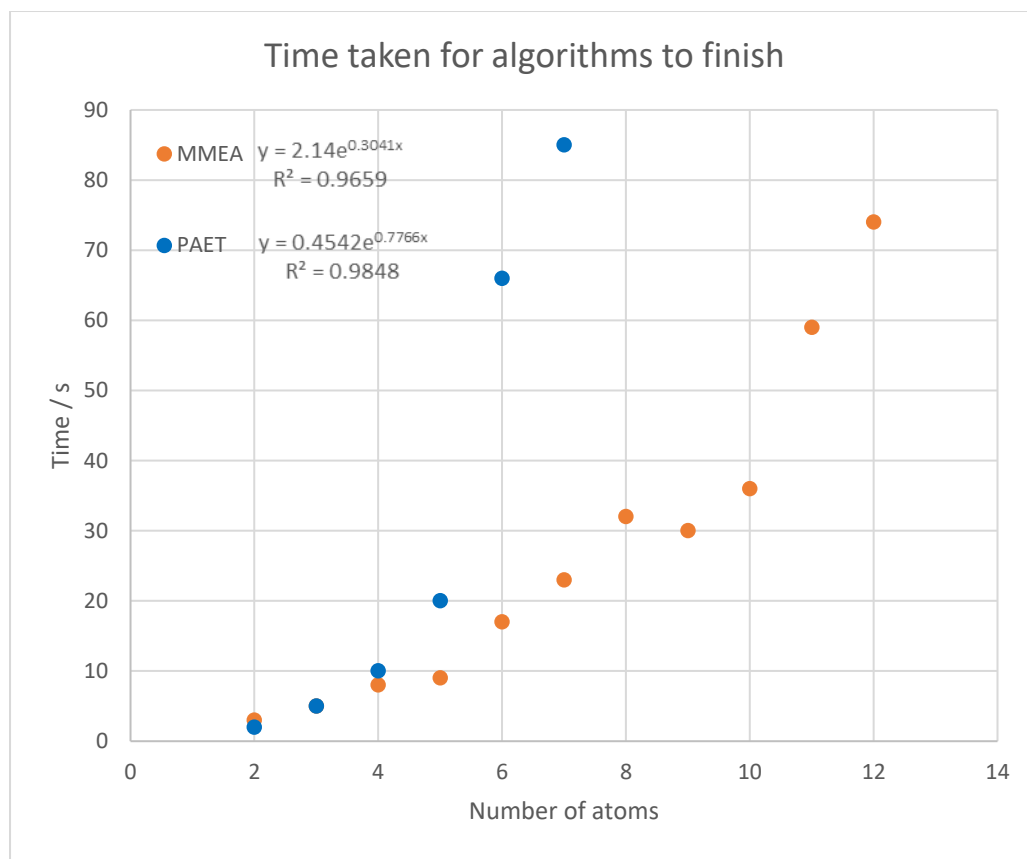


Figure 27 – Time measurements taken shortly after creating the algorithms.

Figure 28 shows the time measurements of the final versions of the algorithms. The trend of the MMEA appeared to have changed from exponential to linear, partly because of the cap applied to the maximum number of iterations, although if the limit was always met it may have needed to be increased. This time, the PAET was able to run with up to 13 atoms before the computer's hardware appeared to struggle, and still took less than a minute. The omission of plots and their data provided a faster execution time which became more noticeable as the number of atoms increased due to the resulting increase in data.

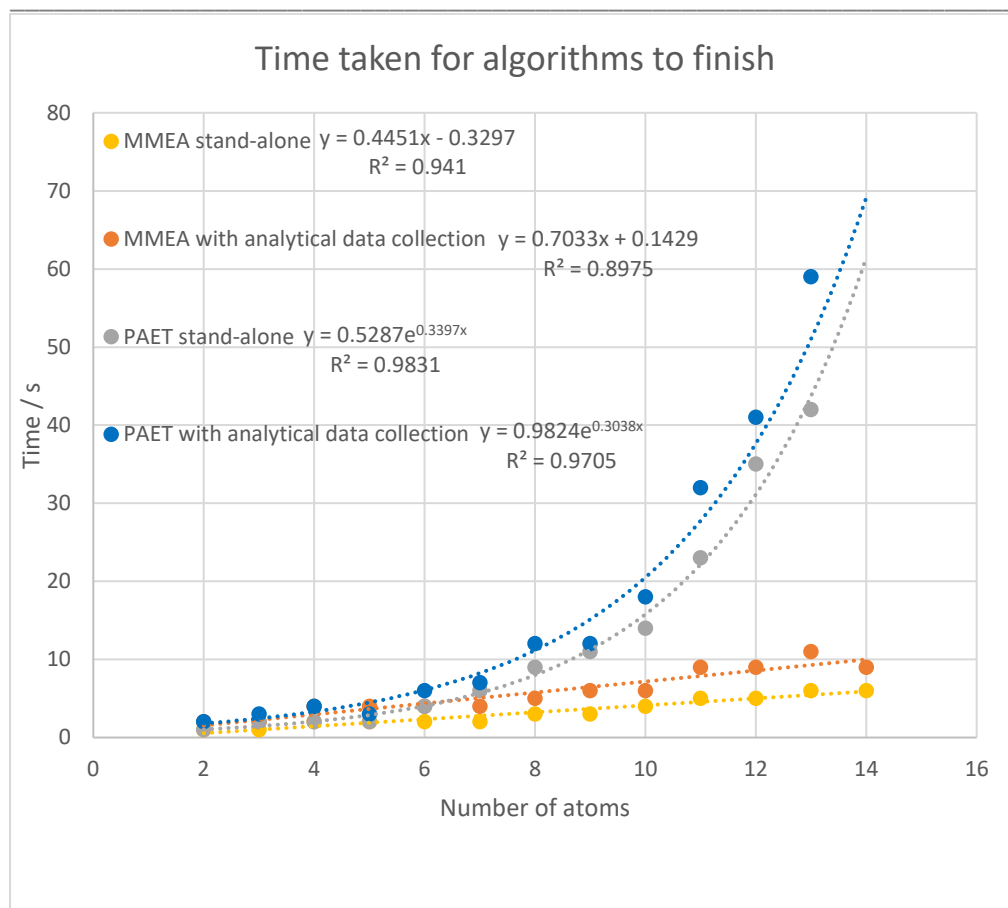


Figure 28 – Time measurements taken after applying optimisations and allowing the user to disable plots.

Figures 29 and 30 give comparisons of the earlier and later algorithms' time measurements. Figure 30 required a logarithmic time scale in order to fit the extrapolated trendlines on the chart because of the steepness of the initial PAET algorithm. It is evident that the optimisations were effective and the execution time of both algorithms was reduced. The MMEA was faster to execute than the PAET for 92 % of molecules which was all numbers of atoms except two.

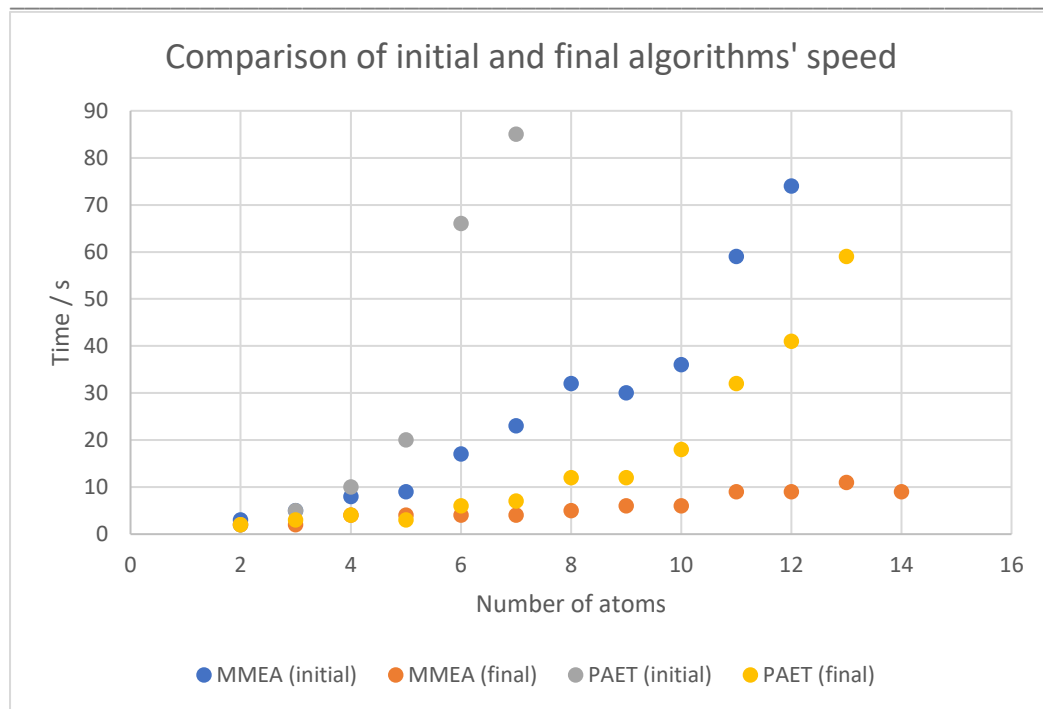


Figure 29 – Time measurements before and after improvements (default plot data enabled).

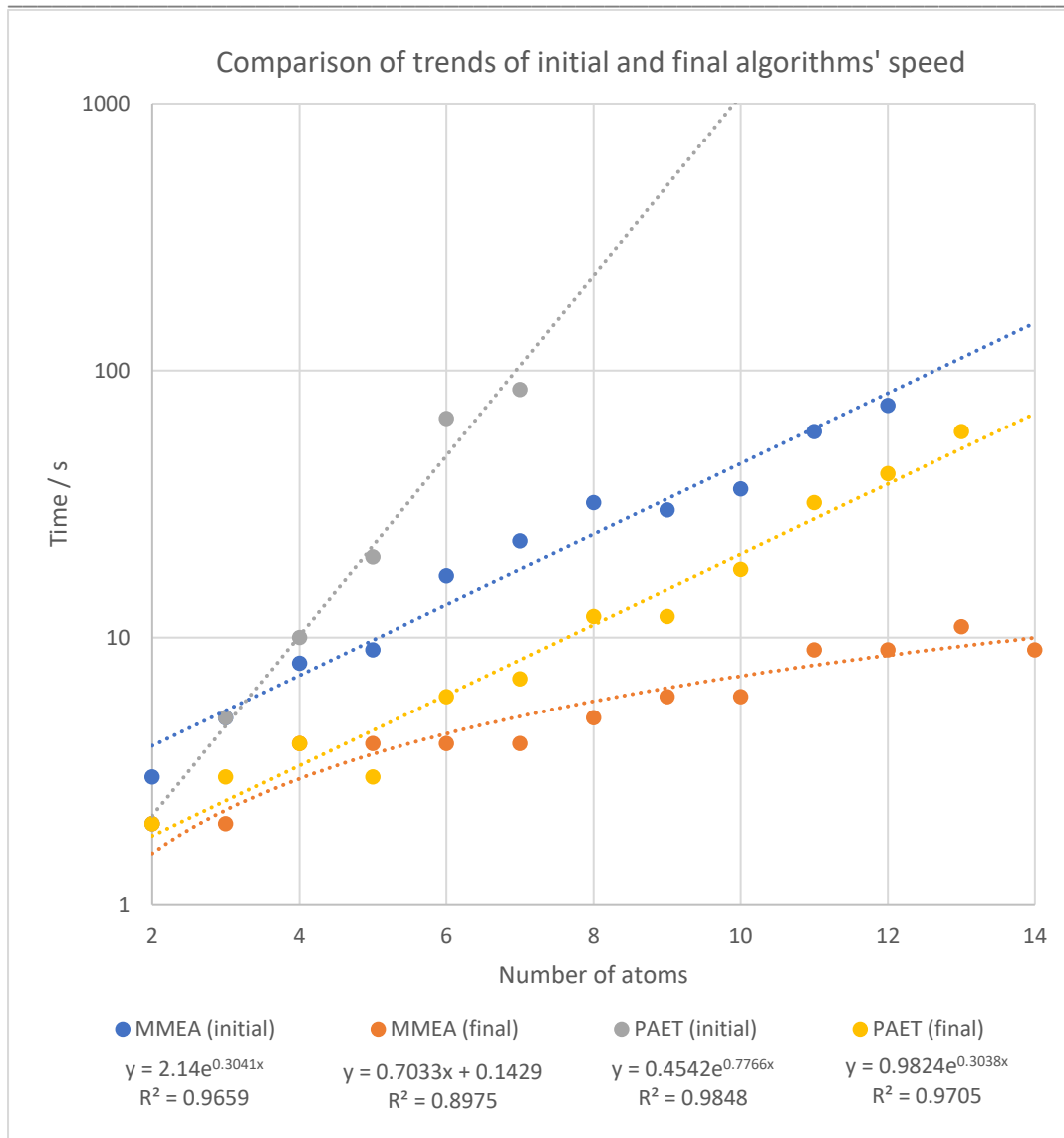


Figure 30 – Trends of the algorithms before and after optimisation.

Reflection on the Project

Sprints were successful overall and mostly went to plan, which indicates that the roadmap was realistic. There were often too many tasks to complete in each sprint but prioritising them meant that the core requirements were met at each sprint and it was better to have too many tasks than not enough to do. The sprints did not always finish on time according to the roadmap but a benefit of Agile is that it is flexible and allows some leeway. Sprint four had not gone to plan; it had been swapped with sprint five and extended to enable the GUI to be recreated using a new design. Although no serious problems were encountered as a result of changing sprint plans, it did take some time away from later sprints which could have been more productive. The amount of time required and its consequences on the timeline should have been considered more carefully and will be important to bear in mind in future projects. Work was continued consistently throughout the academic year on a schedule apart from during a period of illness in spring.

Difficulties with code structure were experienced, as this was the first large project undertaken by the developer. In previous, smaller projects, the importance of maintaining consistent programming practice had not been so evident, but the code for Geopt became complicated. It was planned that the MVC architecture would help to prevent confusion but it actually created confusion due to the chains of functions calling other functions in other parts of the code simply for the purpose of sticking to the MVC pattern. In the future, other architectures would be considered for a program like Geopt.

It may have been better to have studied a single molecule or a single EA rather than trying to generalise to a wide variety of molecules with many algorithms. Because the project required a UI to be created and usability tests to be conducted, this could have been difficult to implement and justify.

Both the algorithms were able to significantly reduce the energies of systems, which was the aim of the project. Execution times were improved without worsening the results obtained, using a combination of code optimisations, multiprocessing and algorithm re-design. Although energy values were optimised, Geopt was not able to accurately predict molecular geometry for most molecules due to its over-generalisation of atoms' interactions and effects on one another, using the EMT calculator on all the configurations and sample values for parameters of many elements. It was suspected that Geopt would provide better geometries if a DFT calculator had been used instead. A free DFT calculator had been found from the same developers who created the EMT package but it did not appear to work on either Windows or Linux even after setting the environment variables and writing a Python script to run it on Linux, as suggested by the developers. It was suspected that this was because it required a specific custom file type but the specifications or availability of this file type could not be found.

Conclusions

The EA and the exhaustive algorithm were both able to significantly reduce the energies of systems, which was the primary aim of the project. The EA was faster to execute than the exhaustive algorithm for 92 % of molecules and it produced better optimised structures for 70 % of systems. Execution times of both the algorithms were improved without worsening the results obtained. Although the energy values were improved, Geopt was usually unable to accurately predict molecular geometries.

Suggestions for Improvement

It was possible that Geopt could suggest different isomers of a molecule. If the project was to be further developed, this feature could be built upon. To do this, a way of deciding whether a structure was a chemically acceptable isomer would need to be introduced. A simple way to attempt this might be to consider the differences between the energy values of all the output structures. If two structures produced similar, low energy values but another structure bore a much higher value, it the two similar molecules would be more likely to be isomers and the other may be an anomaly.

Supervised learning could be added to the program. A human chemist could select the most favourable molecule output from a group of molecules proposed by the computer at different steps in the optimisation process, and this input could be used to inform the algorithm as it progressed. Ijzerman et al. (2005) created an interactive EA in which a chemist contributed to the fitness function, influencing the EA and using their knowledge to improve the outcome. Reinforcement learning could possibly be combined with this to keep and regularly update a record of the

best actions taken during the evolution process. Self-tuning, hyper-heuristic parameters could also have been added to improve the output. The standard deviation of the Gaussian distribution within which each atom was moved around the other atoms was altered as the PAET progressed, as it appeared to have a high impact on the best energy achieved. If the program was to be developed further, it may have been useful to create a function to constantly adapt this range.

More code optimisations such as concurrency, more design patterns, more carefully written algorithms and better chosen data types and structures could be employed to reduce the complexity and the execution time of the program. Low-level, compiled or scientific programming languages could be used inline, such as C, C++ or Fortran, for particularly intensive parts of the code, including loop bodies and floating-point calculations. A graphics processing unit (GPU) could have been utilised to improve parallelism in the code, particularly for the floating-point calculations performed with lists and other data structures. The data could have been separated or tiled and passed to the GPU to be quickly processed on many more threads. A drawback is that it would require more work to ensure that it could run on all computers, include those without a GPU.

References

- Addicoat, M. A., Brain, Z. E. (2010). Using a Meta-GA for parametric optimization of simple gas in the computational chemistry domain. *Genetic and evolutionary computation*. 12, p823-824.
- Alvarez, J. C., Campos, K. R., Coleman, P. J., Dreher, S. D., Garbaccio, R. M., Parmee, E. R., Terrett, N. K., Tillyer, R. D., Truppo, M. D.. (2019). The importance of synthetic chemistry in the pharmaceutical industry. *Science*. 363 (6424).
- Blomqvist, J., Castelli, I. E., Christensen, R., Duřak, M., Friis, J., Groves, M. N., Hammer, B., Hargus, C., Hermes, E. D., Jacobsen, K. W., Jennings, P. C., Jensen, P. B., Kermode, J., Kitchin, J. R., Kolsbjerg, E. L., Kubal, J., Kaasbjerg, K., Larsen, A. H., Lysgaard, S., Maronsson, J. B., Maxson, T., Mortensen, J. J., Olsen, T., Pastewka, L., Peterson, A., Rostgaard, C., Schiøtz, J., Schütt, O., Strange, M., Thygesen, K. S., Vegge, T., Vilhelmsen, L., Walter, M., Zeng, Z. (2017a). *Source code for ase.calculators.emt*. Available: https://wiki.fysik.dtu.dk/ase/_modules/ase/calculators/emt.html. Last accessed 17th April 2021.
- Blomqvist, J., Castelli, I. E., Christensen, R., Duřak, M., Friis, J., Groves, M. N., Hammer, B., Hargus, C., Hermes, E. D., Jacobsen, K. W., Jennings, P. C., Jensen, P. B., Kermode, J., Kitchin, J. R., Kolsbjerg, E. L., Kubal, J., Kaasbjerg, K., Larsen, A. H., Lysgaard, S., Maronsson, J. B., Maxson, T., Mortensen, J. J., Olsen, T., Pastewka, L., Peterson, A., Rostgaard, C., Schiøtz, J., Schütt, O., Strange, M., Thygesen, K. S., Vegge, T., Vilhelmsen, L., Walter, M., Zeng, Z. (2017b). The Atomic Simulation Environment—A Python library for working with atoms. *Condensed Matter*. 29.
- Chen, H. S., Chen, Y. H., Xu, G. J., Zhang, C. R., Zhang, H. L. (2007). Structures and properties of Si₆N₈ clusters: Genetic algorithm and density functional theory approach. *Journal of Molecular Structure-Theochem*. 805 (1-3), 161-166.
- Dawson, W., Gygi, F. (2014). Optimized scheduling strategies for hybrid density functional theory electronic structure calculations. *High Performance Computing, Networking, Storage and Analysis*. p685–692.
- Fan, Y., Ghosh, J., Marru, S., Pamidighantam, S., Singh, N., Vanomesslaeghe, K. (2011). Molecular parameter optimization gateway (ParamChem): workflow management through TeraGrid ASTA. *TeraGrid Conference: Extreme Digital Discovery*. (35), p1-8.
- Filipe, J. (2017). *Joel Filipe*. Available: <https://unsplash.com/photos/uHJubAEZklE>. Last accessed 30th March 2021.
- Grumbling, E., Horowitz, M. (2019). *Quantum Computing: Progress and Prospects*. Washington DC: The National Academies Press.
- Gueorguiev, V., Kuttel, M. (2016). Implementation, Validation and Profiling of a Genetic Algorithm for Molecular Conformational Optimization. *Annual Conference of the South African Institute of Computer Scientists and Information Technologists*. (16), p1-8.
- Hashimoto, M., Maeda, T., Shimazaki, T. (2015). Developing a high-performance quantum chemistry program with a dynamic scripting language. *Software Engineering for High Performance Computing in Computational Science and Engineering*. 3, p9-15.
- Ijzerman, A., Kok, J., Lameijer, E. W. (2005). The molecule evaluator: an interactive evolutionary algorithm for designing drug molecules. *Genetic and evolutionary computation*. 7, p1969–1976.
- Ishimoto, T., Nagashima, U., Teramae, H. (2011). Mixing parameters for geometry optimization using the Hamiltonian algorithm. *Theoretical Chemistry Accounts*. 130 (0), p671-678.

Jacobsen, K. W., Norskov, J. K., Stoltze, P. (1996). A semi-empirical effective medium theory for metals and alloys. *Surface Science*. 366, p394-402.

Johnston, R. L., Lloyd, L. D., Mortimer-Jones, T. V., Roberts, C. (2002). Geometry Optimisation of Aluminium Clusters Using a Genetic Algorithm. *ChemPhysChem*. 3 (5), p408-415.

Kim, D., Moon, S. (2006). Investigation of an effective medium theory for metallic periodic structures: a fitting-based approach. *Photonic Crystal Materials and Devices*. 6128 (4), p372.

Kitchen, J. (2012). *Modeling materials using density functional theory*. Carnegie Mellon University: Free Software Foundation.

Matthews, A., Pfau, D., Spencer, S. (2020). Ab initio solution of the many-electron Schrödinger equation with deep neural networks. *Physical Review Research*. 2.

Mohn, C. E. (2016). Predicting cation ordering in MgAl₂O₄ using genetic algorithms and density functional theory. *Materials and Manufacturing Processes*. 33 (2), p174-179.

Müller, V. C. (2020). Ethics of Artificial Intelligence and Robotics. In: Zalta, E. N. *Stanford Encyclopaedia of Philosophy*. Leeds: Stanford University.

Soni, D. (2018). *Introduction to Evolutionary Algorithms, Optimization by natural selection*. Available: <https://towardsdatascience.com/introduction-to-evolutionary-algorithms-a8594b484ac>. Last accessed 11th May 2021.

Shahab, M. L. (2019). New heuristic algorithm for traveling salesman problem. *Journal of Physics: Conference Series*. 1218

Appendices

Appendix A – Roadmap

| | Week 1 09-Oct | Week 2 16-Oct | Week 3 23-Oct | Week 4 30-Oct |
|--------------------|--|------------------|--|------------------|
| Milestones | Sprint zero | | Sprint one | |
| Plan | Planning Setup Literature review | | User story: I wish to create a molecule | |
| Test | Target user questionnaires | | | |
| Application | | | Allow user to select elements Create a logo Create UI Database Display the molecular formula | |

| | Week 5 06-Nov | Week 6 13-Nov | Week 7 20-Nov | Week 8 27-Nov |
|--------------------|--|------------------|---|------------------|
| Milestones | Sprint two | | Sprint three | |
| Plan | User story: I wish to predict a shape Update risk assessment for COVID Try an EA which makes many molecules and compares them. | | User story: I wish to use a different EA. Try an EA which alters one atom at a time. | |
| Test | See if the EA works to decrease energy See if the molecule's shape is displayed in the cell properly | | Try to get a better structure prediction than before. Test num iterations, E calcs, array changes. | |
| Application | Make the EA Improve exception handling Choose EA parameters Create models of the molecule | | Make the EA. Try without crossover and remove all parents. Move from test area to application. | |

| | Week 9 04-Dec | Week 10 11-Dec |
|-------------------|---|-------------------|
| Milestones | Sprint four | |
| Plan | User story: I wish to view molecule, info & PES | |
| Test | | |
| | Potential energy surface plot | |

| | Week 11 | 18-Dec | | Week 12 | 15-Jan |
|--------------------|---------------------------------|-------------|-------------------|---------------------------------|-------------|
| Milestones | Sprint | five | Christmas holiday | Sprint | five |
| | | | Christmas holiday | | |
| Plan | Testing | | Christmas holiday | Testing | |
| | Feedback & changes | | Christmas holiday | Feedback & changes | |
| | Fixes & improvements | | Christmas holiday | Fixes & improvements | |
| Test | Usability tests | | Christmas holiday | Usability tests | |
| | | | Christmas holiday | | |
| Application | | | Christmas holiday | Act on user feedback | |
| | | | Christmas holiday | Fix errors & bugs | |
| | | | Christmas holiday | | |
| | | | Christmas holiday | | |
| | | | Christmas holiday | | |

| | Week 15 | 05-Feb | Week 16 | 12-Feb | Week 17 | 19-Feb | Week 18 | 26-Feb |
|------------|--------------------|--------|---------|--------------|------------------------|--------|-------------|--------|
| Milestones | Sprint seven | | | Sprint eight | | | Finish code | |
| Plan | Testing | | | | Final fixes & clean up | | | |
| | Feedback & changes | | | | Start writing report | | | |

| | | |
|--------------------|--|--|
| Test | Usability tests | |
| Application | Make changes from user feedback Find and fix bugs Tidy up | Make changes from user feedback Find and fix bugs Tidy up |

| | | |
|--------------------|--|-----------------------------------|
| | Week 19 05-Mar | Week 20 12-Mar |
| Milestones | Sprint | nine |
| | | Submit showcase materials! |
| Plan | Showcase materials Start writing report | |
| Test | | |
| Application | | |

Appendix B – Questionnaire Information Sheet

UNIVERSITY OF PLYMOUTH

FACULTY OF SCIENCE AND ENGINEERING

RESEARCH INFORMATION SHEET

Name of Principal Investigator

Sophie Turner

Title of Research

Molecular geometry prediction software feedback from scientists

Aim of research

To get the opinions of chemists and other relevant scientists about what they want from software which predicts the geometry of molecules.

Description of procedure

Answer some questions about their experiences of similar software.

Description of risks

None.

Benefits of proposed research

Help a computer scientist to design an appropriate application for natural scientists to use.

Right to withdraw

You can withdraw from the research at any time and can request that your data be destroyed by emailing sophie.turner@plymouth.ac.uk.

If you are dissatisfied with the way the research is conducted, please contact the principal investigator in the first instance: sophie.turner@plymouth.ac.uk

If you feel the problem has not been resolved please contact the secretary to the Faculty of Science and Engineering Research Ethics & Integrity Committee: Mrs Paula Simson 01752 584503.

Appendix C – Target User Questionnaire

UNIVERSITY OF PLYMOUTH

FACULTY OF SCIENCE AND ENGINEERING

Consent Form

CONSENT TO PARTICIPATE IN RESEARCH PROJECT / PRACTICAL STUDY

Name of Principal Investigator

Sophie Turner

Title of Research

Molecular geometry prediction software feedback from scientists

Brief statement of purpose of work

I am a final year student of computer science at the University of Plymouth and I am creating a chemistry application for my dissertation project. I am looking for feedback and recommendations from chemists and other relevant scientists regarding the design of this application. These forms are anonymous but will be discussed in, and appended to, my dissertation report. Your name will **not** be included in this. You can request that your answers be deleted and not included in the project by emailing me at sophie.turner@plymouth.ac.uk

The objectives of this research have been explained to me.

I understand that I am free to withdraw from the research at any stage, and ask for my data to be destroyed if I wish.

I understand that my anonymity is guaranteed, unless I expressly state otherwise.

I understand that the Principal Investigator of this work will have attempted, as far as possible, to avoid any risks, and that safety and health risks will have been separately assessed by appropriate authorities (e.g. under COSHH regulations)

Under these circumstances, I agree to participate in the research.

Name:

Signature:

Date:

Molecular geometry prediction software feedback from scientists

1. Which science are you mostly involved in?
2. What is your role in this science (e.g. undergraduate, technician, etc)?

3. Please name all the software you have used for chemical structure/geometry modelling, e.g. GaussView, PubChem, Avogadro, etc. If you have never used software for this purpose, please skip to question 8.
4. Regarding the most recent time you used this software, please describe what you used the software for.
5. Why did you choose this software over others?
6. What do you consider to be the best things about this software?
7. What do you consider to be the worst things about this software?
8. In your opinion, what would be the most important features of software for predicting the geometric structure of a theoretical molecule? What would you want to be able to do, as a user of this software?
9. An evolutionary algorithm is a computer algorithm which can be used to find solutions to problems by following these steps:
 - **Start with an initial estimate of a solution to a problem**, e.g. the positions of atoms in a molecule.
 - **Alter variables which affect this solution**, e.g. the distance between the atoms.
 - **Pass these variables to a function**, e.g. an energy calculation, **which returns an output – another possible solution**, e.g. the energy of the system.

- **Compare this solution to the previous solutions.**
- **Choose the best solution**, e.g. the lowest energy of the system.
- **Repeat these steps until reaching an optimal solution**, e.g. the bond lengths which create the lowest net force on each atom.

Algorithms like this can be used to predict geometric properties of molecules, such as bond lengths and bond angles. As a user of this software, do you think it would be useful to be able to view this process and adjust parts of the algorithm, such as which variables to change, or would you prefer it to be a 'black box' that worked behind the scenes and just showed you the output?

10. Was this questionnaire easy to understand and fill out? Is there anything that you think should be changed about it?

11. Is there anything else you would like to mention which could be useful for this project?

Thank you for your time. Please return your completed form to sophie.turner@plymouth.ac.uk

Appendix D – Participant A's Responses to Target User Questionnaire

1. Which science are you mostly involved in?

Chemistry

2. What is your role in this science (e.g. undergraduate, technician, etc)?

BSc (Hons) Chemistry

3. Please name all the software you have used for chemical structure/geometry modelling, e.g. GaussView, PubChem, Avogadro.cc, etc. If you have never used software for this purpose, please skip to question 8.

Chemdraw, GaussView

4. Regarding the most recent time you used this software, please describe in more detail what you used the software for.

GaussView – Approximate energies of molecular orbitals

Chemdraw – Model skeletal formula in organic chemistry and interactions between molecules

5. Why did you choose this software over others?

Chemdraw – free licence

GaussView – free licence

6. What do you consider to be the best things about this software?

Chemdraw – User-friendly, made by chemists so valencies and structures are sensible.

GaussView – Fast, user-friendly, 3D representation of orbitals on structure given.

7. What do you consider to be the worst things about this software?

Chemdraw – Sometimes incorrectly corrects valencies when working with unusual structures so it can be difficult when working with novel systems

GaussView – Only allows energies to be computed for single systems. Small structural size limits to reduce computational expense.

8. In your opinion, what would be the most important features of software for predicting the geometric structure of a theoretical molecule? What would you want to be able to do, as a user of this software?
- Easy to use
 - Designed by chemists
 - 360 degree view of finished molecule
 - Able to model interactions between different systems of molecules
9. An evolutionary algorithm is a computer algorithm which can be used to find solutions to problems by following these steps:
- **Start with an initial estimate of a solution to a problem or function**, e.g. the positions of atoms in a molecule.
 - **Alter variables which affect this solution**, e.g. the distance between the atoms.
 - **Pass these variables to a function**, e.g. an energy calculation, **which returns an output – another possible solution**, e.g. the energy of the system.
 - **Compare this solution to the previous solutions.**
 - **Choose the best solution**, e.g. the lowest energy of the system.
 - **Repeat these steps until reaching an optimal solution**, e.g. the bond lengths which create the lowest net force on each atom.

Algorithms like this can be used to predict geometric properties of molecules, such as bond lengths and bond angles. As a user of this software, do you think it would be useful to be able to view this process and adjust parts of the algorithm, such as which variables to change, or would you prefer it to be a 'black box' that worked behind the scenes and just showed you the output?

View and adjust, with option to make output simpler when required

10. Was this questionnaire easy to understand and fill out? Is there anything that you think should be changed about it?

Good questionnaire.

11. Is there anything else you would like to mention which could be useful for this project?

An option to analyse the interactions of structures in different solvents would be good.

Appendix E – Participant B's Responses to Target User Questionnaire

1. Which science are you mostly involved in?

Biology

2. What is your role in this science (e.g. undergraduate, technician, etc)?

Postgraduate

3. Please name all the software you have used for chemical structure/geometry modelling, e.g. GaussView, PubChem, Avogadro.cc, etc. If you have never used software for this purpose, please skip to question 8.

PubChem

4. Regarding the most recent time you used this software, please describe in more detail what you used the software for.

Visualising chemical structures.

5. Why did you choose this software over others?

I was instructed to as a learning exercise.

6. What do you consider to be the best things about this software?

7. What do you consider to be the worst things about this software?

8. In your opinion, what would be the most important features of software for predicting the geometric structure of a theoretical molecule? What would you want to be able to do, as a user of this software?

Visualise the structure and view reasoning behind the prediction.

9. An evolutionary algorithm is a computer algorithm which can be used to find solutions to problems by following these steps:

- **Start with an initial estimate of a solution to a problem**, e.g. the positions of atoms in a molecule.
- **Alter variables which affect this solution**, e.g. the distance between the atoms.
- **Pass these variables to a function**, e.g. an energy calculation, **which returns an output – another possible solution**, e.g. the energy of the system.
- **Compare this solution to the previous solutions.**
- **Choose the best solution**, e.g. the lowest energy of the system.
- **Repeat these steps until reaching an optimal solution**, e.g. the bond lengths which create the lowest net force on each atom.

Algorithms like this can be used to predict geometric properties of molecules, such as bond lengths and bond angles. As a user of this software, do you think it would be useful to be able to view this process and adjust parts of the algorithm, such as which variables to change, or would you prefer it to be a 'black box' that worked behind the scenes and just showed you the output?

I would like to be able to view this process if I were using the software.

10. Was this questionnaire easy to understand and fill out? Is there anything that you think should be changed about it?

Yes, however as someone who hasn't used this kind of software much I found it difficult to answer the questions.

11. Is there anything else you would like to mention which could be useful for this project?

Appendix F – Project Initiation Document

COMP3000

Computing Project

2020/2021

Project Title

Molecular Geometry Optimisation Using Evolutionary Computation

Links

Source code: <https://github.com/Squidgeypea/SophieCOMP3000>

Backlog : <https://tasks.office.com/live.plymouth.ac.uk/en-GB/Home/Planner/#/plantaskboard?groupId=94cc8cbf-c90e-473e-a3f2-7d7d5dee52d9&planId=VmtMkciqc0GG6F--BwFrqpYAESf1>

Project Vision

This program is for chemists and physicists who want to estimate and view the structures of molecules without spending hours doing calculations or using a supercomputer. Geopt is a program which uses machine learning to predict the shapes of theoretical molecules.

Risk Plan

| Ref | Risk event | Likelihood | Impact | Exposure | Plan |
|-----|------------|------------|--------|----------|------|
| | | 1=low | 1=low | 1=min | |
| | | 3=high | 3=high | 9=max | |



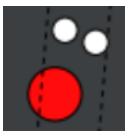


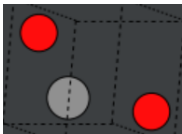




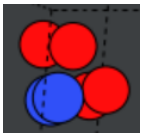
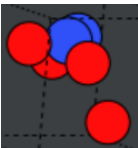


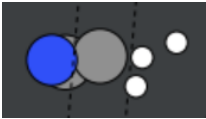
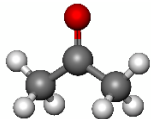
| | | | | | |
|----|---|---|---|---|--|
| R1 | Calculations too computationally expensive | 2 | 2 | 4 | Use Python packages for calculations. Choose appropriate methods. Simplify calculations. Try multithreading/GPU options etc. |
| R2 | Evolutionary algorithms take too long | 3 | 2 | 6 | Use fewer iterations. Use different technique, selection criteria etc. Make code simpler. |
| R3 | Programming difficulties/lack of knowledge | 3 | 2 | 6 | Have regular meetings with David. Do lots of research and practice. |
| R4 | Too much to do/not finish on time | 3 | 3 | 9 | Keep working throughout the year. Adjust plans if necessary. Focus on most important things first. Use Agile. |
| R5 | Unable to get desired results from algorithms | 2 | 1 | 2 | Alter mutations etc. Analyse and test algorithms. Do plenty of research. |
| R6 | Coronavirus or illness | 2 | 3 | 6 | Know the University's EC policies. Work from home where possible. |



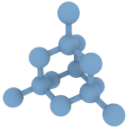
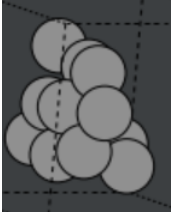
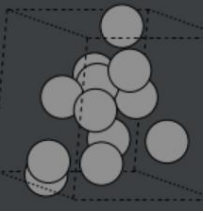
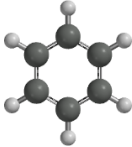
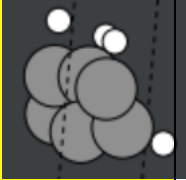

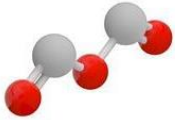
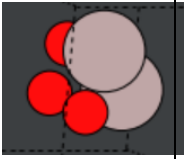
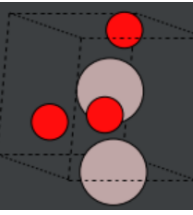
Keywords

Molecular, chemical, atoms, molecules, geometry, energy, optimisation, minimisation, Python, simulation, evolutionary, machine learning.

Appendix G – Performance Comparison

Excellent results are shown in yellow. Overall, the Many Molecules EA performed better.

| Molecule (many of these can rotate at each bond so often won't appear like their pictures) | EA Many Molecules Average best energy / eV | Per Atom Average best energy / eV | EA Many Molecules Average shape | Per Atom Average shape | EA Many Molecules Approx. average time taken / seconds | Per Atom Approx. average time taken / seconds |
|---|---|-----------------------------------|--|--|---|---|
| Water  | 1.900 | 2.531 | Excellent  | Good  | 2 | 2 |
| Carbon dioxide  | 0.9230 | 0.9720 | Poor  | Poor  | < 1 | 1 |
| Methane  | 2.993 | 3.890 | Good  | Poor  | 1 | 3 |
| Dinitrogen tetroxide  | 3.418 | 6.617 | Good  | Poor  | 10 | 4 |
| Acetonitrile  | 1.827 | 4.204 | Poor  | Poor  | 1 | 4 |
| Acetone  | 3.455 | 27.69 | Good | Good | 12 | 20 |

| | | | | | | |
|--|-------|-------|---|--|----|----|
| | | |  |  | | |
| Lattice of 12 carbons  | 3.020 | 11.32 | Good  | Poor  | 2 | 48 |
| Benzene  | 2.416 | 69.32 | Excellent  | Good  | 39 | 50 |
| Aluminium oxide  | 4.467 | 4.139 | Good  | Good  | 8 | 3 |

Appendix H – First Usability Test Questionnaire

UNIVERSITY OF PLYMOUTH

FACULTY OF SCIENCE AND ENGINEERING

Consent Form

CONSENT TO PARTICIPATE IN RESEARCH PROJECT / PRACTICAL STUDY

Name of Principal Investigator

Sophie Turner

Title of Research

Molecular geometry prediction software feedback from scientists

Brief statement of purpose of work

I am a final year student of computer science at the University of Plymouth and I am creating a chemistry application for my dissertation project. I am looking for feedback regarding the design of this application. These forms are anonymous but will be discussed in, and appended to, my dissertation report. Your name will **not** be included in this. You can request that your answers be deleted and not included in the project by emailing me at sophie.turner@plymouth.ac.uk

By participating in this usability test, you agree that:

- The objectives of this research have been explained to you.
- You understand that you are free to withdraw from the research at any stage, and ask for your data to be destroyed if you wish.
- You understand that your anonymity is guaranteed, unless you expressly state otherwise.

-
- You understand that the Principal Investigator of this work will have attempted, as far as possible, to avoid any risks, and that safety and health risks will have been separately assessed by appropriate authorities (e.g. under COSHH regulations)
 - Under these circumstances, you agree to participate in the research.

For each task, please comment on whether the program worked as expected and how easy it was to use. Please also mention any bugs you find.

Date:

Task 1. Please select up to eight atoms from any combination of H, C, O, N and/or Al. Ions and other elements are not yet supported by this program. You can select more than eight atoms if you wish to do so, but the algorithm may take some minutes to perform its calculations as the time taken rises exponentially with the size of the molecule.

Comments:

Task 2. View information about the structures and potential energy surfaces.

Comments:

Task 3. Create another system of atoms.

Comments:

Task 4. View information about the structures and potential energy surfaces.

Comments:

Did you find this form comprehensive and easy to complete?

Additional comments and suggestions:

Thank you for your time.

Appendix I – Participant C's Responses to First Usability Test Questionnaire

Date:

10/02/2021

Task 1. Please select up to eight atoms from any combination of H, C, O, N and/or Al. Ions and other elements are not yet supported by this program. You can select more than eight atoms if you wish to do so, but the algorithm may take some minutes to perform its calculations as the time taken rises exponentially with the size of the molecule.

Comments:

Program worked? Y

Easy to use? Y

I used HCN as the example. HCN is a linear molecule with a triple bond between C and N. The results were unexpected; the structure with the H atom closer to both C and N was lower in energy than the more linear structure with 180 degree bond angles between H, C and N. The program appears to only be able to anticipate singly bonded molecules.

Task 2. View information about the structures and potential energy surfaces.

Comments:

Program worked? Y

Easy to use? Y

Structure information is excellent. It is easy to see the molecules in 3D. It might be nice to be able to rotate the structure diagrams to see more detail about the arrangement of atoms. The PES is good. Can't understand the 'All positions tested' part. The colours all overlap.

Task 3. Create another system of atoms.

Comments:

This time I used CH₂N. The structures were clear, and the lowest energy structure was again unexpected.

Task 4. View information about the structures and potential energy surfaces.

Comments:

It's still difficult to interpret the positions tested and the 3D plot is difficult to read when there are more atoms.

Did you find this form comprehensive and easy to complete?

Yes

Additional comments and suggestions:

You could add a separate yes/no section to the feedback form to make it easier to interpret whether the form was user friendly.

Appendix J – Participant D's Responses to First Usability Test Questionnaire

Date: 25/1/21

Task 1. Please select up to eight atoms from any combination of H, C, O, N and/or Al. Ions and other elements are not yet supported by this program. You can select more than eight atoms if you wish to do so, but the algorithm may take some minutes to perform its calculations as the time taken rises exponentially with the size of the molecule.

Comments: Looks good. All worked fine and easy to understand the UI. User might think the program has crashed or isn't responding because it can take a long time so you could tell the user to wait or have something saying 'calculating...'.

Task 2. View information about the structures and potential energy surfaces.

Comments: I didn't realise at first that I could click the button for more info. Maybe have a 'more info' hint.

Task 3. Create another system of atoms.

Comments: All good.

Task 4. View information about the structures and potential energy surfaces.

Comments: All good. Well done.

Did you find this form comprehensive and easy to complete? Yes

Additional comments and suggestions: Hard to understand if you are not a chemist. I can't comment on that part of it but the UI is good.

Appendix K – Participant E's Responses to First Usability Test Questionnaire

Date: 25/1/21

Task 1. Please select up to eight atoms from any combination of H, C, O, N and/or Al. Ions and other elements are not yet supported by this program. You can select more than eight atoms if you wish to do so, but the algorithm may take some minutes to perform its calculations as the time taken rises exponentially with the size of the molecule.

Comments:

It worked as expected.

Task 2. View information about the structures and potential energy surfaces.

Comments:

Very good UI and nice graphs. Don't understand them though.

Task 3. Create another system of atoms.

Comments:

I chose N N O O and it said it was not a valid molecule. Didn't work.

Task 4. View information about the structures and potential energy surfaces.

Comments:

Did you find this form comprehensive and easy to complete?

Yes.

Additional comments and suggestions:

Perhaps you could let the user choose how long it takes, how many iterations etc.

Appendix L – User Guide

Molecular Geometry Optimisation Using Evolutionary Computation

Vision

This program is for chemists and physicists who want to estimate and view the structures of molecules without spending hours doing calculations or using a supercomputer. Geopt is a program which uses machine learning to predict the shapes of theoretical molecules.

User Manual

Intended use of Geopt & limitations

Geopt uses an Effective Medium Theory (EMT) energy calculator to search for optimal geometric configurations of molecules. The EMT calculator is provided by The Atomic Simulation Environment (ASE) and has been adapted by Sophie Turner for this use. The EMT calculator is unable to work with ions or double or triple bonds. This is because it does not consider valency or any bonds between atoms, only the forces on atoms caused by the presence of other atoms nearby. This is why it favours some geometries which are not chemically realistic. EMT was originally only intended for use with solid metal structures.

Software required

The following software is required to use Geopt.

Python 3

Windows: <https://www.python.org/downloads/>

Linux command: `sudo apt-get install python3`

ASE

Pip command: `pip install --upgrade --user ase`

Linux command: `sudo apt-get install python3-ase`

ttkwidgets

Pip command: `pip install ttkwidgets`

Linux commands: `sudo add-apt-repository ppa:j-4321-i/ttkwidgets`

`sudo apt-get update`

`sudo apt-get install python3-ttkwidgets`

Creating a molecule

Choose elements by either typing the molecular formula or selecting elements from the periodic table. Use correct capitalisation for element symbols.

Many-molecule evolution

The many-molecule algorithm is an evolutionary algorithm which creates a population of versions of the molecule with different configurations, applies mutations to generations of versions and selects the best versions based on the lowest total potential energy. It is recommended for molecules with two to twelve atoms.

Per-atom exhaustive test

The per-atom exhaustive test is an algorithm which moves each atom in the molecule around each other atom, constantly testing for the configuration with the lowest total potential energy. The amount of processing required is exponential to the number of atoms and it is recommended for molecules with two to six atoms.

Periodic boundary conditions

Periodic boundary conditions are used to approximate a larger system from repeating the unit cell.

Population size

The population size is the number of versions of the molecule that will exist at the same time at each step in the algorithm, in each process. A larger population makes the algorithm take longer to complete but may offer more potential configurations.

Parallel processes

The number of parallel processes is how many times the entire algorithm will run. It is recommended that this number is equal to the number of CPU cores in your computer. This is detected and set by default. If the program is unable to detect the number of CPU cores in your computer it will set the default number to one, which you can change manually. A higher number may offer more potential configurations, but you are advised to avoid setting this to a higher number than the number of CPU cores for molecules with more than two atoms.

Plot data size limit

This is the maximum number of data points in the datasets used to create plots. A smaller number will speed up execution time and may also create clearer graphs.

Radial mutation distribution

This alters the position of atoms and moves them randomly according to a distribution around other atoms or themselves. The distribution size determines how far apart atoms can move from one another. Moving atoms too far apart can cause the energy calculator to treat each atom as a standalone system. Moving atoms too close together can result in high energies and repulsion.

Permutation

This swaps the positions of atoms in the molecule during an evolutionary algorithm.

Crossover

This combines positions of atoms from two parent molecules to form a new configuration during an evolutionary algorithm.

References & Other Documentation

Atomic Simulation Environment

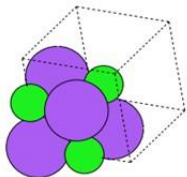
<https://wiki.fysik.dtu.dk/ase/>

Effective Medium Theory testing calculator source

<https://wiki.fysik.dtu.dk/ase/modules/ase/calculators/emt.html#EMT>

Authors of the research and creators of the above packages

Ask Hjorth Larsen, Jens Jørgen Mortensen, Jakob Blomqvist, Ivano E. Castelli, Rune Christensen, Marcin Dułak, Jesper Friis, Michael N. Groves, Bjørk Hammer, Cory Hargus, Eric D. Hermes, Paul C. Jennings, Peter Bjerre Jensen, James Kermode, John R. Kitchin, Esben Leonhard Kolsbjerg, Joseph Kubal, Kristen Kaasbjerg, Steen Lysgaard, Jón Bergmann Maronsson, Tristan Maxson, Thomas Olsen, Lars Pastewka, Andrew Peterson, Carsten Rostgaard, Jakob Schiøtz, Ole Schütt, Mikkel Strange, Kristian S. Thygesen, Tejs Vegge, Lasse Vilhelmsen, Michael Walter, Zhenhua Zeng, Karsten Wedel Jacobsen. The Atomic Simulation Environment—A Python library for working with atoms J. Phys.: Condens. Matter Vol. 29 273002, 2017



Geopt

Molecular Geometry Optimisation Using Evolutionary Computation

Overview

Geopt is a program for chemists and physicists who wish to predict and view the geometric structure of a theoretical system of atoms without the use of a super-computer or any bonding information.

Objectives

- Predict the structure of a molecule.
- Find the lowest total potential energy.
- Create an interactive evolutionary algorithm.
- Create an interactive exhaustive algorithm.
- Gather and display analytical data.

Methods

Success criteria:

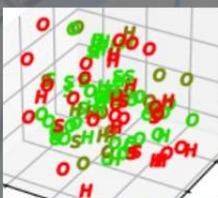
- Lowest possible total potential energy.
- No excessively large distances between atoms.

Aided by:

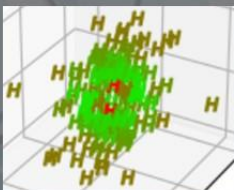
- Atomic Simulation Environment (ASE) - models.
- Effective Medium Theory (EMT) - calculator.
- Extensible Markup Language (XML) - atom data.
- Tkinter – user interface.

Results

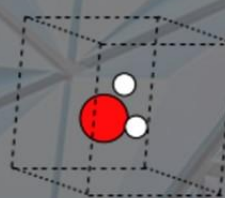
Co-ordinates tested to model sulfuric acid



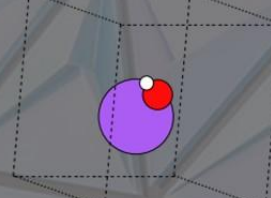
Symmetric mutation distribution of dihydrogen



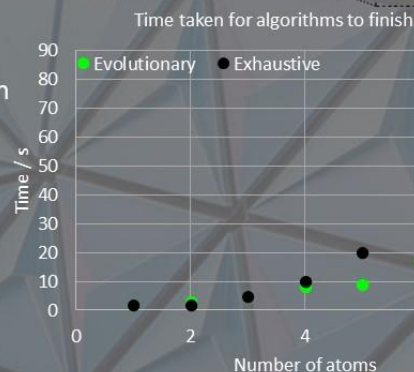
Water prediction (evolutionary)



Sodium hydroxide prediction (exhaustive)



- ✓ The evolutionary algorithm was fastest to execute.
- ✓ Accuracy depended on the type of molecule chosen.



Potential energy surfaces discovered during run-time

