

Molecular Geometry Optimisation

Using Evolutionary Computation

Dissertation Project

Developer: Sophie Turner

Supervisor: Dr. David Walker

May 2021

Contents

[Abstract 2](#_Toc69384078)

[Introduction 2](#_Toc69384079)

[Links to Work 2](#_Toc69384080)

[Background 3](#_Toc69384081)

[Aim 3](#_Toc69384082)

[Literature and Software Review 3](#_Toc69384083)

[Project Plan 9](#_Toc69384084)

[Roadmap & Sprint plans 9](#_Toc69384085)

[Risk Assessment 10](#_Toc69384086)

[Legal, Social, Ethical and Professional Matters 11](#_Toc69384087)

[Implementation 12](#_Toc69384088)

[Design Patterns & Programming Conventions 12](#_Toc69384089)

[Technologies Used 12](#_Toc69384090)

[Energy Calculators 13](#_Toc69384091)

[User Interface 13](#_Toc69384092)

[Sprint Zero – Planning 13](#_Toc69384093)

[Sprint One – Building a Molecule 13](#_Toc69384094)

[Sprint Two – Predicting a Shape 15](#_Toc69384095)

[Sprint Three – Other Algorithms 16](#_Toc69384096)

[Sprint Four – Viewing Results 18](#_Toc69384097)

[Sprint Five – Testing 25](#_Toc69384098)

[Sprint Six – Interactivity 29](#_Toc69384099)

[Sprint Seven – Testing 30](#_Toc69384100)

[Sprint Eight – Final Touches 31](#_Toc69384101)

[Sprint Nine – Showcase Materials and Report 31](#_Toc69384102)

[Algorithms 31](#_Toc69384103)

[Many-Molecule Evolutionary Algorithm 31](#_Toc69384104)

[Per-Atom Exhaustive Test 34](#_Toc69384105)

[Other Algorithms 35](#_Toc69384106)

[User Feedback 35](#_Toc69384107)

[Discussion & Evaluation 35](#_Toc69384108)

[Conclusions 37](#_Toc69384109)

[Suggestions 37](#_Toc69384110)

[References 37](#_Toc69384111)

[Appendices 38](#_Toc69384112)

[Appendix x – Questionnaire Information Sheet 38](#_Toc69384113)

[Appendix x – Target User Questionnaire 39](#_Toc69384114)

[Appendix x – Participant A’s Responses to Target User Questionnaire 43](#_Toc69384115)

[Appendix x – Participant B’s Responses to Target User Questionnaire 45](#_Toc69384116)

[Appendix x – First Usability Test Questionnaire 47](#_Toc69384117)

[Appendix x – Participant C’s Responses to First Usability Test Questionnaire 49](#_Toc69384118)

[Appendix x – Participant D’s Responses to First Usability Test Questionnaire 50](#_Toc69384119)

[Appendix x – Participant E’s Responses to First Usability Test Questionnaire 50](#_Toc69384120)

[Appendix x – Second Usability Test Questionnaire 51](#_Toc69384121)

[Appendix x – Participants’ Responses to Second Usability Test Questionnaire 51](#_Toc69384122)

[Appendix x – Project Initiation Document 51](#_Toc69384123)

[Appendix x – Performance Comparison 53](#_Toc69384124)

[Appendix x – Roadmap 54](#_Toc69384125)

# Abstract

The aim of this project is to create an evolutionary algorithm to find energetically optimal configurations of atoms in space. The program is for chemists and physicists whowish to estimate and view the structures of molecules without spending hours doing calculations or using a supercomputer. The program’s name is Geopt and it uses an evolutionary algorithm to predict the shapes of theoretical molecules.

# 

# Introduction

## Links to Work

YouTube video of demonstration of the program at sprint five: <https://youtu.be/ylv4J85m95Y>

YouTube video of demonstration of the program at sprint seven: <https://youtu.be/llvHbYyEO6Q>

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

YouTube video of demonstration of the final product:

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

Microsoft Planner:

Showcase:

## 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 bonds, considering 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 the 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 a desktop computer without the need for a supercomputer, and it 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 literature review and discussion and analysis sections of this report.

Evolutionary algorithms (EAs) are nature-inspired, heuristic techniques to solve problems. The underlying concept is that a population of potential solutions is generated and the population undergoes a reproduction process based on those seen in the natural world. Randomness is important for EAs because it is needed to create individuals, or 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 problems 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 of potential solutions:

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.

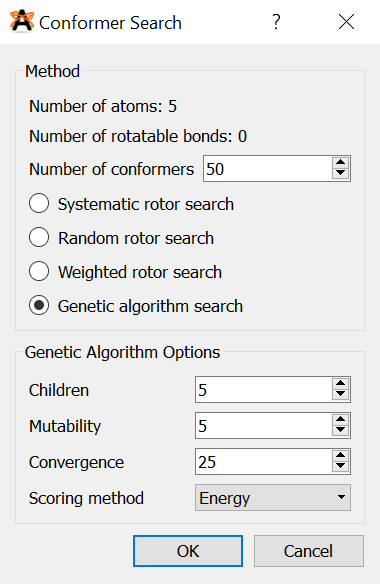
The unit of measurement for distances within molecules in this project is the angstrom, whose symbol is Å. It is equivalent to 10⁻¹⁰ metres. The report refers to distances between atoms rather than ‘bond lengths’ because the algorithms and calculation methods used in this project do not take into account any information about bonding and have no knowledge of the existence, types or locations of bonds present in the molecule. In real applications, a chemist might only be interested in one specific bond in a molecule because they may be considering the chemical reactions that could involve that particular bond. This report refers to the ground state total potential energy of configurations simply as their ‘energy’, and pseudo-randomness as ‘randomness’ for the purpose of concision.

## Aim

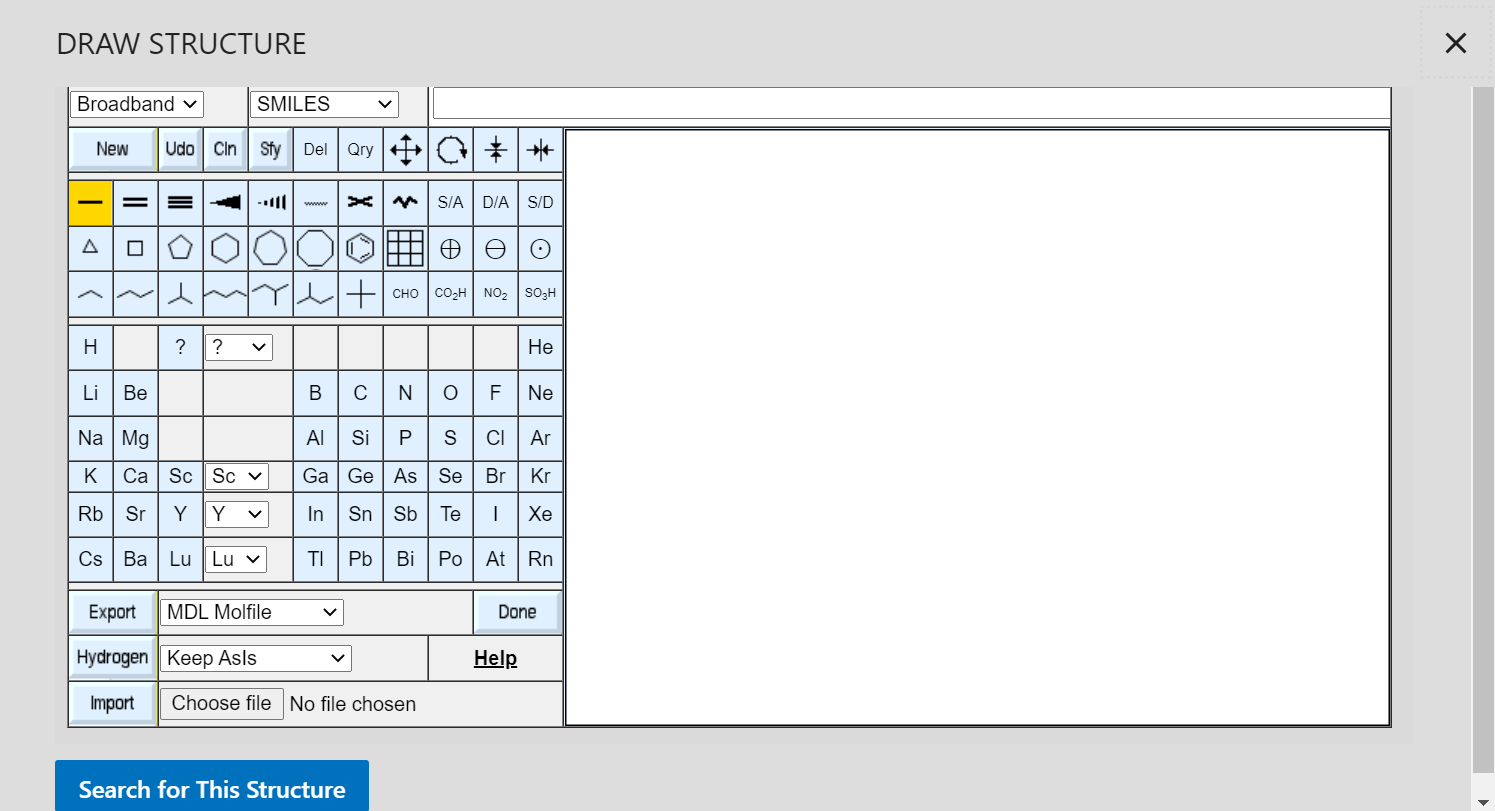
The aim of this project was to create an interactive EA to predict the geometric structure of a molecule or system of atoms, based only on an estimate of the system’s total potential energy. This was a one-objective problem. The objective was to find the arrangement of atoms, and therefore their bonds, where the net force on each atom was as close to zero as possible.

# Literature and Software Review

Software which performed similar tasks was identified and studied, and 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 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. It was not possible to use GaussView because it was not free to use and the University did not have full access to it, but 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.



*Figure x – Avogadro’s geometry optimisation user interface.*



*Figure x – Building a molecule from common groups in PubChem.*

Ijzerman et al. (2005) created an interactive EA where the user, an experienced chemist, contributed to the fitness function, allowing the chemist to influence the EA. This gave the chemist the ability to use their knowledge to improve the outcome. Interactive EAs have a broad range of uses and not all interactive EAs are aimed at users who are specialists in the subject area, and they also do not always influence the algorithm’s decision-making process. Interactive features were added to Geopt to allow the user to tune the EA’s variables according to their preference, but a self-tuning EA could also have been used.

Scientists have had greater success designing algorithms to optimise a single molecule or type of molecule than generalising a particular model to all chemicals. Johnston et al. (2002) created a GA to find different aluminium crystal structures. The flow chart for this algorithm is shown in figure x. Figure x 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 x 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. Comparing these similar algorithm designs reveals that they all used mutations for new generations, but Mohn’s (2016) algorithm also used crossover and maintained a population of only two solutions.Although it was not expected that Geopt would be able to generalise well to a range of molecules much more vast than those studied in the literature, the option was provided as a proof of concept.

Fan et al. (2011) studied the significance of initial guesses of chemical structures preceding optimisation and found that the best results were obtained when the parameters of the algorithm were repeatedly recalculated and adjusted to minimise error so that the initial guesses could be improved each time, resulting in fewer iterations of the EA. This may have been more applicable to an algorithm designed for one specific type of structure than an EA which generalised, though.

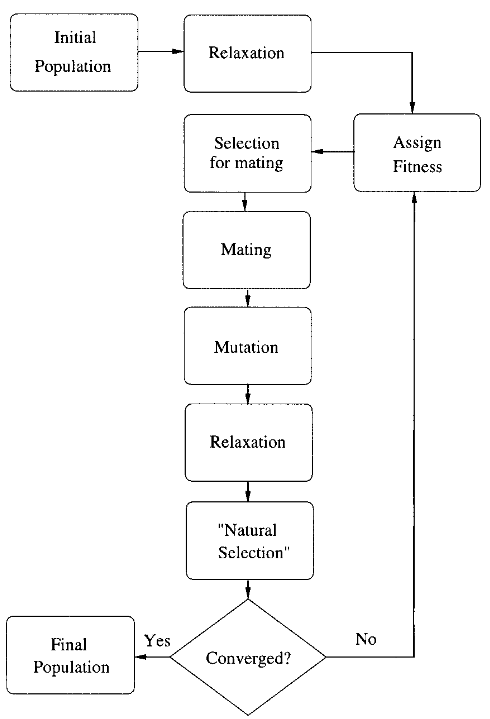
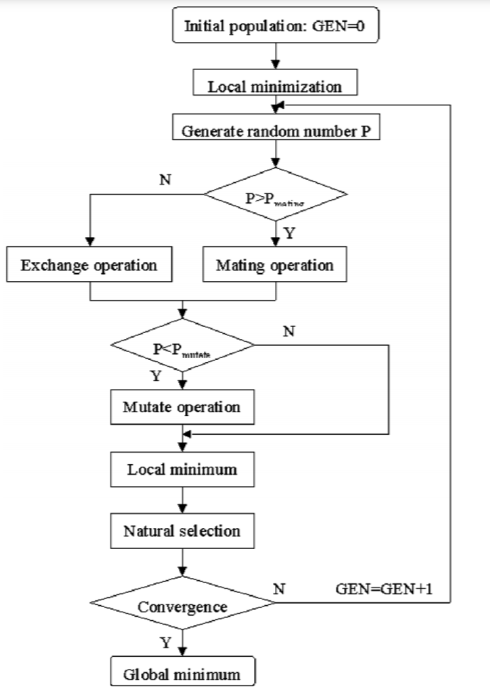
Hashimoto et al. (2015) reviewed the choice of programming languages for use in high-performance quantum chemistry programs and chose Python as their first choice. The researchers concluded that dynamic scripting languages may have been preferable to compiled languages such as C for this task, mainly 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(3n) 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(3n), 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.

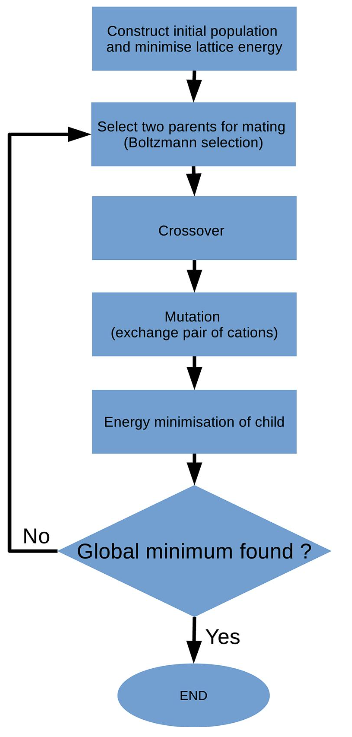
Later, Gueorguiev and Kuttel (2016) similarly used a GA for molecular conformational optimisation. Their aim was to discover three-dimensional structures of proteins that were formed from combining different amino acids. The greatest problem previously encountered by Addicoat and Brain (2010) was still 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.

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 an ordinary 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.

*Figure x – Flow chart of GA by Johnston et al. (2002).* *Figure x – Flow chart of GA by Chen et al. (2007).*



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

EAs are not the only way to conduct geometry optimisation. Ishimoto et al. (2011) investigated the use of a Monte Carlo simulation method for geometry optimisation of butane, a simple carbon chain, 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. The method was subjected to parameter optimisation, as well as a Hamiltonian algorithm to search the PES and an improvement was consequently achieved. Geopt was planned to create a PES so that the user could view the points of lowest energy on the plot.

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 personal computer, although it lacks accuracy as it makes many generalisations and assumptions. Moon and someone (2006) preferred binary searches to the technique employed by Ishimoto et al. (2011), and reported that, although it was sometimes useful, EMT often failed to describe structures correctly. This 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 x contains the roadmap, which outlined the plan 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:

1. Planning the project.
2. User story: a user wishes to create a molecule.
3. User story: a user wishes to predict a shape using an EA.
4. User story: a user wishes to predict a shape using a different algorithm.
5. User story: a user wishes to view a molecule and its analytical information, including a PES.
6. Usability testing and making changes to the program.
7. User story: a user wishes to interact with the algorithm.
8. Usability testing and making changes to the program.
9. Making finishing touches and bring the programming part of the project to an end.
10. 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 x 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 and try different techniques. This was effective and was added as an interactive feature which the user may adjust.

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 any 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 extenuating circumstances (EC) 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 x – Risk assessment.*

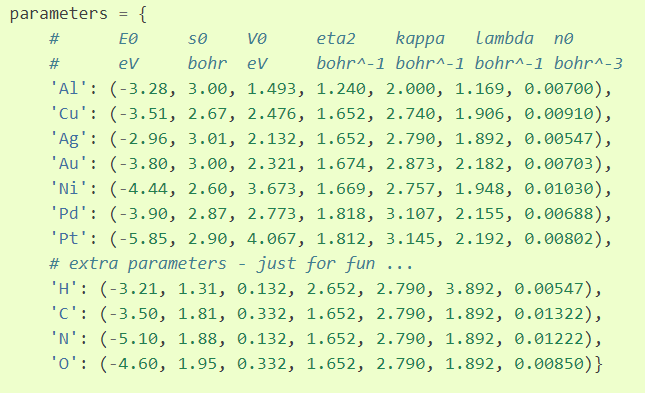
## Legal, Social, Ethical and Professional Matters

Ethical considerations for artificial intelligence (AI) projects are not as prominent in theoretical chemistry as in some other fields, but they do exist. According to Müller (2020), a widespread concern is that the use of AI and other 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 jobs in this way, and is not currently able to do so. Another concern is that AI may produce solutions to problems without its method being understood by the humans who use it, or even those who created 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 replicated 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 supplied by Dr. Shirley Atkinson 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 x to x. The questionnaires were anonymous, so no identifying information about participants is given in this report, its appendices or any other documentation.

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

Figure x 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.



*Figure x – 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 user interface in line with regular user feedback. The model-view-controller design pattern was used as it enabled clear separation between the user interface, the data and the functionality in the code.

Python was used as the primary 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 planned 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 mentioned previously. Furthermore, graphs, plots and visualisations are easy to show in Python using Matplotlib.

Tkinter was employed for the user interface as its ‘widgets’ are particularly useful for creating desktop-based forms and executable programs such as Geopt. 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 very 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 (DOM) in Python was a fast and simple process.

## Sprint Zero – Planning

Once the project idea was approved and the supervisor was determined, some research was done into the subject area, as described in the literature review of this report. Software which performed similar tasks was identified and studied, as discussed in the software review section of this report. After considering the existing software, Geopt was planned to be unique by suggesting multiple structures which could potentially even be isomers in some circumstances, and by not restricting the molecule creation to realistic formulae so that the user could study theoretical systems that would be unlikely to occur in reality, as the study of entities which do not appear in the natural world 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 also use random initial molecule structures rather than estimating initial starting positions based on chemical properties. This became the main topic of research for this project; 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 also expected that the number of atoms in the molecule would be restricted to a small value by the computational expense required.

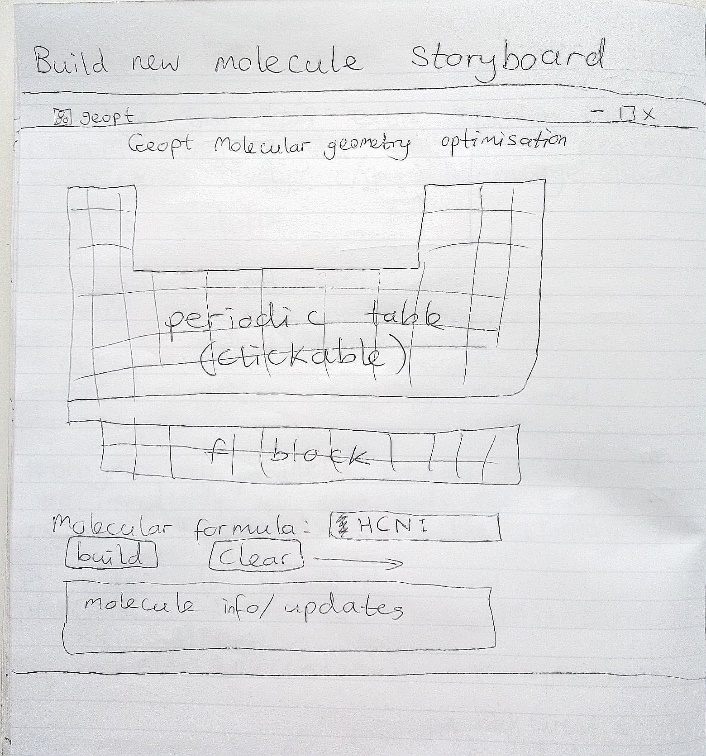
A questionnaire was created and sent to target users, science students, so that the project could be designed to meet their requirements. This questionnaire received two responses which can be seen in appendices x and x. The full, uncompleted version of the questionnaire can be found in appendix x.

After this, the backlog began and the roadmap was composed. The GitHub repository, Microsoft Planner and ReadMe file were created and the primary language was chosen as Python for the reasons explained previously. The EMT calculator was discovered, tested and chosen.

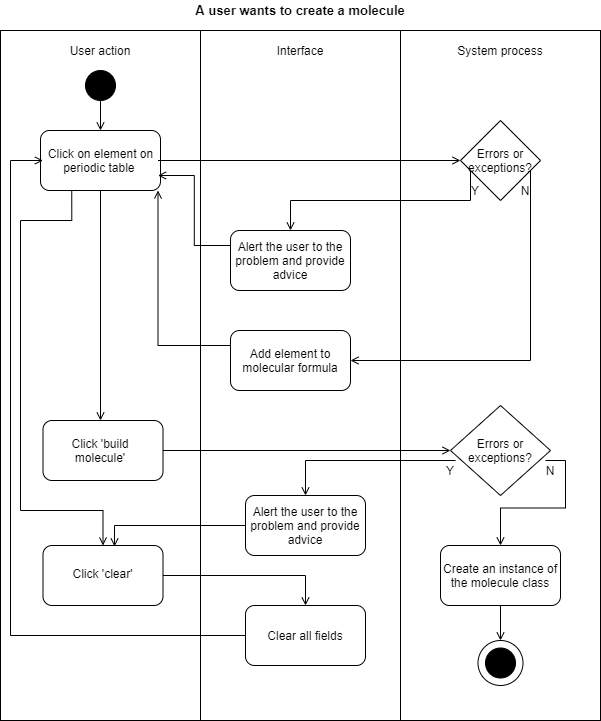
The project initiation document was compiled and submitted. This can be viewed in appendix x. The supervisor suggested that the project may have been at risk of focussing too much on chemistry and not enough on computer science. In response to this, it was planned that pre-existing Python modules would 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

After the main planning phase, sprint one was planned. It was based around the user story, ‘a user wishes to create a molecule’. Firstly, a page layout plan was drawn, and is shown in figure x. Secondly, an activity diagram was drawn and is shown in figure x. 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. Later in the project, this was changed, as discussed further in this report. The entry point of the application and user interface was created in Python. The user interface for this task was built using the Tkinter tool in Python. An XML file was constructed to hold information about each element, and this was used to populate a clickable periodic table. Figure x displays the XML for the first two elements.



*Figure x – Initial layout design for sprint one.*



*Figure x – An activity diagram for sprint one.*



*Figure x – Part of the XML file created for the elements.*

During the second week of the sprint, the idea for the user interface was changed to improve the layout and make the options clearer and simpler. The string typed into the text entry widget, which was for the user to type the molecular formula, 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’.

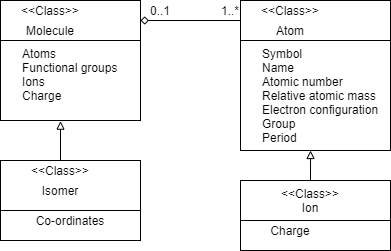
## Sprint Two – Predicting a Shape

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.

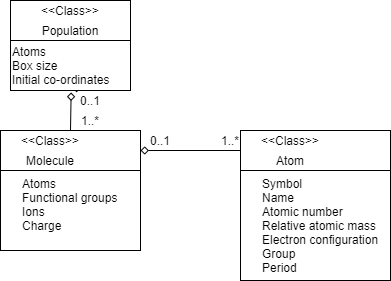
Calculations were created to place the atoms in a grid so that they were evenly distributed within the grid. Atoms were different sizes which meant that extra calculations needed to be done to work out the atoms’ sizes and adjust their positions and the grid size accordingly. The previously created XML files were used for this, which was possible because the atomic radii followed a pattern on the periodic table, so their places on the table were used to estimate relative size values.

The molecule’s atoms were placed into a unit cell with relative co-ordinates. These co-ordinates could then 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 x and x, 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 and unnecessary 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 x – UML diagram for the first plan for Geopt’s classes.*



*Figure x – UML diagram for the second plan for Geopt’s classes.*

An evolutionary algorithm was created to adjust the positions of the atoms to find the lowest overall energy. Functions for selection, random generation, 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. As with the cell scaling tests, this was performed in the test directory of the application to find out which methods and parameters worked best and worst. The results were somewhat disappointing as the EAs were unable to predict geometries well, although the energy values were improved considerably after adjustments, which was the main objective. 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 best (lowest) energy remained within a certain range for a specified number of iterations. The problem of getting trapped in local minima was harder to overcome than had been anticipated. The algorithm was named the many-molecule EA (MMEA) because it was an EA which used a population of many versions of a molecule.

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 actually made the permutations less favourable in some cases, so it was omitted in 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, or almost exhaustive, method so that it could be compared with the EA, as explained previously. The algorithm was named the per-atom exhaustive test (PAET) because it considered one atom at a time and worked on a single molecule.

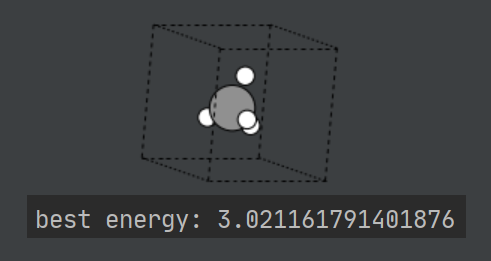
The PAET started with an empty cell and introduced one atom at a time, moving all other atoms systematically, testing a set number of random locations centred around one another upon each atom addition, continuously testing for the best energy. The main problem associated with this was that since every 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. The standard deviation of the Gaussian distribution within which each atom was moved around the other atoms was altered as the algorithm 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. This would need to be applied carefully because the algorithm already took a long time for larger molecules, with thousands of calculations, function calls and array changes per molecule.

Possible ways to further improve results of both the algorithms were identified. The algorithms could be furnished with a more accurate calculator, such as VASP, and Python multiprocessing to speed them up. Consequently, multiprocessing was applied to the PAET. Experiments were performed to choose the number of processes to run in parallel and the number of iterations to be run at each stage of each of these processes. Table x presents the results of testing the multiprocessing implementation. Six threads and six iterations were chosen, although this was later changed in the interactivity sprint.

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 x. Although table x 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.

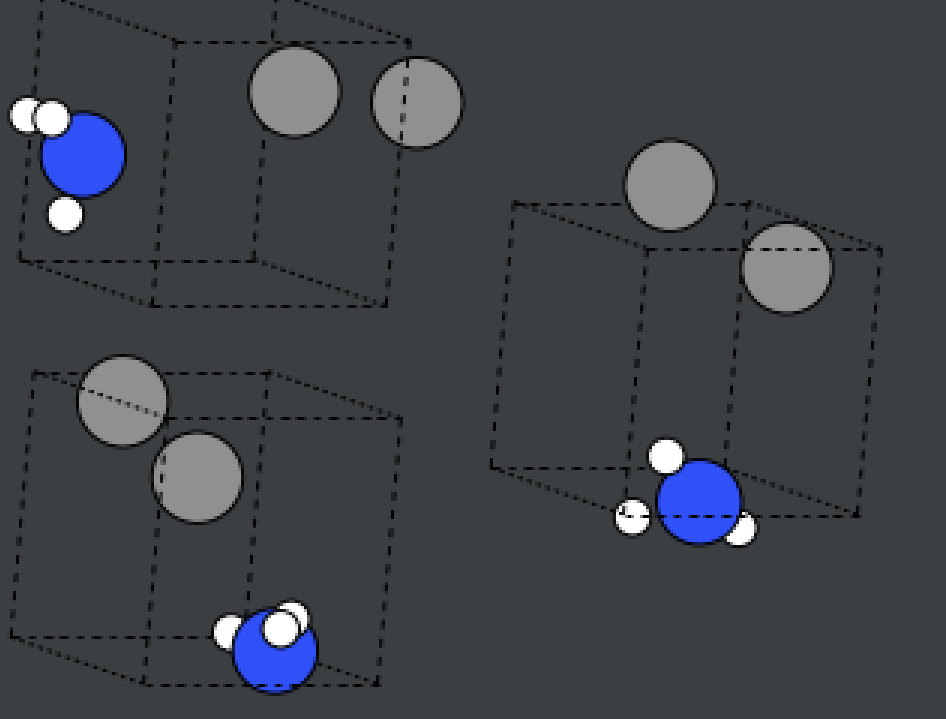
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Molecule tested | 4 threads, 4 iterations | 4 threads, 8 iterations | 8 threads, 4 iterations | 6 threads, 6 iterations |
|  | Energy / eV Time / s | Energy / eV Time / s | Energy / eV Time / s | Energy / eV Time / s |
| C12 | 4.5 21 | 3.9 75 | 4.5 47 | 3.2 40 |
| C6H6 | 14 19 | 23 42 | 26 33 | 20 40 |
| C3H6O | 16 10 | 27 22 | 7.0 15 | 14 17 |
| C2H6 | 4.4 12 | 8.5 62 | 3.8 36 | 5.4 39 |
| N4O4 | 1.3 21 | 1.7 58 | 1.3 21 | 1.2 25 |
| Al2O3 | 3.1 17 | 3.2 33 | 3.2 16 | 3.3 16 |

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

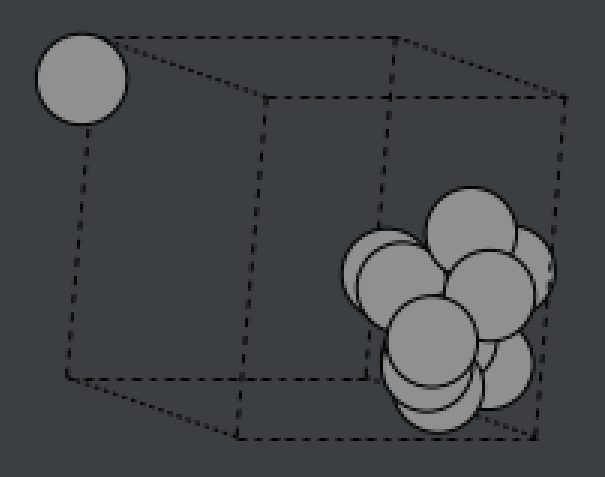


*Figure x – Successful prediction for methane.*

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 x. Figure x 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. Restructuring the data helped to alleviate problem.



*Figure x – Recurring separation problem with acetonitrile.*



*Figure x – A group of atoms clustered together.*

The algorithm was altered so that after each atom had fully circulated the group the structure reverted to the most energetically favourable state found during that loop. Table x shows the effects of this. Although it did not cause a large improvement, the updating feature was kept because the amount of processing required was small compared to amount used by the energy testing.

|  |  |  |
| --- | --- | --- |
|  | Energy achieved / eV | |
| Molecule | Update | No update |
| C12 | 3.2 | 3.9 |
| Benzene | 8.0 | 9.0 |
| Acetone | 5.5 | 8.6 |
| Ethane | 2.7 | 2.6 |
| N2O4 | 3.6 | 1.2 |

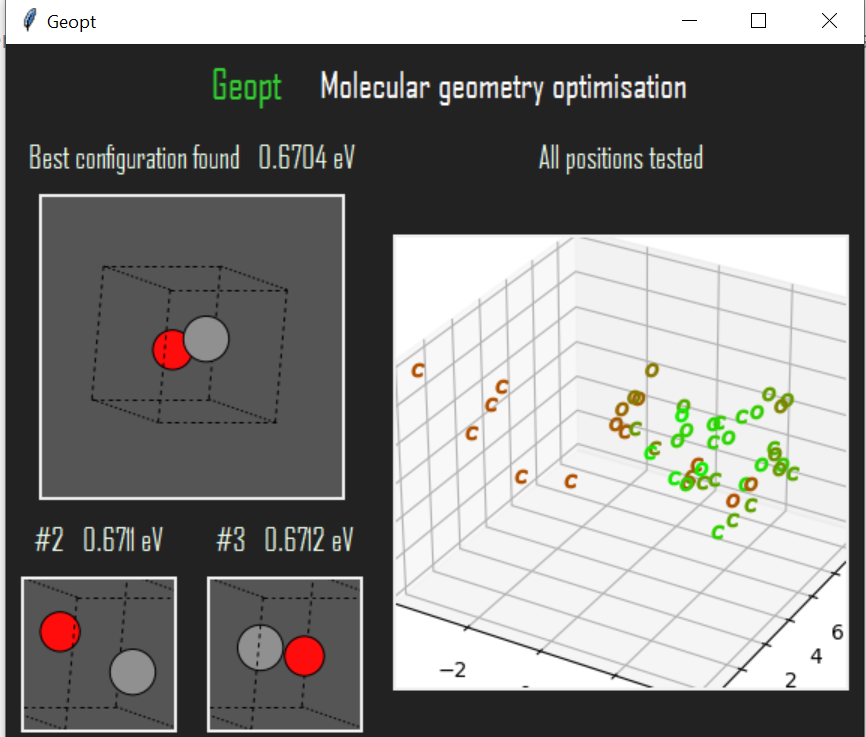
*Table x – Results of updating and not updating the structure after each atom addition loop.*

## Sprint Four – Viewing Results

When the roadmap was first planned, sprint four had been dedicated to testing, user feedback and amendments, and sprint five had been allocated to displaying visualisations and analytical information. As sprint four began, however, it became apparent that sprints four and five ought to be swapped because it would be difficult to conduct questionnaires with participants without the presence of more visual displays in the application. It was difficult to find participants who had an understanding of both chemistry and EAs so participants were not expected to comment on the algorithms but to comment on the usability of the program. Because of this, 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.

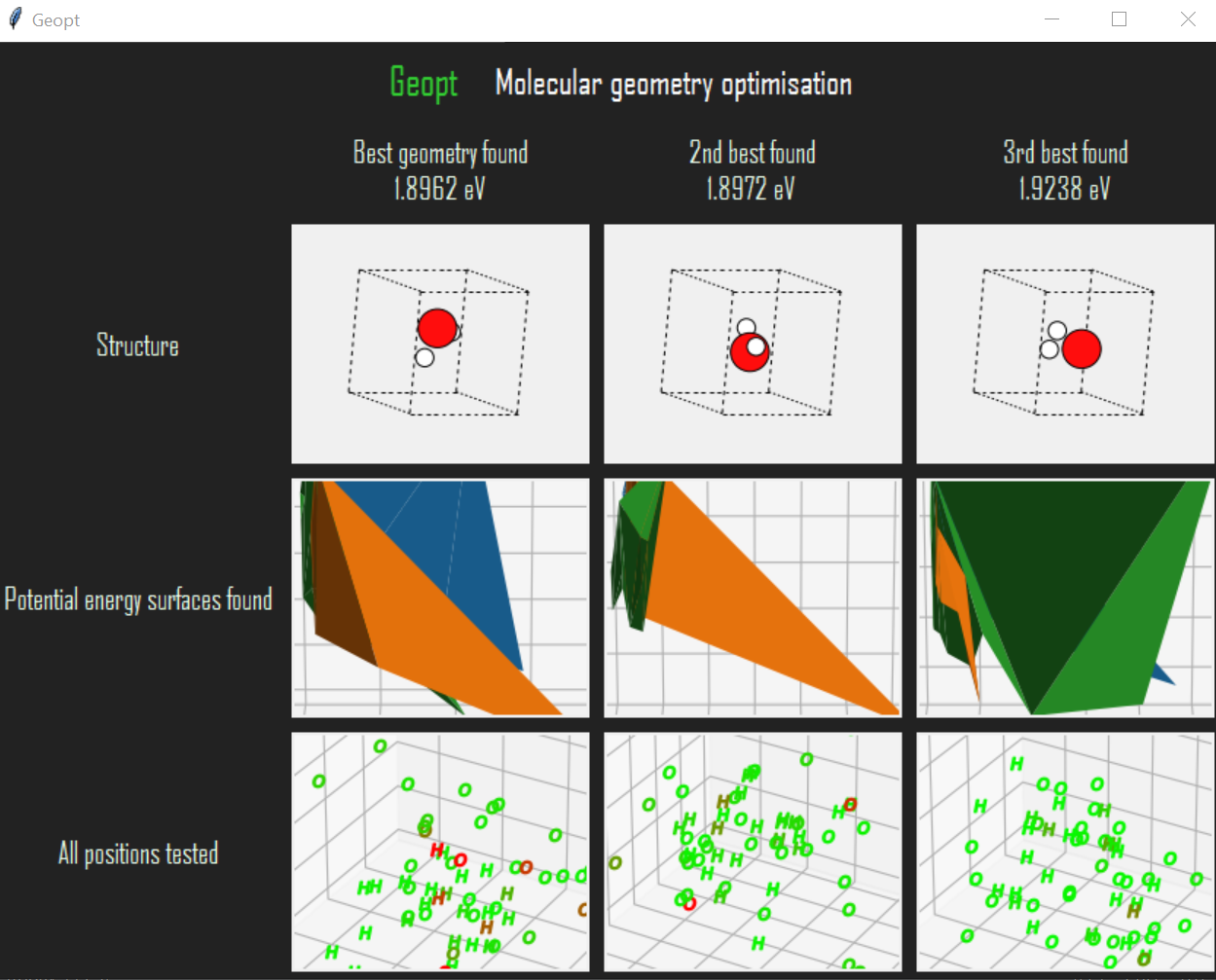
The optimised geometric structure was depicted on the screen and a window was designed to show the user an analytical breakdown of the optimised molecule. Plots were designed to display the different positions tested during the evolution. Datasets were created to hold information needed for the plots, 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. These datasets were stored as Python lists because their initial memory references would be quickly accessible in memory after the code was built and compiled and newly added values in the lists could also be held in cache during run-time, depending on the space available. Additionally, the data did not need to persist; they would need to be recreated during each execution. It was decided that 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 the range of zero to one so that the lowest energies yielded by the algorithm were close to zero and the highest were close to one. Python’s existing standardisation and scaling functions were considered, but it was simpler to implement a division of 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 x.

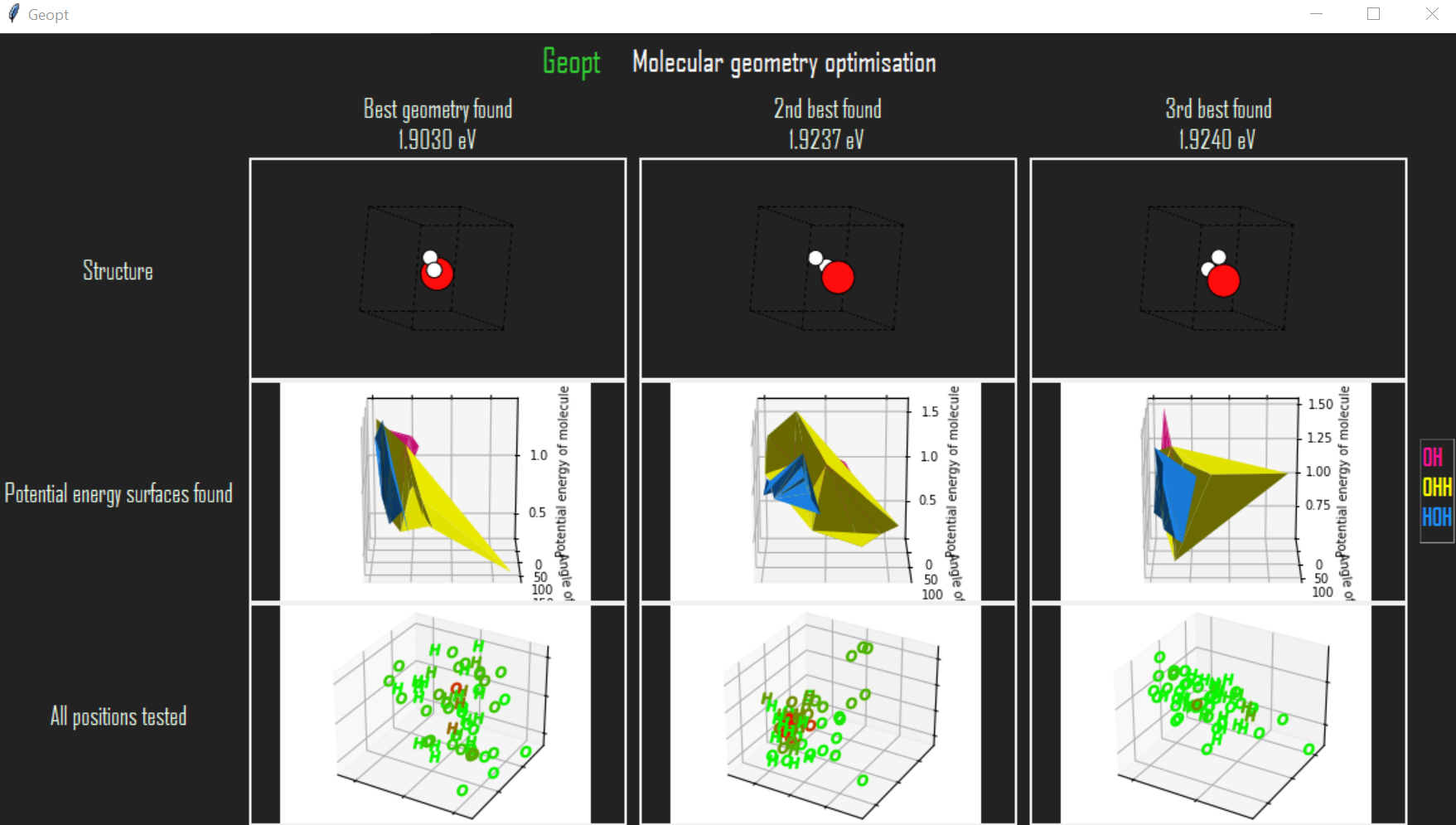


*Figure x – The first design.*

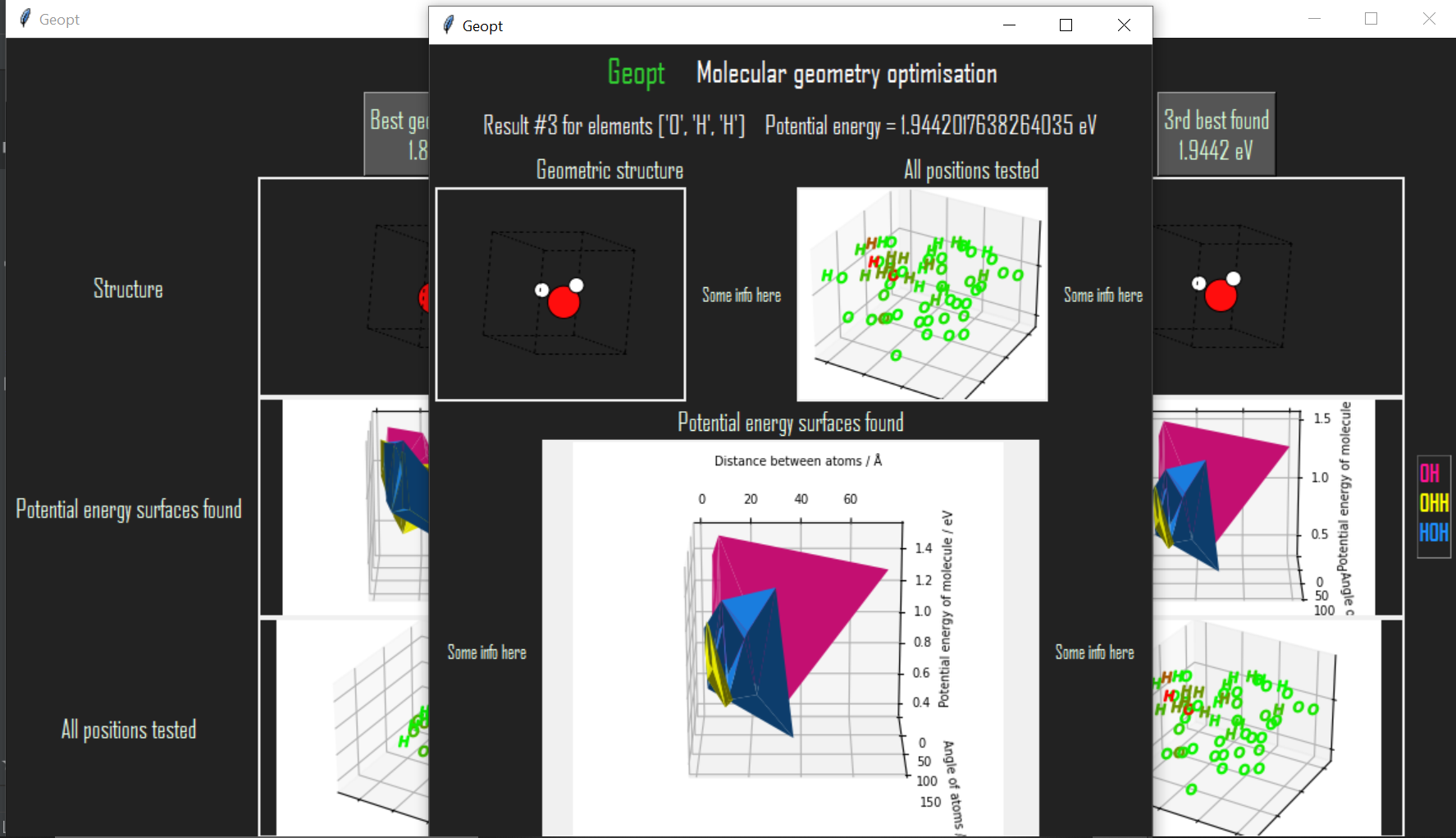
Next, the analysis view was expanded to show the three most optimal versions of the molecule so that the user could view the differences between them. This could enable a chemist to use their own knowledge to decide which structure was the most favourable and it could also potentially propose isomers of some molecules. The plots were initially displayed on the screen using a grid layout, as shown in figure x, 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. The axis rotation on the PES plot did not display properly. It was possible to specify the 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 larger plots and more information to the user.



*Figure x – The second design.*



*Figure x – The analysis window of the third design.*



*Figure x – The information window of the third design.*

The PES compared the total potential 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 way not have even 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 increased and the dimensionality of the graph 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.

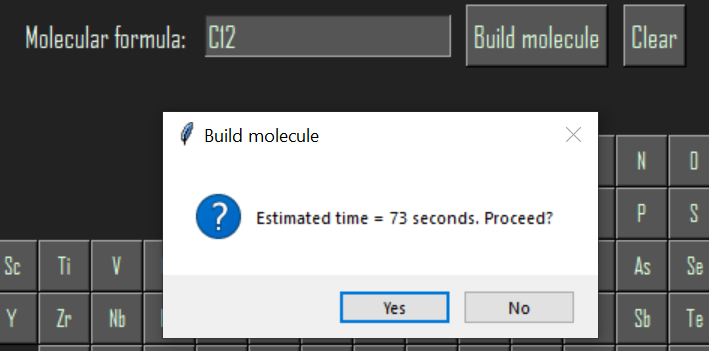
Several of the functions created in this sprint were amalgamated into one, larger function. This was done to speed up the algorithm because it meant that multiple things could be computed inside the same loop, reducing the total number of iterations and function calls during execution. A drawback of 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. Their completed questionnaires can be viewed in appendix x. Both the computer scientists reported difficulty understanding the application’s chemistry content, but the chemist had no difficulty with this. Although the computer scientists were unsure about the chemistry content, they found the application easy to use, overall, and were able to construct molecules in the way intended, without help.

The tests brought to light some errors in the code. One of the participants 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 a certain number (twenty at that time) of 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!’, for example. 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.

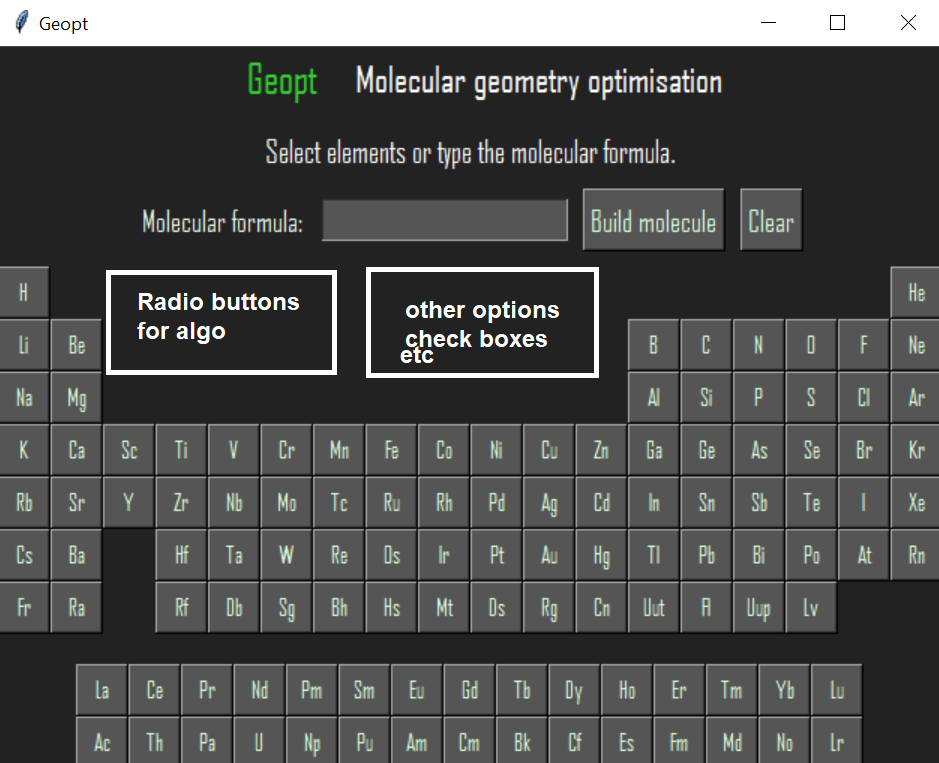
As well as finding bugs, the participants provided some advice to improve the program. It was suggested that the information buttons should have been made clearer, as a participant was unaware of the buttons’ functionality. It was recommended that the test positions plot should be made easier to read and understand. The layout of the user interface was changed according to these comments. It was also advised that the user should be reassured that the program was calculating and had not crashed or 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 x. 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.

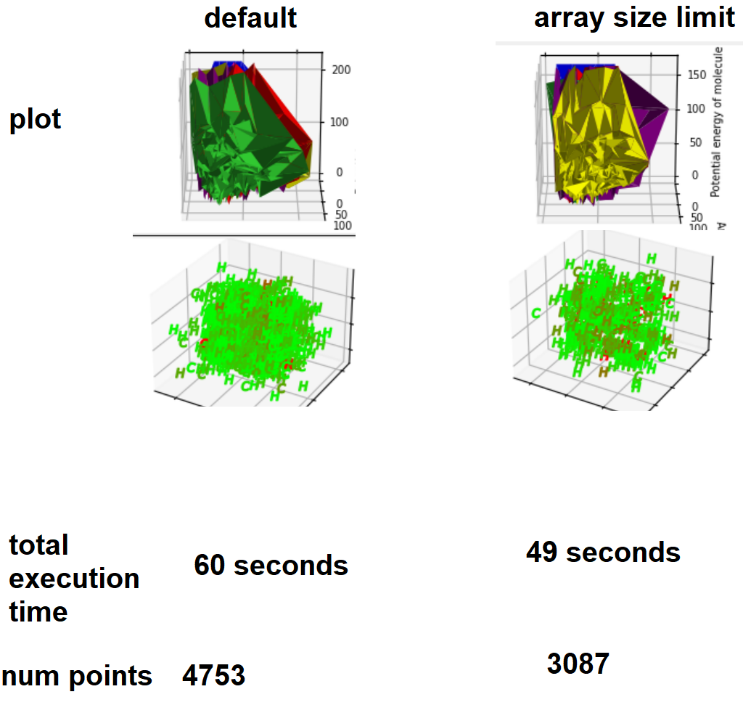


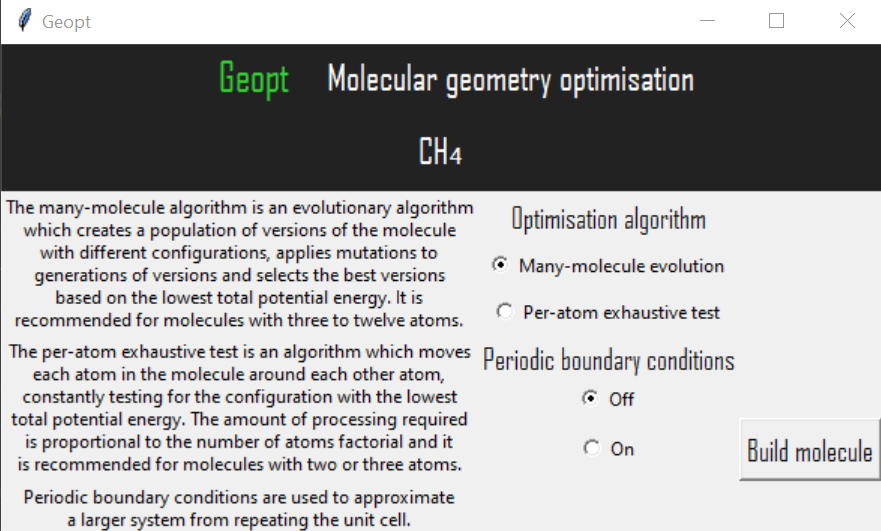
*Figure x – Time estimate of the algorithm.*

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



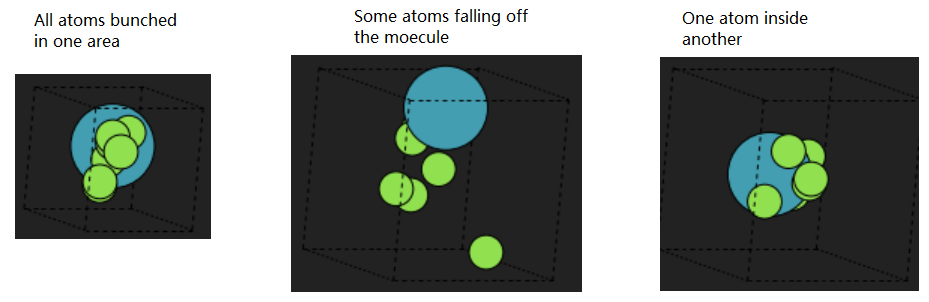


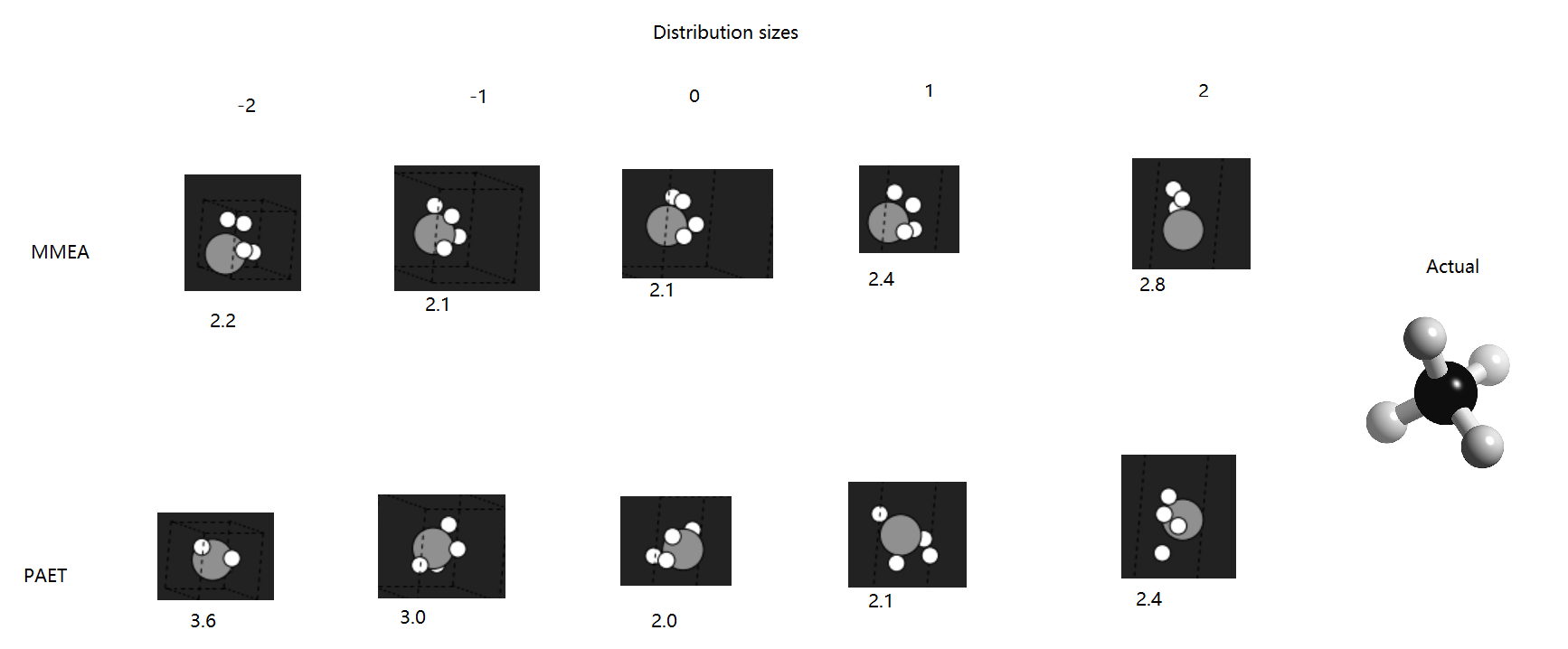


## Sprint Seven – Testing

Sprint seven was allocated to acting on user feedback and making final improvements to the project. A video was used to demonstrate the program to a group of students at the University of Plymouth in order to gather more feedback. 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 replaced with tooltips. Due to the random placement of atoms, the view of the molecule was sometimes observed to hide 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.

Some errors were also discovered and fixed. What errors were fixed? Look at Github





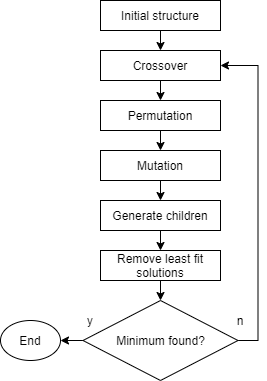
## Sprint Eight – Final Touches

## Sprint Nine – Showcase Materials and Report

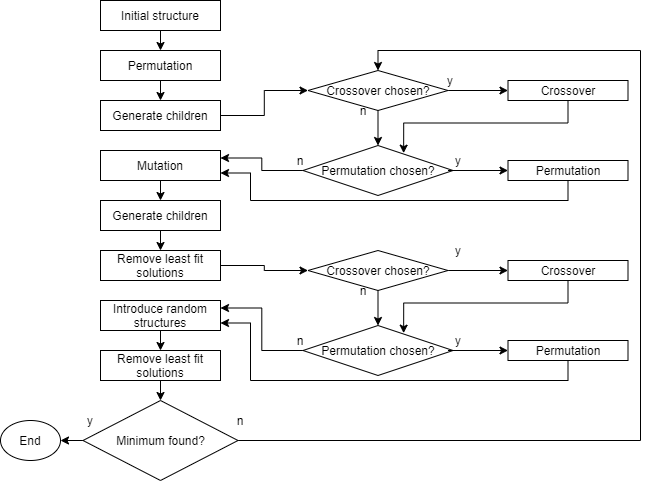
# Algorithms

## Many-Molecule Evolutionary Algorithm

A smaller random movement was used for parents’ descendents and a larger random movement was used for the new, unrelated versions which were introducted to diversify the population and avoid becoming trapped at local minima.



The original plan



how it ended up!

initial guess could have been random co-ordinates

the literature review some people had shown that when evolutionary algorithms had been used in geometry optimisation, the optimisation could continue for an indefinite period of time, similarly to the continuous nature of biological evolution. It was assumed that if the best energy value found did not change significantly for several consecutive iterations, the process was likely to have reached the point at which it should be ended and its output should be retrieved.

The first execution of the MMEA 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 higher, 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 very few ways in which atoms could reasonably be arranged in a system, particularly in such small molecules.

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.

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 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 yielded a more favourable arrangement. The frequency of permutation was increased.

The population size was increased, although this added a significant load to the computer. 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.

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.

An additional random structure was introduced at each generation iteration in an attempt to overcome local minima.

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 noticably increase the time taken.

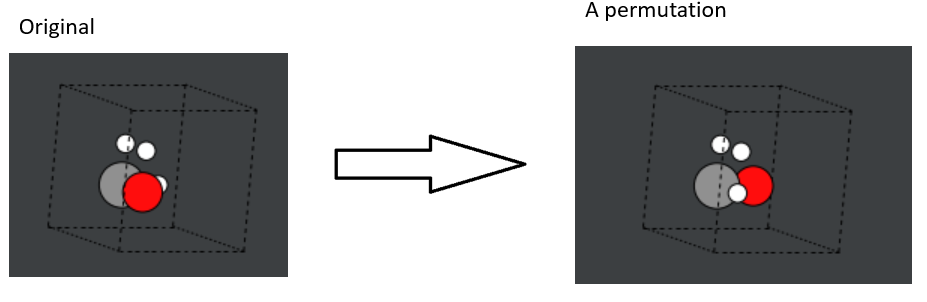
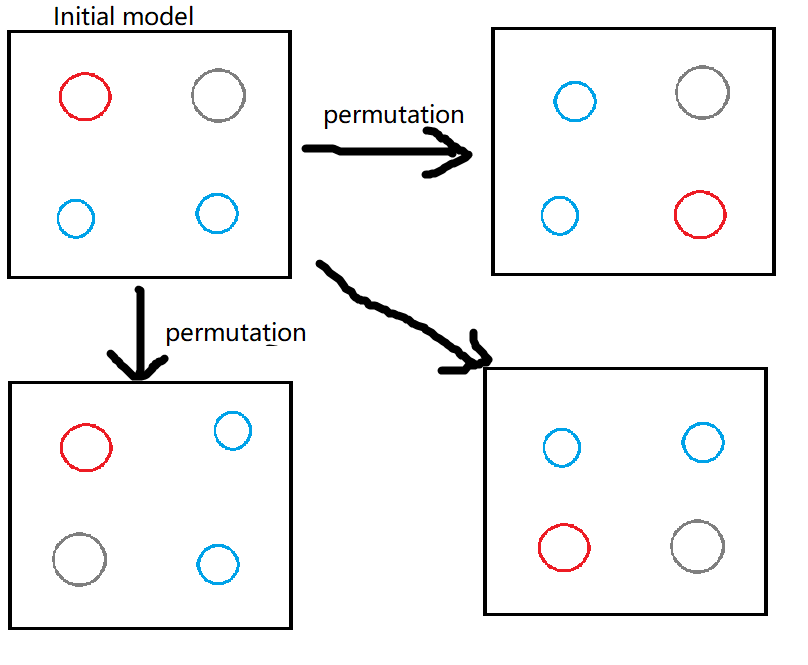
A corrective function was created to push hydrogen atoms towards carbon atoms because of their tendency to move further away from the molecule with each iteration. It was later determined that this method was not desireable because it did not address the problem, which was partly that the EMT calculator 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.

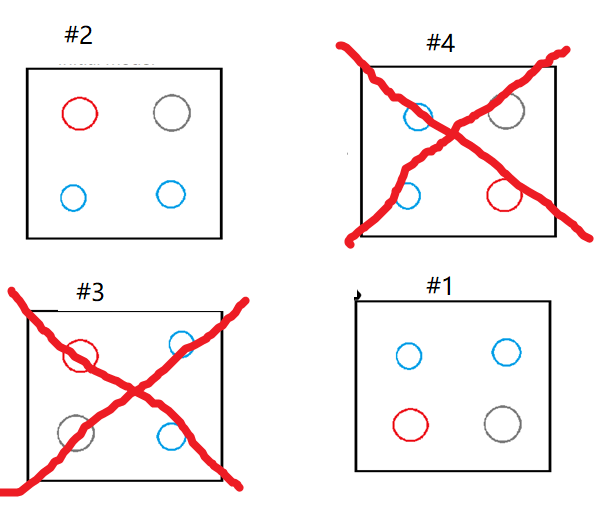
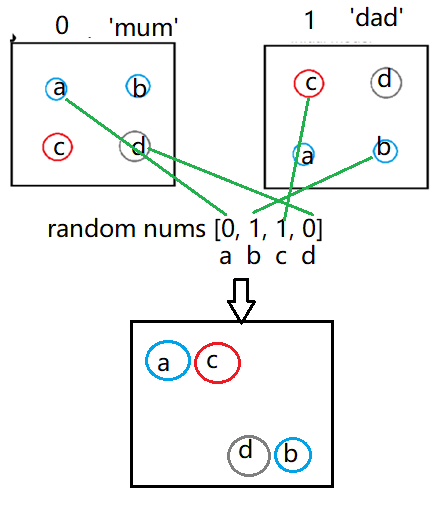
To encourage atoms to move, the space between atoms in the initial model was decreased to ensure that some repulsion existed in the system, resulting in more force being exerted on atoms.

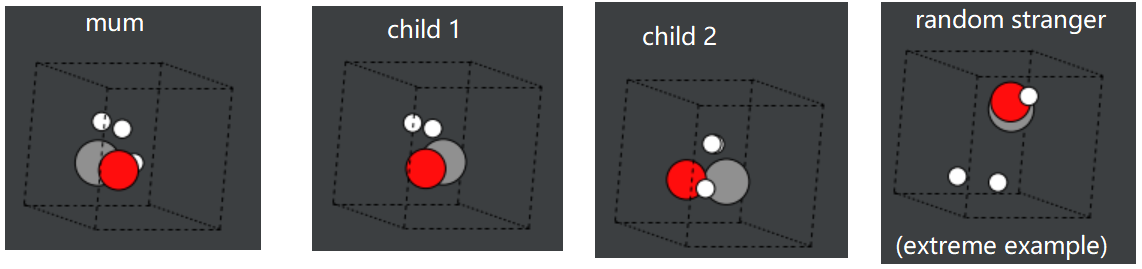
As previously mentioned, this was a one-objective problem in which the fitness function was a measure of the total potential energy of the molecule at each step. Later into the project, at around sprint x, it became apparent that this was usually not sufficient, and the possibilty of adding a second 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. Talk about them and show diagrams. Although this could have improved the performance of the algorithms, it was not implemented because, although some of the project’s time had been allocated for general improvements, this 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 evolutionary algorithm 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, such as the distances between atoms being within a certain range, based on various properties such as bond order.

Diagram

Description automatically generated





## Per-Atom Exhaustive Test

Draw a flow chart

The list of the system’s atoms was rearranged to place all hydrogen atoms at the . This was a fast solution which only needed to be performed once per execution. The corrective function created for hydrogen in the MMEA was used in addition to this.

During the previous EA tests, the atom movements were almost always unfavourable apart from the permutations. Another problem which had previously arisen 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.

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 involved. The atom movements were still mostly unfavourable and the evolution still often became stuck in local minima. The EMT calculator was partly to blame for the undesirable outputs due to its inability to calculate energies accurately for most combinations of elements. Another problem was that the large amount of processing – often thousands of calculations, array changes and function calls per iteration – appeared to put a strain on the computer’s hardware.

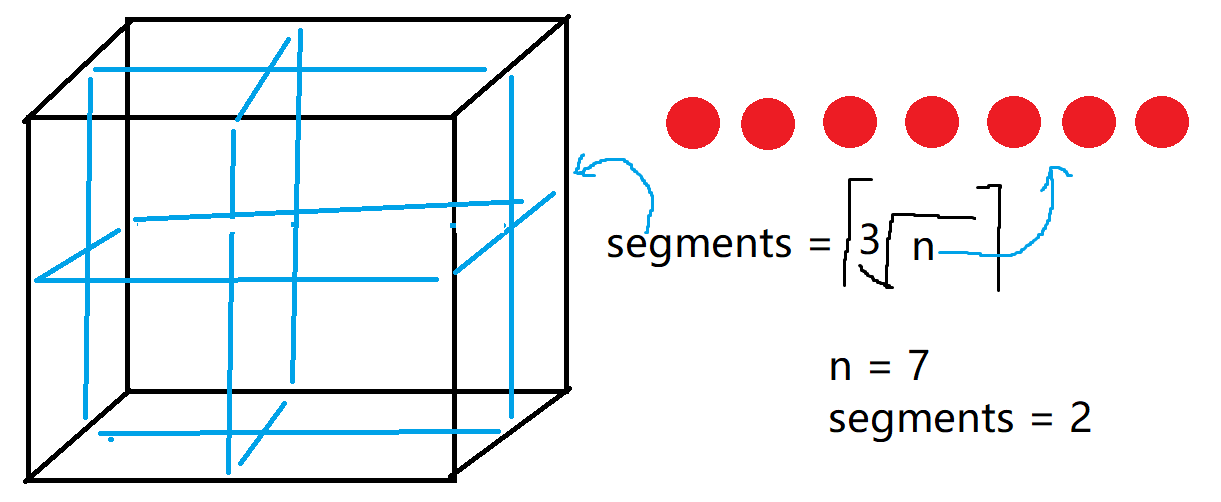
Multiprocessing, which is a parallelism tool provided by Python and based on multithreading, was used to reduce the time spent in loops, which made up the vast majority of the execution time. How did it work to reduce the time?

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 the key concept behind the optimisation.

Code optimisations such as scalar replacement and strength reduction were applied. Functionality for different parts of the program was moved into the same loop kernel 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 but it was deemed acceptable, especially since there was only one developer. 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.

Evolutionary features 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.

## Other Algorithms



# User Feedback

User feedback was sought in three stages, at sprints zero, five and seven. Originally, it had been planned that the second stage of user feedback would be conducted at sprint four, but it was decided that this would be more fruitful if it was moved to sprint five to allow work to be done on the user interface first, as usability testing could be difficult and confusing if it was conducted without a substantial user interface.

The tests were conducted via remote access to the developer’s PC or by watching videos of the application being used.

# Discussion & Evaluation

Add a ‘reflection’ section.

Talk about how there weren’t enough users to test it. Cite someone saying how many you should have. Justify why we only had a few.

Talk about VASP & DFT

Talk about how the fitness function wasn’t good enough and that limited the whole project.

Talk about getting lost in reams of code and the importance of good practice.

Talk about adding another objective

Talk about how, in reflection, it might have been better to choose a simpler topic focusing one one specific algorithm or concept in EAs.

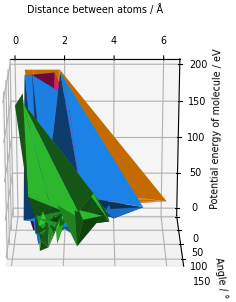
Compare my complexity to the literature.

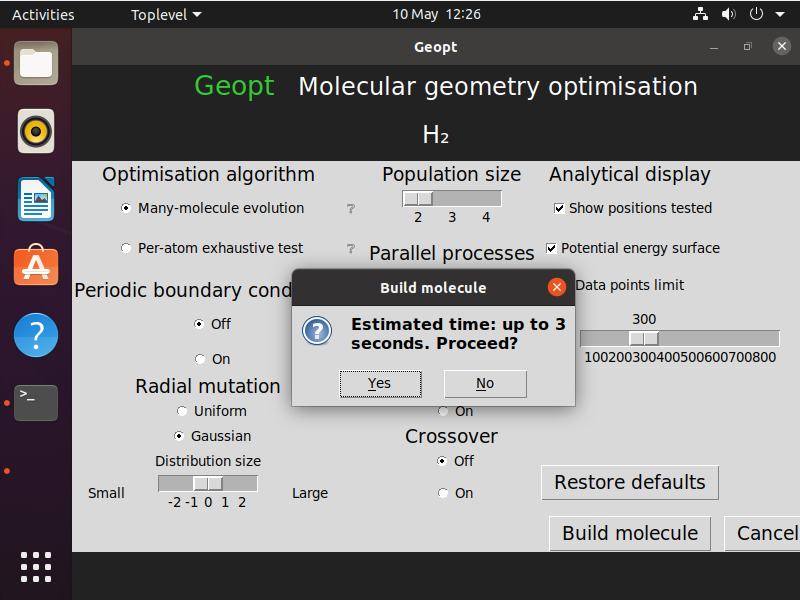
The sprints did not always finish on time according to the roadmap but a benefit of Agile is that it is flexible and allows some leeway. Sprint four had not gone to plan; it had been swapped with sprint five and extended to enable the GUI to be recreated using a new design in which graphs fitted better in the windows. Although no serious problems were encountered as a result of changing the sprint plans on the fly, 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.

The precision of some variables used in calculations was reduced in order to speed up the algorithms.

Earliest measurements ^

Latest measurements ^





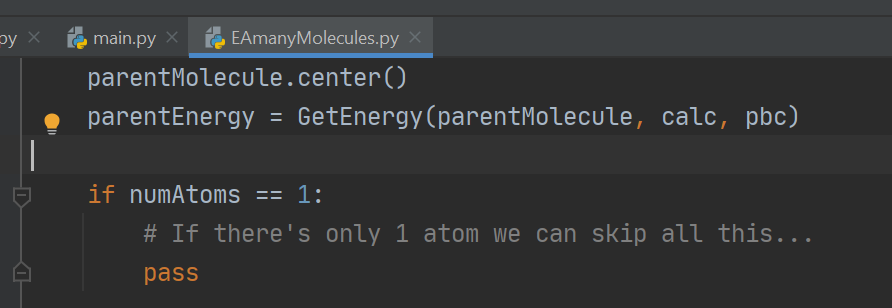
# Conclusions

compare outcome to the aim

compare the 2 algos

how much does the energy get reduced?

# Suggestions



It was possible that Geopt could suggest different isomers of a molecule when the molecule was optimised multiple times in separate processes, each yielding a particular configuration. If the project was to be further developed, this feature could be exploited to identify isomers. 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 would be to consider the differences between the energy values of all the outputted 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.

The standard deviation of the Gaussian distribution within which each atom was moved around the other atoms was altered as the EA 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. This would need to be applied carefully because the algorithm already took a long time for larger molecules, with thousands of calculations, function calls and array changes per molecule.

Supervised learning could be added to the program to inform the algorithms. 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, for example. Find literature on this and check lectures too.

Could introduce reinforcement learning by using and updating a Q table of best results or something? Look into this.

Look at memoisation.

self-tuning, hyper-heuristic .

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.

# 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., Dułak, 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., Dułak, 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 evoluator: 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.

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 x – 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 be 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 x – Target User Questionnaire

UNIVERSITY OF PLYMOUTH

**FACULTY OF SCIENCE AND ENGINEERING**

**Consent Form**

CONSENT TO PARICIPATE 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](mailto: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?

1. 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?

1. Was this questionnaire easy to understand and fill out? Is there anything that you think should be changed about it?
2. 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](mailto:sophie.turner@plymouth.ac.uk)

## Appendix x – Participant A’s Responses to Target User Questionnaire

1. Which science are you mostly involved in?

*Chemistry*

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

*BSc (Hons) Chemistry*

1. 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

1. 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

1. Why did you choose this software over others?

Chemdraw – free licence

GaussView – free licence

1. 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.

1. 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.

1. 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

1. 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*

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

*Good questionnaire.*

1. 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 x – Participant B’s Responses to Target User Questionnaire

1. Which science are you mostly involved in?

Biology

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

Postgraduate

1. 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

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

Visualising chemical structures.

1. Why did you choose this software over others?

I was instructed to as a learning exercise.

1. What do you consider to be the best things about this software?
2. What do you consider to be the worst things about this software?
3. 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.

1. 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.

1. 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.

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

## Appendix x – First Usability Test Questionnaire

UNIVERSITY OF PLYMOUTH

**FACULTY OF SCIENCE AND ENGINEERING**

**Consent Form**

CONSENT TO PARICIPATE 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](mailto: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 x – 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 x – 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 x – 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 x – Second Usability Test Questionnaire

## Appendix x – Participants’ Responses to Second Usability Test Questionnaire

## Appendix x – 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 whowant to estimate and view the structures of molecules without spending hours doing calculations or using a supercomputer. Geopt is aprogram which uses machine learning to predict the shapes of theoretical molecules.

Risk Plan

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Ref | 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. |

Keywords

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

## Appendix x – 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  H2O | Chemistry of Water | 1.900 | 2.531 | Excellent | Good | 2 | 2 |
| Carbon dioxide | 0.9230 | 0.9720 | Poor | Poor | < 1 | 1 |
| Methane  Ch4 Molecule Images, Stock Photos & Vectors | Shutterstock | 2.993 | 3.890 | Good | Poor | 1 | 3 |
| Dinitrogen tetroxide  Nitrogen tetroxide (dinitrogen tetroxide, N2O4) rocket propellant molecule.  3D rendering. Atoms are represented as spheres with conventional colour  coding: nitrogen (blue), oxygen (red Stock Photo - Alamy | 3.418 | 6.617 | Good | Poor | 10 | 4 |
| Acetonitrile  Acetonitrile | 1.827 | 4.204 | Poor | Poor | 1 | 4 |
| Acetone | 3.455 | 27.69 | Good | Good | 12 | 20 |
| Lattice of 12 carbons  cubic crystal system | NIH 3D Print Exchange | 3.020 | 11.32 | Good | Poor | 2 | 48 |
| Benzene  Illustrated Glossary of Organic Chemistry - Benzene | 2.416 | 69.32 | Excellent | Good | 39 | 50 |
| Aluminium oxide | 4.467 | 4.139 | Good | Good | 8 | 3 |

## Appendix x – Roadmap

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

*Table x – Sprints zero and one.*

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

*Table x – Sprints two and three.*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | Week 9 | 04-Dec | Week 10 | 11-Dec |
| **Milestones** | |  | **Sprint** | **four** |  |
|  |  |  |  |  |  |
|  |  | **User story: I wish to view molecule, info & PES** | | | |
| **Plan** |  |  |  |  |  |
|  |  |  |  |  |  |
| **Test** |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  | **Potential energy surface plot** | | |  |
| **Application** | | **Display best molecules** | | |  |
|  |  | **Annotations** | |  |  |
|  |  | **Display information (angles, distances)** | | | |
|  |  | **Use of colours** | |  |  |

*Table x – Sprint four.*

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

*Table x – Sprint five and the Christmas holiday.*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | Week 13 | 22-Jan | Week 14 | 29-Jan |
| **Milestones** | |  | **Sprint** | **six** |  |
|  |  |  |  |  |  |
|  |  | **User story: I wish to interact with the algorithm** | | | |
| **Plan** |  |  |  |  |  |
|  |  |  |  |  |  |
| **Test** |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  | **Implement plot limits in perAtom** | | |  |
| **Application** | | **Include buttons etc for interactions** | | | |
|  |  | **Make user interface for choosing settings** | | | |
|  |  | **Let user choose the algorithm** | | |  |
|  |  | **Let user set cell boundaries** | | |  |

*Table x – Sprint six.*

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | Week 15 | 05-Feb | Week 16 | 12-Feb | Week 17 | 19-Feb | Week 18 | 26-Feb |
| **Milestones** | |  | **Sprint** | **seven** |  |  | **Sprint** | **eight** |  |
|  |  |  |  |  |  |  |  | Finish code | |
|  |  |  | **Testing** |  |  |  | Final fixes & clean up | | |
| **Plan** |  |  | **Feedback & changes** | |  |  | Start writing report | |  |
|  |  |  |  |  |  |  |  |  |  |
| **Test** |  |  | **Usability tests** |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  | **Make changes from user feedback** | | | | **Make changes from user feedback** | | |
| **Application** | |  | **Find and fix bugs** | |  |  | **Find and fix bugs** | |  |
|  |  |  | **Tidy up** |  |  |  | **Tidy up** |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |

*Table x – Sprints seven and eight.*

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

*Table x – The final sprint of the code, leading to the showcase submission.*