MSE 2017 Project Final Report

Optimization of Wave Energy Converters

Yuanzhong Xia
The University of Adelaide
Adelaide, SA 5005
a1700831@student.adelaide.edu.au

ABSTRACT

This report firstly introduces the progress our team have achieved in both Semester 1 and Semester 2. Then, my contributions since the end of Semester 1 are introduced in detail. That includes research contributions on exhaustive grid search, 3-D landscape plotting analyse and maximum tether elongation study, and software contributions on web application back-end development using Java Spring Boot framework, part of front-end development on security and integration with web back-end, and High Performance Computer (HPC) back-end development of writing machine learning tools and integrating with web back-end. Besides, our motivations, others' related works on doing wave energy optimization research and building the web application are mentioned as well. In the end, there are conclusions and my reflections about the whole project.

KEYWORDS

Machine Learning, Wave Energy, Optimisation, Web Application, High Performance Computer, Java Spring, Software Engineering

1 INTRODUCTION AND MOTIVATION

This report is supposed to introduce the progress happened in Semester 2 only. However, our works in Semester 2 are based on Semester 1's efforts. Therefore, for a better context consistency, outcomes from Semester 1 are shortly introduced in this report when necessary.

1.1 Wave Energy

Wind energy is a clean, renewable and widely distributed energy form. By studying into wind energy, researchers found that wind energy is often transformed into another form - sea wave energy [20]. Wave energy mainly comes from the temperature differences between oceans and lands, and the Earth rotations.

In our world, ocean covers 70.9% of Earth's surface [17], and wind energy upon seas are not likely to be captured [21]. Due to the increasing energy resource demands from countries of all over the world, making use of sustainable energy becomes more and more critical.

Besides, Australia has the most wave energy in the world. Research conducted by the Commonwealth Scientific and Industrial Research Organization (CSIRO) [3] shows that southern coastline of Australia has excellent wave resources, because the strong Southern Ocean winds which travel northwards to Australia's southern coast generate large waves consistently. They also mention that wave energy could contribute up to 11% of Australia's energy and has the potential to produce enough electricity to power a city in the size of Melbourne by 2050.

1.2 Challenges

Although sea wave energy is a massive potential energy form in the world, using it efficiently is always an unsolved challenge. Therefore, in the current stage, wave energy still lacks researching and exploration. According to previous researchers, wave energy cannot be used directly, and it has to be captured and converted to electrical energy by an extra device. This device is called Wave Energy Converter (WEC). The first challenge here is to improve WEC's efficiency of converting wave energy to electrical energy.

In this project, we work on one specific type of WEC - "buoy", and more specifically each buoy is submerged and connected with three tethers to the seafloor. Because the previous research outcomes, our works are based on this type of buoy. As we all know, the capacity of energy captured by a single buoy is insufficient. If we put more than one buoys in a close area arbitrarily, they will affect each other, which results in producing unexpected less energy. Also, more buoys cost more money; thus our goal is also reducing the cost. We define a way of placing a specified number of buoys in a size-fixed area (also named a wave energy farm) as a layout. Therefore, the second challenge, which is also the primary challenge, is how we can optimize the multi-buoy layout to generate as much energy as possible, and cost as less as possible.

1.3 Solutions

To overcome the first challenge, this project cooperates with researchers from mechanical engineering. They work on the design of buoys and solving problems in physics.

Our work is trying to solve the second challenge, which is based on existing researches by Wu *et al.* [23], Arbones *et al.* [1], and Ding *et al.* [4]. We aim to find the best layout of a certain number of buoys using machine learning technologies to maximize the wave energy production and minimize the cost on deployment. Also, we provide researchers (our users) with a friendly web application containing lots of helpful tools connected with HPC to accelerate their research.

In Semester 1, we spent most time in discovering the ways to train fast and accurate surrogate machine learning models, to replace the original time-consuming physical Computational Fluid Dynamics (CFD) models, which are described in Scruggs *et al.*'s work [13] and Wu's work [22]. In this way, researchers can do more experiments within shorter times without losing much accuracy.

In Semester 2, our work is building the web app using Java Spring Boot framework, so this report covers the details on this web app, including its implementation and the developing process. Our decisions on tool selection and system design, and my contributions are included as well. This web app allows users to upload and randomly generate datasets, and train studied machine learning

models based on Scikit-learn, WEKA and MATLAB in a highly customized way. Then users can use trained models to predict other datasets. Also, some visualization tools are included for users to observe and tune configurations. Because the model training process takes much time and lots of hardware resources, this process is handled by HPC provided by our university. Users can check the progress regularly on the web app. When the training process is finished, email notifications are sent to users. Therefore, this web app involves two servers: the web server and HPC portal server. More details will be introduced in Section 2.3.

Apart from the main project, we also did a side project which is shown on university's OpenDay event. That project has an Android app and a web ranking app, allowing users to make their buoy layouts on an Android tablet, moreover, there will be the layouts' power output predictions submitted to the web ranking app. The prediction uses WEKA Multi-layer Perceptron (MLP) model which also runs on the web server. Then, a big screen is used for showing the rank list with corresponding layouts, and the list is updated frequently. Details can be found in Section 2.2.

2 CONTRIBUTIONS

In Semester 1, I worked on machine learning technologies (Random Forest, AdaBoost, Bagging, Extra Trees and Gradient Boosting) based on a package named Scikit-learn powered by Google™, and web back-end development using PHP + MySQL + Nginx technology stack.

In Semester 2, we started the web app project from scratch using Linux + Java Spring Boot + MySQL + Nginx technology stack, Although there were not much research components in Semester 2, I helped my supervisor Markus with some research-related data processing works. As to the software development, I still mainly worked on back-end development and server maintenance, plus, I worked on some front-end security enhancements and the integration between front-end and back-end.

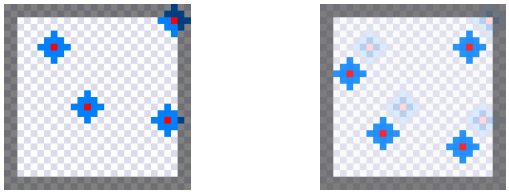
As well, due to the requirements of this report, only contributions made by me in Semester 2 are introduced in this section.

2.1 Research

As is mentioned, in Semester 1, lots of research works on using machine learning technologies have been done to build surrogate models to replace the time-consuming CFD model; while in Semester 2, we do not really have research components, thus there are not much research works for me to do. Therefore, research related works that I have done in Semester 2 are all about data processing.

2.1.1 Computer Vision Dataset. At the beginning of Semester 2, our supervisor Markus came up with the idea of treating each layout as an image. Therefore, we should firstly try Convolutional Neural Network (CNN) which was very popular in computer vision and image processing to solve the challenge. Previously, we tried other machine learning technologies with the input of plain buoy coordinates or extracted features based on plain buoy coordinates, however, we did not get satisfying results with an acceptable amount of training data.

As a result, I was responsible for generating the training data in a requested format. This training data is a set of images with the size of 28×28px. In the 28×28px area, only the middle 24×24px area can be used for buoy coordinates.



(a) original **(b) moved**

Figure 1: (a) 28×28px image, of which the middle 24×24px area is be used for placing buoys. Four buoys are represented in red grids. Around each red grid, twelve more grids represented in blue are used for enlarging the area of each buoy in the image. In final 2-bit black/white image, red grids and blue grids become black, and the rest of them become white. (b) The four buoys maintain their relative positions and are moved to the centre of the image.

In our previous research, the area available for four buoys was $283 \times 283\text{m}^2$. Therefore my first job was to generate buoy coordinates in 24×24px area (see Figure 1a), and then scale them up into the coordinates in $283 \times 283\text{m}^2$ area. For example, in the first row of a 24×24px image, the mapping relations are shown as below:

- $(0, 0) \longrightarrow (0.000, 0.000)$
- $(1, 0) \longrightarrow (12.304, 0.000)$
- $(2, 0) \longrightarrow (24.608, 0.000)$
- $(\cdots, 0) \longrightarrow (\cdots, 0.000)$
- $(22, 0) \longrightarrow (270.696, 0.000)$
- $(23, 0) \longrightarrow (283.000, 0.000)$

After this step, I generated 100,000 4-buoy layouts. Then, I submitted those layouts to HPC to calculate their power outputs using the slow CFD model, and I would have the power output for each layout.

Next, as requested, I moved the layout to the center of the area (translation transform, see Figure 1b), and converted the final coordinates into 2-bit (black and white) 28×28px images.

In the last step, I encoded all those 2-bit images into a *.csv file, where each grid in 28×28px image is a column, and the one more last column store the power output. This dataset was forwarded to a postdoctoral researcher of the Machine Learning Lab, who will be looking into this further after the conclusion of our project.

2.1.2 Buoy Layout Study with Fixed Buoys. In Semester 1, I did some layout power output landscape studies on 2-buoy cases, which can be found in Section 4.2.4 in Semester 1's final report. It was done by fixing one buoy's position and moving the other buoy to all possible positions. That helped a lot with the study of energy output distribution.

In 2-buoy case, one buoy is fixed to $(0, 0)$ position, and the other buoy moves. Therefore, in the 3-D plotting, the three dimensions are x-coordinate of the last buoy, y-coordinate and power output. However, the reason why I did not do 3-buoy, 4-buoy or even 5-buoy cases was that, if I fixed only one buoy, there would be more

than three dimensions in those problems. Therefore, they cannot be shown in 3-D images.

In Semester 2, our supervisor came out with an idea of still moving only one buoy and making the reset buoys fixed. i.e. in 3-buoy case, two buoys are fixed; in 4-buoy case, three buoys are fixed; and in 5-buoy case, four buoys are fixed. The fixed locations are also provided:

- In 3-buoy case, the size of the area is $245 \times 245\text{m}^2$, and the fixed two buoys are located at: $(122.5, 0)$ and $(122.5, 50)$.
- In 4-buoy case, the size of the area is $283 \times 283\text{m}^2$, and the fixed three buoys are located at: $(141.5, 0)$, $(141.5, 50)$ and $(141.5, 100)$.
- In 5-buoy case, the size of the area is $316 \times 316\text{m}^2$, and the fixed four buoys are located at: $(158, 0)$, $(158, 50)$, $(158, 100)$ and $(158, 150)$.

The results are shown in Figure 2. In this way, we can see the distribution of wave energy production and apply some strategies on optimization works. We also found the issues in the CFD model which can be seen in Figure 3. There are "spikes" (unusually high energy production) at some locations, and they are symmetrical. I doubled checked the results, in the experiments that I have done, they happened only on 2-buoy and 5-buoy cases. The results have been shared with colleagues from the School of Mechanical Engineering. The reason is still being investigated.

Moreover, in our study, waves come from left. So, from Figure 3b, we can see the differences between placing the last buoy (the moving buoy) "before" and "after" the other buoys. If the last buoy is placed before the other buoys, the wave energy production is less than when it is placed after the other buoys. This general finding is somewhat expected, however, the scale of the impact was unknown beforehand. This observation can later be exploited in custom optimization approaches.

2.1.3 Buoy Layout Grid Search based on Best Layouts. Based on previous power output landscape studies, the optimization team have their current best layouts. However, their best layouts are possible to be local optima. Therefore, our supervisor wants me to move each buoy and do the landscape study respectively. When moving one buoy, the rest three buoys are fixed. Thus the final landscapes can be represented in 3-D image again (see Figure 4).

By looking at those landscapes, we can investigate if any buoy has been placed in a local optimum. However, there is not an apparent local optimum for the existing best 4-buoy layout. So, I run the 16-buoy landscape study again, which is also based on optimization team's best layout. However, the CFD model for one 16-buoy layout takes more than 20 minutes on each layout when calculating the power production on HPC. Additionally, the available area for 16-buoy layout is $566 \times 566\text{m}^2$, which takes much more time than any of previous landscape study. Therefore I choose 4-meter intervals for the grid search. In fact, resulting four 4-buoy figures use 237,537 layouts in total, which take HPC around three days to calculate; while all 32 16-buoy figures (16 elongation figures and 16 power output figures) use 227,519 layouts in total and they take around one week to calculate on HPC, because each 16-buoy layout takes much more time than any 4-buoy layout. Finally, we find obvious local optima in optimization team (see Figure 5a and 5b). This will help the optimization team to improve their algorithms.

2.1.4 Tether Elongation Study. In one of our regular bi-weekly meetings, we found a new issue that we did not take into consideration in previous studies: each buoy had three tethers whose lengths were finite and limited by the buoy's design. In previous buoy design, each tether has an elongation length of three meters, however, some buoy layouts are not suitable for 3-meter-tether buoy because the simulated maximum elongations for those layouts would be greater than 3-meter, and that violates the irresistible physical constraint.

In this case, I added some codes for dumping tether elongations into the CFD model and did the testing on the best 16-buoy layout from optimization team. By doing the same 4-meter interval grid search mentioned in Section 2.1.3, we find that all 16-buoy layouts are infeasible. As is shown in Figure 5c and 5d, in most area the max elongations are roughly flat, but they are mostly less than 4 meters instead of 3 meters.

Above all, this elongation study can show us the distribution of max elongations of specific layouts, and it can help with the optimization works and the design of buoy to some extent.

2.2 OpenDay Side Project

To attend the OpenDay event held on 20th, September 2017, we made a small side project which allowed visitors to make their 4-buoy layout using human brains. Then, they can use their layouts' power outputs to compete with each other and with our machine learning best results. Because this side project is highly related to our main project and it is integrable to our main project, we plan to add it as a small component in our final web app, where people can use human brain to compete with machines. Therefore, this side-project is also shown at Ingenuity event on 31th, October 2017.

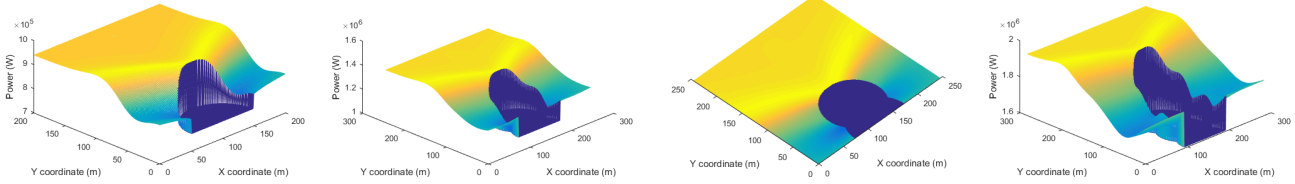
2.2.1 Software Architecture. In this project, we have three main components:

(1) **Front-end User Interface (UI).** This includes a ranking list web page and an Android app for user to make their 4-buoy layout. It is done by Mengyu, using HTML5 technology on both web page and Android app.
(2) **Machine Learning model.** This is done be Chenwei, using WEKA MLP model.
(3) **Web back-end and server maintenance.** This is what I am responsible for, and I use Linux + Nginx + PHP + MySQL to build the ranking system.

Our software architecture is shown in Figure 6. Web ranking page and Android app are isolated, they cannot communicate with each other. The web server receives the user's layout and stores it in the database, then, web page sends a request to the web server for the stored data. Before users' layouts are stored in the database, those layouts are predicted by WEKA MLP model first. So, each layout will have a power production stored with the layout together in the database.

Our front-end shows the top-ranked layouts (see Figure 7), and it is updated every 5 seconds by sending a request to our web server and get back the response which contains the sorted ranking list and buoy coordinates in each layout from our web server.

2.2.2 Back-end. The back-end uses university provided OpenStack™server, running Red Hat Enterprise Linux (RHEL) system,



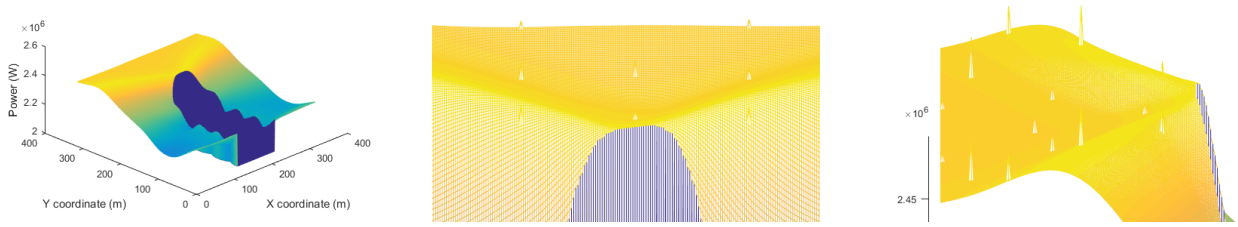
(a) 2-buoy power output landscape with one buoy moving and one buoy fixed at (100, 0).

(b) 3-buoy power output landscape with one buoy moving and two buoys fixed at (122.5, 0) **and** (122.5, 50) **respectively.**

(c) Top view for 3-buoy power output landscape.

(d) 4-buoy power output landscape with one buoy moving and three buoys fixed at (114.5, 0), (122.5, 50) **and** (122.5, 100) **respectively.**

Figure 2: Power output landscapes on 2-buoy, 3-buoy and 4-buoy cases.

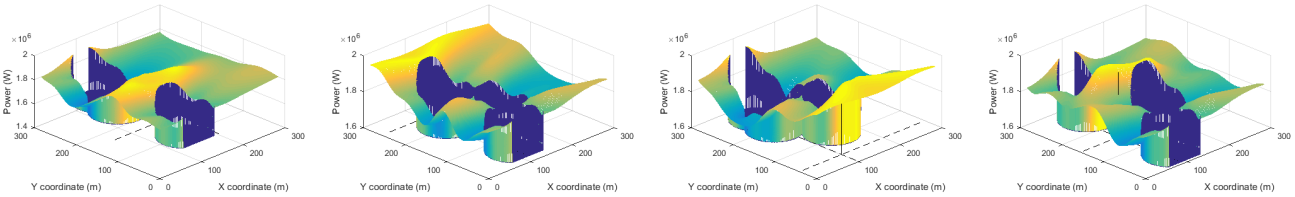


(a) 5-buoy power output landscape with one buoy moving and four buoys fixed at (158.0, 0), (158.0, 50), (158.0, 100) **and** (158.0, 150) **respectively.**

(b) Right-side spike zoomed-in view of 5-buoy power output landscape.

(c) Left-side spike zoomed-in view of 5-buoy power output landscape.

Figure 3: Power output landscapes on 5-buoy case and the previews on spike issues.



(a) Moving the buoy at (141.89, 106.66).

(b) Moving the buoy at (95.24, 236.58).

(c) Moving the buoy at (92.56, 33.01).

(d) Moving the buoy at (83.36, 184.91).

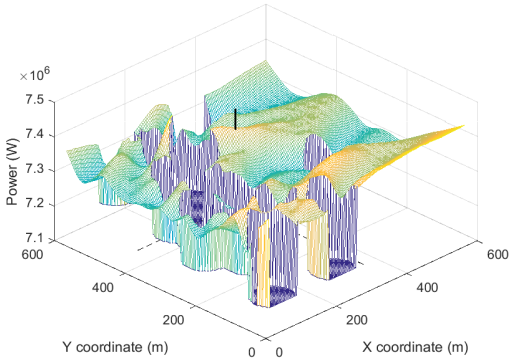
Figure 4: All 3-fixed-buoy grid search landscapes based on the best 4-buoy layout from optimization team. The four buoys are located at (141.89, 106.66), (95.24, 236.58), (92.56, 33.01) **and** (83.36, 184.91)**. In each sub-figure, there is a black reference vertical line, with the height of greatest power output in current landscape, and with the moving buoy's original x-axis and y-axis.**

with "Java Runtime Environment (JRE) 1.8", "PHP 5.4.16" and "MySQL MariaDB distribution 5.5.56" installed. The reason why I use "JRE 1.8", "PHP 5.4.16" and "MySQL MariaDB distro 5.5.56" versions is that those versions are the latest versions that RHEL's official repository provides; and according to the MySQL history [10], I choose the community-supported one - MariaDB.

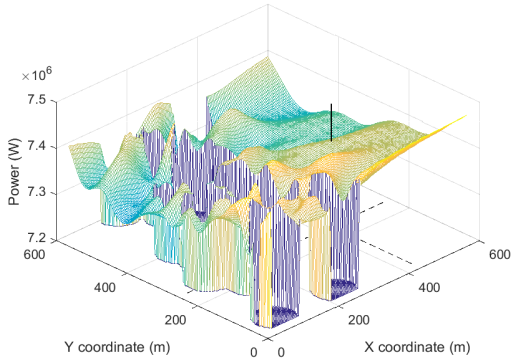
The web server provides a set of APIs for Android development and web development. Those APIs contain the following functions: submitting a new layout, fetching a specified number of top layouts and clear specified number of oldest submissions. Those API documents and examples can be found at the beginning of corresponding source files [1].

I also bind the server IP address with a domain [2], so that when I move the website to a different server, I only need to change the Domain Name System (DNS) record to the new server IP address. Then,

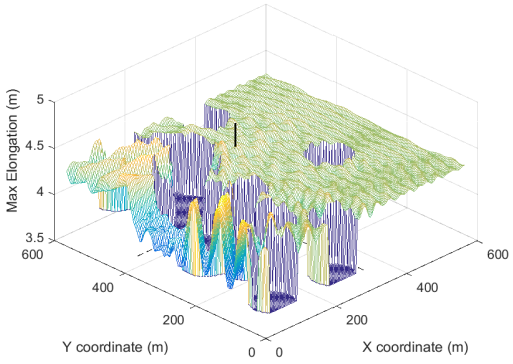
[1] OpenDay side project API source codes and documents: https://github.com/MewX/mse2017-buoy/tree/master/Sem2Websites/OpenDayServer/api.
[2] OpenDay web ranking page URL: https://od.mewx.org/.



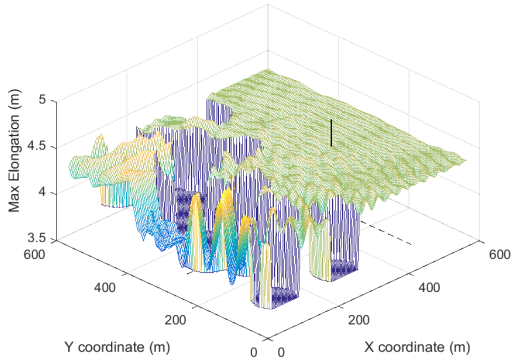
(a) Power distribution when moving the buoy at (277.3, 363.13).



(b) Power distribution when moving the buoy at (408.2, 228.13).



(c) Max elongation distribution during the simulation when moving the buoy at (277.3, 363.13)**. Longest elongation is 4.702639 meters, while the shortest elongation is 3.792125 meters.**



(d) Max elongation distribution during the simulation when moving the buoy at (408.2, 228.13)**. Longest elongation is 4.754083 meters, while the shortest elongation is 3.835580 meters.**

Figure 5: Two of the sixteen 15-fixed-buoy grid search landscapes based on the best 16-buoy layout from optimization team. The 16 buoys are located at (140.12, 565.12), (341.75, 566), (344.66, 514.07), (204.68, 489.22), (224.65, 441.5), (183.68, 383.66), (111.65, 363.49), (277.3, 363.13), (120.92, 310.45), (66.15, 237.04), (173.27, 237.47), (408.2, 228.13), (114, 166.3), (131.67, 114.74), (49.45, 29.2) **and** (219.08, 29.13)**. In each sub-figure, there is a black reference vertical line which plays the same role as the one in Figure 4. Elongation is a new indicator which is introduced in Section 2.1.4.**

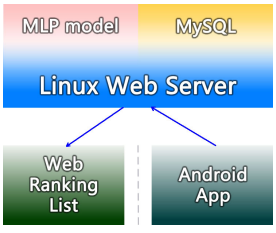


Figure 6: Software architecture of our OpenDay side project.

the applications which are using the domain name for specifying the server can work normally without any modifications.

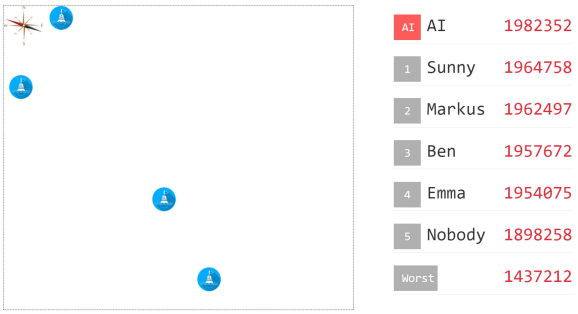


Figure 7: OpenDay side project web ranking list with current buoy layout displayed.

Besides, OpenDay was a public event, thus I added a bad word filter [14] to the back-end. This bad word filter replaces the bad words using asterisks. e.g. "bad words" is replaced by "*** *****".

I also used pre-compiled SQL queries so that it was not that easy to be injected. Most common SQL injection attacks are achieved by embedding SQL queries in data sent to server app; pre-compilation can destroy the SQL query syntax. For example, "' or 1=1" will be converted to "\' or 1=1".

2.3 Main Project

Our main software engineering project is to make a web app for researchers to play around with studied machine learning technologies. This web app is a platform that allows researchers to upload datasets, train models and perform predictions.

In this platform, researchers can test their ideas at any time and anywhere, simply by clicking their mice. For machine learning experiments, tuning parameters is a very time-consuming job. However, with our web app, researchers can simply submit lots of training jobs and wait for their results. Every parameter can be exposed to researchers to tune and inspect the prediction results to evaluate the performance of the model.

2.3.1 Software Architecture. This web app has three main components:

(1) **Web front-end.** This is the UI for researchers to view and perform actions on datasets and models.
(2) **Web back-end.** This is our web serving software, I build it using Linux + Nginx + Java Spring Boot + MySQL.
(3) **HPC back-end.** This is used for long-time and high-spec-required computing jobs, training models and predicting datasets. The programming languages are: Tcsh [19], Java and Python.

I am mainly responsible for the web back-end development, HPC back-end development. Although Chenwei does part of the web back-end development like logging in function and "remember me" function, I work on the most web back-end codes and all HPC back-end developments. For front-end, I only added the password encryption strategy which is introduced in Section 2.3.4 in detail. Mengyu does the most front-end works.

The software architecture is shown in Figure 8. It also shows the relations between all components.

In our system, only web back-end server is entirely under our control; thus it acts as the centre of the whole system. Web browser can send a request to web server and get the response from it, and web server can push message to web browser. However, HPC is a passive server; it can only receive commands and execute it. Therefore, web server must positively read information from HPC.

2.3.2 Web Back-end Development. In Semester 2, we changed the back-end programming framework from plain PHP to Java Spring Boot, while the rest tools remained the same: Nginx + MySQL.

The reasons why I choose those technologies are as follows:

(1) *Java Spring Boot.*

The reason why I performed this change was that back-end was fully managed by me, so I have the privilege to choose what technology to use. The other reason was that I had been writing Java for a few years, but I did not have a chance to write Java web app. So, I wanted to try a Java web framework. Also, our supervisor Markus liked Java, so finally, I decided to have this change.

Using other web scripting languages like PHP/ASPX is not secure because potentially any *.php/*.aspx file dropped into the website directory can be executed without any check. This is also why there are so many web shells used for exploits [15]. In contrast, Java web app is safer because each app is a package, and it is not easy to be hacked simply by uploading something like a *.class file. On the other hand, Java classes can be obfuscated, that means even the server app is fetched by attackers, the codes are still encrypted. The attackers cannot recover the server source codes directly; while PHP and ASPX source codes are not compiled, once attacked, the plain source codes are stolen. So, this is the reason why I changed the back-end framework into a Java framework. Java is proved to be safe and robust in the industry.

However, there are many available choices on new popular Java web frameworks, like Spark Java [3], Dropwizard [4], Vaadin [5], etc. To be honest, they all have their strengths and attractive features, and they are suitable for this project. However, I still prefer to choose Java Spring Boot. Because Java Spring has ten years' history which is the oldest Java web framework [18] among all those popular frameworks, and it is still trendy because of its robust design and a well-maintained community with various types of plug-ins. Moreover, this is my first time to use Java for web development; thus I prefer to start with a mature framework that is used by many developers, so that I can solve almost all issues that I meet by referring to official documents and community answers.

(2) *MySQL.*

I choose "MySQL" for database because this is the database system that I am familiar with the most, and I want to share the database system that I installed for OpenDay side project. However, Java Spring recommends using "H2" database which is light-weight and integrated with Spring very much. So, I am sure that both "MySQL" and "H2" are suitable for this project, and just selected "MySQL".

(3) *Nginx.*

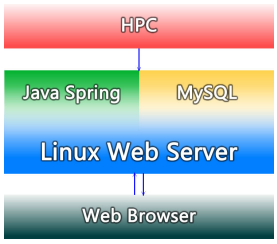


Figure 8: Software architecture of our main project.

[3] Spark Framework (An expressive web framework for Kotlin and Java): http://sparkjava.com/.
[4] Dropwizard Framework (A damn simple library for building production-ready RESTful web services): http://www.dropwizard.io/.
[5] Vaadin Framework (Fight for Simplicity): https://vaadin.com/.

Nginx is a lightweight and fully-functional server software. I choose it instead of apache because I am familiar with it. On our server, front-end developer wants to use static pages with dynamic requests. So I use Nginx as a flexible static-dynamic router. i.e. requests on "/api/*" and "/login" are sent to active Spring Boot service, while the rest are directed to static files. Besides, the reason why "/login" is necessary is that Spring Boot handles this log-in page's authentication process.

With Nginx, setting SSL license for enabling HTTPS is also very easy with "Let's Encrypt" licensing bot. [6] Hence, our main project can be accessed via: https://mse.mewx.org/. As well, the OpenDay ranking web page mentioned in Section 2.2.2 is also under the same SSL license.

In this web back-end, I wrote all the codes except "log-in", "register" and "remember me" functions. When writing those codes, I separated the project classes into the following packages:

(1) *mse*: global classes, most of them are used for overriding Spring system functions.
(2) *mse.controller*: web back-end logics.
(3) *mse.domain*: database table definitions;
(4) *mse.model*: prototypes used for data serialization and transmission;
(5) *mse.repository*: auto handled table instance;
(6) *mse.service*: extensions for basic Spring data functions;
(7) *mse.util*: universal classes used by other classes.

In this way, the packages are arranged in an organized structure, and it can make readers easier to understand my codes.

Followed by the suggestion from Christoph, I set the "crontab" to enable continuous deployment. Every day 12:00 am, the system pulls the latest codes from "master" branch, compile it, and run it as a service. That uses a feature provided by GitHub that a private repository can set deploy SSH keys, which has only "read" access to this repository. Using deploy keys, pulling from a repository does not require entering the password so that it can work non-interactively as a scheduled daily command.

There is a special design in the web back-end. The parameters required for the various machine learning models are different. So, I used Java "Object" class as a unified parameter used in the request (see Figure 9). As you know, "Object" class is the parent class of all classes except for itself, therefore it can represent any class if used as a variable. It is beneficial to be used as parameters for training a model which will be mentioned in Section 2.3.5.

When we use university-provided OpenStack server for the first time, it has a complex network topology to be configured shown in Figure 10. Otherwise, the web server could not access to and be accessed from the external Internet. Due to the rule, I have to create servers under an internal sub-net, then, I need to use routers to allow external access to this sub-net. It is not easy to configure it at the beginning because each step involves with lots of parameters to configured, like choosing a sub-net, assigning IP address for the router, assigning floating public IP address to a server, configuring the security rules, etc.

However, there is a new issue that I found with OpenStack. Servers that I have tried created at any sub-net cannot access any

[6] Let's Encrypt (Free SSL/TLS Certificates): https://letsencrypt.org/.

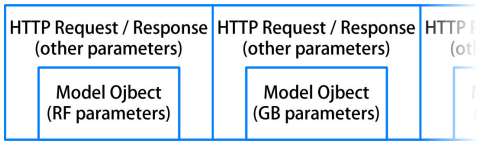


Figure 9: A list of models represented in Java programming language. Each model object is an independent class, where "RF" means Random Forest and "GB" means Gradient Boosting. Different model classes have different parameters (members). By using Java reflection feature, a general object type is elegant for reusing a class.

Figure 10: The network topology of OpenStack web server. Each colour represents a sub-net, and sub-nets cannot communicate with each other by default. The icon between "External" sub-net and "a1700831-net" sub-net represents a router, and the icon under "a1700831-net" sub-net is our web server.

domains under The University of Adelaide, and even IP addresses. That is a huge obstacle for us because we need the access to HPC from web server. I also ask other groups who are using the same OpenStack service like us; they cannot access any domain under our university as well. It is also very urgent because the final demonstration is upcoming, so, I deployed our websites into a backup server, with the domain name: https://mse-bak.mewx.org. Because I have written down the deploy instructions [7], building a secured backup server can normally take less than 2 hours.

2.3.3 HPC Back-end Development. HPC uses "Slurm Workload Manager" to manage and distribute works into different nodes. Time-consuming works should be submitted as jobs into a queue.

In this system, I use HPC as a command line executor via SSH. Every function is completed by executing a command via SSH inputs. Apparently, some works might take much time, like training a model, so, the web server read the output job ID from "slurm". After that, web server regularly checks the status of this job.

[7] Deploy instructions of the main project: https://github.com/MewX/mse2017-buoy/wiki/Deploy-instructions.

So far, all HPC functions are done by me only, and the working directory is configured at the "Constants" class in the main project. During the development and demonstration, the working directory is at "/fast/users/a1700831/mse/".

- Dataset-related functions, such as generating new dataset, partially fetching contents from a dataset, and pre-processing a dataset (checking the format of the dataset and detect whether the dataset has an expected output in each record), are written by me using "tcsh" and "Java".
- Current supported machine learning models are Random Forest and Gradient Boosting based on "Scikit-learn", and they are all done by me. In our original plan, models from Mengyu, Chenwei and me will be all supported. However, Mengyu and Chenwei have not successfully turned their models into a fully command-line trainable model, though WEKA MLP can use the command line to predict.
- Predictions use the same command line arguments for Random Forest and Gradient Boosting, taking a dataset and a model, and predicting into a new dataset.

Input checking is also an essential step before running anything on a server. Parameters sent to the tools on HPC are validated by the tools themselves before they are fully running, because it is convenient for developing and debugging, the error message is around the where it is used.

To connect HPC from web back-end, I am not using Linux system default SSH client, because it is an external program. Instead, I am using a Java implementation of SSH client - JSch [8]. It runs as a native Java program, so, developers can control all the messages and events with the connect.

While developing programs on HPC, I met a bug caused by "Slurm" package manager (see Figure 11). It seemed that it could not add the Java path into the environment variable, so I could not invoke Java in this way. The solution is to use the absolute path. When I load Java after logging in, I can use "which java" to find the Java command location. Then, I can update the absolute command in "Constants" class in the main project. The reason why I did not find this issue in Semester 1 was that this issue happened only when loading Java. In Semester 1, I only used remote Python commands, and loading Python via SSH command worked normally.

2.3.4 Web Front-end Security. Nowadays, our web security highly depends on the HTTPS protocol which encrypts the data transferred on the network between browsers and web servers. However, using HTTPS protocol does not mean that it is safe.

In Figure 12, a middle man can pretend to be a hidden link between the client and server, listening and forwarding requests and responses. Even though the web browser will notify users that this web server's certification is not trusted, in one simplest case, if this middle man simply removed the HTTPS protocol, the whole connection becomes a public one. Everything transferring on this link is open to everyone, including usernames and passwords.

Besides, the key is not fully-trustable. Recently WPA2 protocol, which is widely used among all Wi-Fi devices, has been attacked that in some cases a tamper private key can be used between hotspots and Wi-Fi devices [16].

To avoid most potential attacks and to protect the safety of user's password. I added the front-end password encryption. Under this circumstance, the password is firstly hashed by JavaScript SHA-256 function [8], then the hash results will be used as the users' passwords to send to the server (see Figure 13).

The reason why I choose to use SHA-256 is that it is one of the most secure hash algorithms among all popular hash algorithms.

Popular hash algorithms are mostly from SHA (Secure Hash Algorithms) family and MD (Message Digest) family. Their latest versions are SHA-3 and MD5 respectively. However, SHA-3 is still secure now, but its core function - "Keccak-f" exists a full-24 round attack [2]. Although this attack does not affect whole SHA-3 algorithm, it is a threatening. Regarding MD5, it has been proved that its distribution is not safe since 2004 [7]. That means it is highly possible to find two input strings having the same MD5 hash results. Later in 2013, Xie *et al.* published a method for a very quick collision attack [24], and it takes less than a second on a regular computer.

So far, SHA-256 has the best attack on 57-round (out of 64 rounds) [9]. Therefore, it is still secure to use it as the hash function now.

2.3.5 Interface. In this project, front-end is separated from web back-end. So, there are two interfaces which can be evidently seen in Figure 8: the one between browser and web back-end, and the one between web back-end and HPC back-end. As well, I designed and implemented all functions in the two interfaces by myself.

The interface between the browser and web back-end in this project is called Application Programming Interface (API). Legacy web app does not require API because front-end and back-end work together: back-end declared interface in front-end templates; then the front-end developer can work on those templates.

However, in this project, our front-end developer Mengyu is not familiar with Java Spring template engine - Thymeleaf [9]. So, after lots of discussions, we decide to separate the two parts entirely.

I maintain an API document [10] which describes the requests and responses in detail so that Mengyu can refer to it at any time. I try to make the API follow *Microsoft RESTful Guidelines* [11]. For example: to get the list of datasets that the currently logged-in user has, the API method is "GET https://mse.mewx.org/api/dataset/list"; also, to create a new model, the API method is "POST https://mse.mewx.org/api/model/create" with model parameters.

All parameters are encoded in JavaScript Object Notation (JSON) so that front-end JavaScript and other programming languages can quickly convert it into variables. I highly look at this point, because machine learning models implemented in different languages require different parameters, in this way, the web back-end can simply forward the input JSON string into the machine learning model trainer script on HPC.

When developing back-end, I set the HTTP response code according to the *Hypertext Transfer Protocol* document [6]. Each API request has those HTTP state codes handled:

- Code 200: everything is fine with the request.
- Code 400: the request content format is not correct.
- Code 403: the request requires logging in.
- Code 404: the target page does not exist.
- Code 415: when receiving unwanted content type.

[8] A simple SHA-256 hash function for JavaScript supports UTF-8 encoding: https://github.com/emn178/js-sha256.
[9] Thymeleaf (a modern server-side Java template engine): http://www.thymeleaf.org/.
[10] Main project API: https://github.com/MewX/mse2017-buoy/wiki/API.

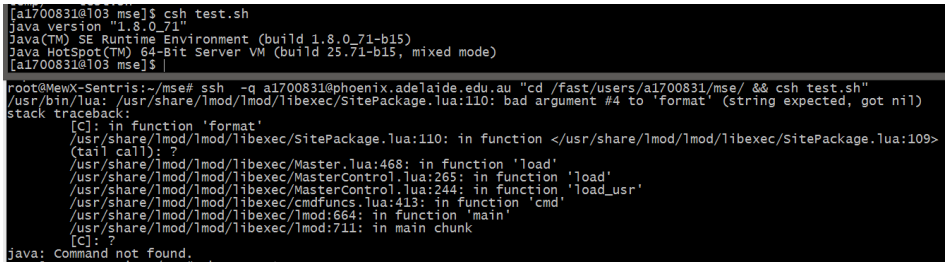


Figure 11: Slurm Java package loading bug: executing the scripts while logged works normally; while sending the command stream via SSH command cannot execute successfully.

Figure 12: The potential middle-man-attack issue in current browser-server architecture.

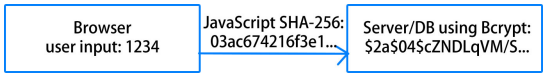


Figure 13: The process how the password is encoded in the whole system. The password is firstly hashed from user browser by SHA-256, then it is sent to the server and hashed by Bcrypt. Next, it is stored in database or verified by comparing with the hashed password in database.

Besides, the other HTTP state codes like "5xx" codes are handled by Nginx automatically.

The interface between web back-end and HPC is the SSH tunnel. It simply works by sending HPC commands from web back-end to HPC. This is mentioned in detail in Section 2.3.3.

2.4 Evaluation and Testing

2.4.1 Research. Research works are mostly for data processing. To evaluating the results, they are firstly checked by myself using result validators in my programs, then they are presented to our supervisors and research colleagues via email; evaluations, feedbacks and analysis are directly shown in email groups.

2.4.2 Software Engineering. I try my best to follow Test Driven Development (TDD) in scrum development, and write test cases to evaluate the functions that I just finish. Every test must be passed before being committed or merged into "master" branch. As well, I use a code testing coverage tool to evaluate the quality of testing (see Figure 14).

Moreover, every fortnight is a sprint, and I publish a release on GitHub release page. Besides, all latest version on "master" is also working which is a nightly build, because we work every day.

Furthermore, I use meaningful variable and function names while coding, and there are necessary comments. For example, the comments explain how a model is created when receiving the "create model" request. The extensibility is also scientific that every module

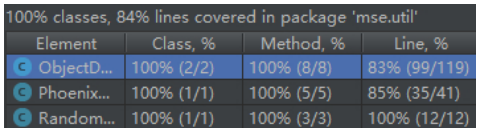


Figure 14: Test coverage report from IntelliJ IDEA on "mse.util" package.

is separated with each other, especially the design shown in Figure 9 which supports unlimited future add-ons.

However, due to the time limits and my workloads in this project, I am not able to finish all function testing while developing. So, it involves with the trade-off between code quality and completion of functions; in most time, I choose to finish functions first. As a result, I am appending the lost testings now.

2.4.3 Web Application Security. Based on a checklist from Microsoft [11], I have carefully handled the prevention of SQL injection, server file system permissions, Cross-Origin Resource Sharing (CORS) strategy, data encryption and so on. Therefore, the back-end security is better than most websites.

3 RELATED WORK

In Section 3 of Semester 1's final report, I mentioned the related works about buoy design and previous optimization works on wave energy. However, wave energy is still a new type of energy, and it has not been largely explored; using machine learning technologies in energy prediction is still rare. Even they use machine learning, their researching goals are different from us. For example, Fernandez *et al.* did a research [5] for predicting the wave height and energy flux range using ordinal classifier method; Dripta *et al.* [12] worked on 40-buoy layout using optimized physics models.

Our main project is an online machine learning platform. Unlike existing commercial machine learning platforms from Amazon [12], Google [13], Qwiklabs [14] etc., our platform is specially designed for solving our specific research problem - finding best buoy layouts.

To do the project, existing software are essential for accelerating the developing speed. Those software include:

[11] Checklist for Security Best Practices: https://technet.microsoft.com/en-au/library/bb735870.aspx.
[12] Artificial Intelligence on AWS: https://aws.amazon.com/amazon-ai/.
[13] Google Cloud AI: https://cloud.google.com/products/machine-learning/.
[14] Qwiklabs (online machine learning labs): https://qwiklabs.com/.

- Linux operating system: used by web server and HPC;
- Google and StackExchange: used for finding solutions to troubles while developing;
- Security Protocol and cryptography algorithms: SSL and HTTPS are used all the time;
- Fiddler 4: used for API debugging and testing;
- IntelliJ IDEA and PyCharm: used for developing web back-end and machine learning models;
- MATLAB and GIMP: used for editing images;
- Open-source frameworks: Java Spring for web development and Scikit-learn for machine learning.
- GitHub, ZenHub and Slack: used for efficient communication and scrum development.

without those handy tools, I can hardly imagine how long this project would take. They simplify the development works, and as a result, they improve the development speed a lot.

4 REFLECTION

The project from Semester 1 to the end of Semester 2 contains a research component and a software component.

For research, I realise that research is a tough and unpredictable work. Usually, demanding results are not easy to get; and sometimes, research requires a co-operation of people from different areas. Also, ideas are often inspired by other's work, just like the choice of Random Forest mentioned in Section 4.1.4 in Semester 1's report.

For software, I think the management of workloads was not handled very well; the workloads of every team member are not average that I had too many workloads. Moreover, I feel like overworked: I designed and implemented the whole API system, however, only a part of them were used by front-end developer. Although I always wanted to follow TDD, I did not have enough time for that. That resulted in lacking unit testing codes, and it finally proved that these codes were very likely to contain bugs.

To avoid this issue, in future projects I must consider the required time carefully, including learning time, designing time, coding time and debugging time; teammates should also help with each other even when the tasks are decided at the beginning. Also, to ensure the agile progress, I learned to separate works on average, instead of trying to finish everything just before the meeting.

However, I do not regret that I spend quite a long time learning the big popular Java Spring framework, because it does simplify my back-end development work and save me much time to build a reasonably secure web app.

5 CONCLUSION

This is a priceless chance to know and work on the new renewable energy - wave energy which looks so powerful, and it motivates me to contribute more. However, this project is also somewhat challenging, because I need to develop on both Linux web server and HPC. Plus I am using a back-end framework (Java Spring) that I have never used before; it adds the difficulty to me.

In this project, I learned the methodology of doing research, and I was involved in training machine learning regression models, doing data processing and data visualization works. They mostly were new to me, so I am happy to gain those experience.

I think I have done well on software development and document maintenance because I followed best practise in software engineering. Also, it is my first time to work with HPC and enterprise service RHEL OpenStack platform, though some configuration confused for me, I overcame all of them and gained lots of precious experience.

Moreover, I am very grateful for the guides and help from my supervisor Markus, course coordinator Christoph, tutor Marian and two teammates Chenwei and Megnyu.

To conclude, this project is challenging and in the front line of research. I have learned lots of new technologies and tools from this project. In the future, if this project is continued by someone else, I am also confident that the codes are easy to be maintained and extended!

REFERENCES

[1] Dídac Rodríguez Arbonès, Boyin Ding, Nataliia Y Sergiienko, and Markus Wagner. 2016. Fast and effective multi-objective optimisation of submerged wave energy converters. In *International Conference on Parallel Problem Solving from Nature*. Springer, 675–685.

[2] Christina Boura, Anne Canteaut, and Christophe De Canniere. 2011. Higher-Order Differential Properties of Keccak and Luffa.. In *FSE*, Vol. 6733. Springer, 252–269.

[3] CSIRO. 2017. CSIRO projections show that wave energy could provide a viable contribution to Australia's future energy mix. (oct 2017). https://www.csiro.au/en/Research/OandA/Areas/Marine-technologies/Ocean-energy/Wave-energy

[4] Boyin Ding, Leandro Souza Pinheiro da Silva, Nataliia Sergiienko, Fantai Meng, Jonathan David Piper, Luke Bennetts, Markus Wagner, Benjamin Cazzolato, and Maziar Arjomandi. 2017. Study of fully submerged point absorber wave energy converter-modelling, simulation and scaled experiment. (2017).

[5] Juan Carlos Fernández, Sancho Salcedo-Sanz, Pedro Antonio Gutiérrez, Enrique Alexandre, and César Hervás-Martínez. 2015. Significant wave height and energy flux range forecast with machine learning classifiers. *Engineering Applications of Artificial Intelligence* 43 (2015), 44–53.

[6] Network Working Group. 1999. Hypertext Transfer Protocol - HTTP/1.1. (jun 1999). https://tools.ietf.org/html/rfc2616#section-10

[7] Philip Hawkes, Michael Paddon, and Gregory G Rose. 2004. Musings on the Wang et al. MD5 Collision. *IACR Cryptology ePrint Archive* 2004 (2004), 264.

[8] JCraft. 2016. JSch - Java Secure Channel. (sep 2016). http://www.jcraft.com/jsch/

[9] Dmitry Khovratovich, Christian Rechberger, and Alexandra Savelieva. 2012. Bicliques for preimages: attacks on Skein-512 and the SHA-2 family. In *Fast Software Encryption*. Springer, 244–263.

[10] MariaDB. 2017. About MariaDB. (oct 2017). https://mariadb.org/about/

[11] Microsoft. 2017. Microsoft REST API Guidelines. (aug 2017). https://github.com/Microsoft/api-guidelines/blob/vNext/Guidelines.md

[12] Dripta Sarkar, Emile Contal, Nicolas Vayatis, and Frederic Dias. 2016. Prediction and optimization of wave energy converter arrays using a machine learning approach. *Renewable Energy* 97 (2016), 504–517.

[13] JT Scruggs, SM Lattanzio, AA Taflanidis, and IL Cassidy. 2013. Optimal causal control of a wave energy converter in a random sea. *Applied Ocean Research* 42 (2013), 1–15.

[14] Snipe. 2017. banbuilder: Composer package for censoring profanity in web applications, forums, etc. (aug 2017). https://github.com/snipe/banbuilder

[15] John Troony. 2016. php-webshells: Common php webshells. (apr 2016). https://github.com/JohnTroony/php-webshells

[16] Mathy Vanhoef and Frank Piessens. 2017. Key Reinstallation Attacks: Forcing Nonce Reuse in WPA2. (2017).

[17] Wikipedia. 2017. Ocean. (oct 2017). https://en.wikipedia.org/wiki/Ocean

[18] Wikipedia. 2017. Spring Framework - Wikipedia. (oct 2017). https://en.wikipedia.org/wiki/Spring_Framework

[19] Wikipedia. 2017. tcsh - Wikipedia. (oct 2017). https://en.wikipedia.org/wiki/Tcsh

[20] Wikipedia. 2017. Wave power. (oct 2017). https://en.wikipedia.org/wiki/Wave_power

[21] Wikipedia. 2017. Wind. (oct 2017). https://en.wikipedia.org/wiki/Wind#Wind_force_scale

[22] GX Wu. 1995. The interaction of water waves with a group of submerged spheres. *Applied ocean research* 17, 3 (1995), 165–184.

[23] Junhua Wu, Slava Shekh, Nataliia Y Sergiienko, Benjamin S Cazzolato, Boyin Ding, Frank Neumann, and Markus Wagner. 2016. Fast and effective optimisation of arrays of submerged wave energy converters. In *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference*. ACM, 1045–1052.

[24] Tao Xie, Fanbao Liu, and Dengguo Feng. 2013. Fast Collision Attack on MD5. *IACR Cryptology ePrint Archive* 2013 (2013), 170.