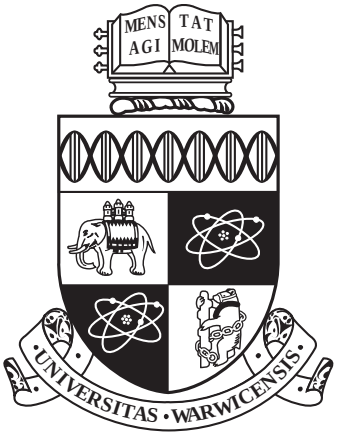

Efficient Allocation of Renewable Energy Sources Under Uncertainty in the UK

James Page

1906422

Supervisor: Long Tran-Thanh

Department of Computer Science

University of Warwick

2021-2022

Abstract

With climate change continuing to be an unsolved issue, the importance of installing new renewable energy generation facilities is ever increases. When planning the installation of these new systems, generation efficiency is the main consideration. We investigate methods to ensure this efficiency when considering a range of location where the performance of the renewable energy is not certain. Using regression techniques to predict wind turbines generation amount with respect to weather features and a genetic algorithm approach we show that these AI techniques are well suited to tackling the large search space this problem presents and allows for an insight into the most suitable locations for future funding in the renewable energy sector.

Keywords: Machine Learning, Genetic Algorithms, Regression, Allocation, Renewable Energy, Wind Power

Acknowledgements

I would like to thank my supervisor Long Tran-Thanh for the input, suggestions and support throughout this project, all allowing me to explore new ideas I would not have considered alone.

I would also like to acknowledge Maply (www.Maply.com) as we have made use of their map data visualisation tool throughout this project to help understand our results, as well as VisualCrossing.com for providing an extremely useful source for our data set.

Definitions

In this report we will be making use of a range of specific keywords throughout which may not be obvious in their meaning, therefore we will be defining them here:

- Unit of Generation: A measure of generation capacity describing exactly one standardised capacity generator, i.e. When considering wind turbines, one unit would be a 2.5MW capacity wind turbine.
- Efficient/Efficiency: The ratio of cost to generated electricity amount, i.e. more efficient implies a larger amount of electricity per unit of generation installed.
- Budget: The number of units of generation we allow the program to allocate as maximum.
- Allocation: The number of units of generation selected to be installed in each candidate location, i.e. Location A - 2 Wind Turbines, Location B - 0 Wind Turbines, Location C - 10 Wind Turbines.
- Energy Type/Source:
- Load Factor:
- Pipeline

Contents

Chapter 1

Introduction

1.1 Problem Statement

Due to the ongoing threat of climate change in the current day, the focus on carbon neutral energy sources has never been higher. As such, weather dependent renewable energy generation is demanding a rising amount of investment to transition away from traditional carbon-based fuels. With this rising investment and the importance of these new energy sources, the generation efficiency of the sources installed with this investment is as high as possible as to not waste the opportunity presented. The problem is that the most common renewable energy sources, Wind Solar are inherently weather reliant, introducing the uncertainty and inconsistency often referenced in any push back against these energy types.

With the aim of achieving a high efficiency we discover two main problems, how do we predict the amount of energy a location is going to produce, and how do we find a set of locations to optimise the overall generation amount? This presents us an intractable problem evaluating every combination of locations to find the best possible solution, a problem that AI techniques are very well suited for. However, "AI techniques" covers a wide range of approaches, some much more suitable than others so we will be investigating the best combination of algorithms and implementations to ensure our solution is scalable, consistent, and reliable.

In this report we will detail how a combination of existing techniques can be changed to be applied to a real-world use case, with the goal of producing a solution that can return us an "optimal" configuration of renewable energy sources. This will include two main sections of

implementation, each focused on one of the two main challenges this problem poses us: Generator Output Prediction; and Optimal Location Selection, each requiring a unique approach with their own issues to tackle.

1.2 Related Works

The idea of approaching environmental challenges such as renewable energy as optimisation problems is not a novel one. There have been a variety of past works that have approached similar problems and examining the contributions they have made will allow us further insight into our problems and solution.

Şerban and Lytras's (1) 2020 paper explores an assortment of ways AI can be adopted into the renewable energy sector and the impacts it could have. Their examination concludes that the use of AI at a low level could help reduce the variability of renewable energy by working with large amounts of data. They believe this can help lead towards a more sustainable energy sector but point out there is still much work to be done in areas such as macro scale implementations helping to optimise energy grids and the development of more sophisticated optimisation algorithms.

An article in 2019 using a case study of the Pakistan national grid (2) showed how optimisation algorithms could be used to try all combinations of different energy and storage types to find a suitable configuration. This approach allowed them to simulate the performance of different configurations using hourly meteorological data, with the results showing that Wind Power can meet annual demand with 30,000MW of capacity installed if sufficient storage solutions are also installed.

One of the benefits of offshore wind power is the amount of open space and location flexibility it often gives you for the trade off of cost. Using this flexibility in placement, a 2012 paper explores the option of using a Genetic Algorithm approach to improving the positioning of individual turbines in a farm (3). The fitness function modelled the performance of a configuration based on it's load factor, which proved to guide the algorithm to an optimal configuration after 1000 generations.

Some of the more recent publications on the area, are the "Energy Conversion and Management" articles on using extreme gradient boosting to predict daily solar radiation, and the use of Deep Reinforcement

Learning forecast based energy storage systems. The prior (4) shows that the use of a generalised 'XGBoost' model using temperature and geographical data could predict daily solar radiation more accurately than a similar empirical model and was a promising new approach for use in locations that have no historical data. The former finds the Reinforcement learning approach employed showed promise in controlling the system while accounting for a range of variables such as wind turbine generation, it did struggle with balancing competing the objectives when subjected to restricted degrees of freedom. (5) This shows how it is important to acknowledge that a technology such as deep reinforcement learning is not the silver bullet for solving complex problems and that while successful in other areas it is a perfect solution here.

From these exploring these related works we highlight the importance of the application of AI to the renewable energy sector, and show how there are a wide range of varying technologies being applied in different ways. Where possible, we hope to draw on these ideas as inspiration for tackling our own problem where similar scenarios may show themselves.

Chapter 2

Background

To understand how to model this problem appropriately we need to understand the background and intricacies of this area. Specifically, the areas requiring further research are the types of renewable energy currently being invested in; their pros and cons; locations that are being invested in; requirements for setting up new renewable energy source; and how weather factors can impact generation. This information will guide our decision making to ensure that we are describing a problem that we can feasibly solve while still giving us meaningful insight into the problem.

2.1 Renewable Energy Types

There are a wide range of renewable energy sources in use across the UK. However, to keep this project focused we will initially focus into the most common weather-based sources. In 2021 in the UK the highest growth in capacity was in Onshore Offshore Wind, closely by Solar Photovoltaics (6). We also see that of the reported renewable energy types, the most electricity generation came from Wind energy, making up 15.1% (onshore & offshore combined) of the UK's electricity generation in Q3 2021. Next highest was Bio-energy related generation, however as we are focusing on the impact on location-based factors such as weather we will not be considering bio-energy as a candidate energy type. Finally, the last sizeable portion of the renewable energy generated was made up by solar photovoltaics (PV) which generated 6.2% of the UK's energy in the same period.

From these two energy types, we can examine the difference in costs. As we are looking to make the most efficient use of an investment the

first step to doing that is to ensure we are using the most cost-effective generation method. From the International Renewable Energy Agency (IRENA) generation costs report in 2020 we can compare the costs per kW of capacity of different technologies in different countries (7). For our case, the relevant values are the price per kW of solar PV, onshore and offshore wind in the UK. From this report we can see that in the UK the cost of commercial scale solar PV was 1545 USD/kW with the cost decreasing year to year at a rate of 7%. Extrapolating that cost trend gives us a figure of 1336 USD/kW in 2022. To verify these values, we can compare to installation quote estimations from greenmatch.co.uk (8) which gives a cost of 1337.67 USD/kW (when including a 20% VAT) for a 60kW installation, close to the extrapolated value from the report, giving us further confidence in the values it presents. Viewing similar values from the report for offshore wind is by far the most expensive in our considerations with a cost of over two times that of any other at 4552 USD/kW. Finally looking at onshore wind, we see that in 2020 the cost in the UK was 1710 USD/kW making it more expensive than solar, however, when we factor in the 13% year to year price decrease and extrapolate to the current year as we did for solar, we see that the cost of 1294 USD/kW is the cheapest energy source we look at. These downward price trends are a result of increasing adoption and investment into the technology driving advancements that allow for the prices to drop.

Another consideration is the generation efficiency of these energy types. By looking at the reported load factor of each energy type in the UK we can determine which of the above will give us the best cost to generation amount, in a way that better reflects the results we'd actually see rather than just looking at capacity. These numbers will also give us an idea of the performance of currently installed systems in the UK, allowing us to determine if our approach could provide improvements to current decision making processes. From the UK government annual reports on renewable generation (9), we can see the load factors of all energy types we have been considering by year since 2009. By far the lowest performing in the UK is solar. With an average load factor of 10.86% we can safely add another factor to our list of why it is not a ideal candidate for us to consider. On average the best performing option was offshore wind power with an average load factor at 39.2% however this 12% performance benefit over its onshore counterpart is not enough the make up for its significantly larger cost per kW of capacity. This again highlights the all around strength of onshore wind power, showing why it is at the forefront of UK renewable investment.

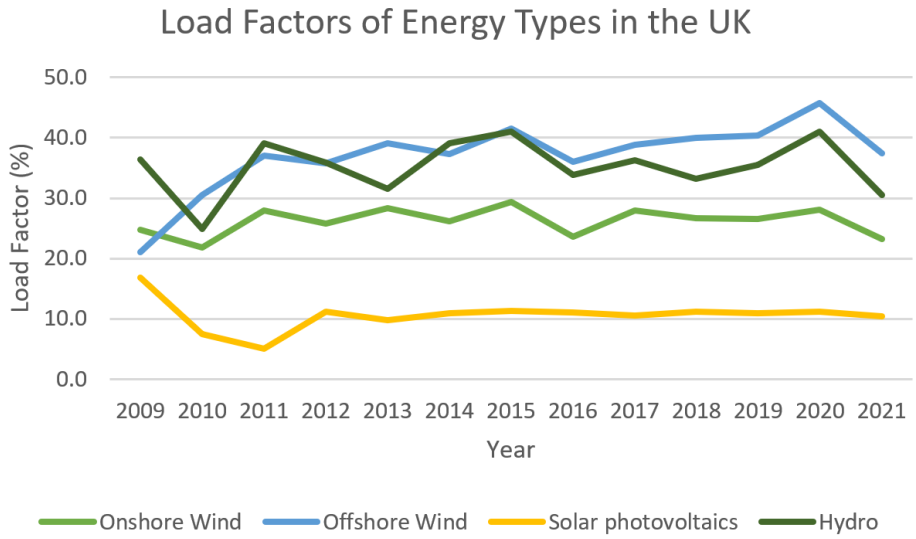

Figure 2.1: Load Factors of select energy types in the UK. (9)

Looking at all this information we can see the most viable candidate for this project is onshore wind power, simply due to it currently being the most relevant and invested in renewable technology and it being on track to be the cheapest per unit of generation. This decision will allow us to be confident in our results being relevant as guidance in the real world while keeping the scope limited to a fixed energy type.

2.2 Locations

Just as important as the different energy types, the candidate locations we consider for investment will have a major impact on the feasibility and performance of our solutions. As we are modelling a real-world scenario, price of land, land ownership and planning permission are variables that we should take into consideration. If we ignore these, we risk finding a result that cannot be implemented in the real world due to prohibitive land costs or some restriction in what the proposed location can be used for.

To work around these restrictions, we decided the most appropriate approach for this project would be to continue under the assumption

that the locations provided to the program are preemptively vetted for issues that might cause lead to illegibility. This allows us to remove potentially subjective case by case constraints from our model of the problem, simplifying it significantly. We do not think this will impact the appropriateness of our model as we believe that our solution would be most useful in aiding investment decisions between locations already under consideration rather than being a tool to conduct an in-depth search of all possible locations in the UK.

However, to evaluate our results, we need to ensure that in this report we give the program a set of locations to consider that already exist within the real world. By letting the program consider existing locations we can show that our results are based on sites that are eligible removing a layer of uncertainty in translating our results to real world performance. The locations we have selected have been extracted from a data set acquired in the data collection stage. These locations are arbitrary and could be replaced with any other existing locations. However, as we would be producing a list of wind turbine locations for use in a training data set anyway, we saw no reason to not use them as they covered a range of coastal, flat and mountainous regions. We will further discuss in detail the process of collecting data for suitable locations in the implementation section.

2.3 Scope

As discussed above, we will be restricting the scope of this project to allow us to focus on achieving the best possible results in this narrower field of investigation. The restrictions are that the only energy type we will be considering from this point on will be onshore wind, and that the locations the program is allowed to consider will be predetermined with the program having no capacity to model location specific factors in it's choices. As such, the refined problem we will be tackling is choosing the most efficient allocation of wind turbines across our predetermined locations (shown in figure 1). We believe these restrictions will not impact the importance of our results. This is because despite the restrictions, the methods highlighted in this report will allow for modification to work for other energy types in future works. Therefore, by spending more time on getting the best results in this narrower investigation we allow ourselves the possibility of improved results in future iterations. We have also discussed above how in the main use

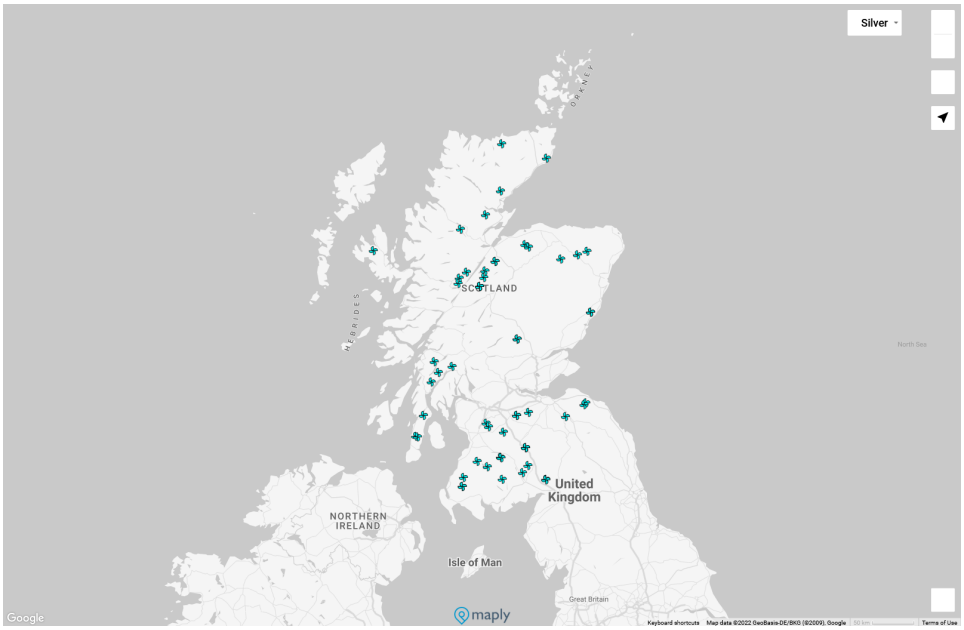

Figure 2.2: Wind Farm Locations Being Considered as Candidates

case for this work, it is reasonable for candidate locations to already have been selected before the stage that our solution would be in use.

Chapter 3

Methodology

3.1 Task Definitions

The problem we are tackling can be split into two distinct sections, each with their own challenges that will be recombined into a final program.

3.1.1 Generation Prediction

The first problem we are approaching in this report is predicting the performance of a wind turbine based on the weather. Formally, we want to define a function that takes as input a vector W of weather features and outputs a load factor $l \in [0, 1]$.

$$W = \begin{bmatrix} WSpeed \\ WGust \\ Precip \\ AirPressure \\ CloudCover \\ Visibility \\ Humitity \\ Temperature \\ Dew \end{bmatrix}$$

$$f(W) : \mathbb{R}^{9\times 1} \rightarrow [0, 1]$$

This vector of weather features will consist of 9 attributes: Temperature, Wind Speed, Wind Gust Speed, Precipitation Amount, Air Pressure, Cloud Cover, Visibility, Humidity, and Dew. To built this function we will be taking a machine learning approach to build a regression model that can closely predict the load factor expected given the weather conditions. To do this we will require a large training data

set made up of real world examples based upon real data, with enough variation and data points to ensure our model is neither over-fit nor under-fit.

3.1.2 Allocation Optimisation

The larger problem of the two is searching for the optimal allocation. This is a complicated problem due to the extremely large combinatorial search space we are working in. In this problem we have a set of n locations L, and a maximum budget B such that:

$$B \in \mathbb{N}$$
$$L = \{l_1, l_2, ..., l_n\}$$
$$\forall i \in [1, n], l_i = \begin{bmatrix} loadFactor(i)_{mean} \\ loadFactor(i)_{variance} \\ loadFactor(i)_{min} \\ loadFactor(i)_{max} \end{bmatrix}$$

We define an legal allocations A as:

$$A = \{a_1, a_2, ..., a_n\}$$
$$\forall i \in [1, n] : a_i \in \mathbb{N} \cup \{0\}$$
$$\sum_{i=1}^{n} a_i \leq B$$

Where each a_i denotes the number of units of generation located location l_i. If we take the assumption that all of the budget is allocated, the number of possible allocations C grows exponentially with the budget and number of locations.

$$C = \Theta(n^B)$$

This search space is far to large to brute force for anything more than comparing a small handful of locations at a small budget, therefore we want to find a heuristic method to maximise a selection of objective values defined by function p that measure the performance of a given allocation.

$$p(A) : A \to \mathbb{R}$$

As such we need to define a function g, taking a set of locations and a performance function as input which outputs an optimal allocation in regard to the performance function P.

$$g(L, p) : L, p \to A$$

Such that there exists no allocation $a' \in A$ where $P(a') > P(g(L, P))$.

3.1.3 Objectives

To measure the success of our project we set ourselves the following objectives, these allow us to track our progress and evaluate our success.

Generation Prediction

- We will construct a data set of 5000+ entries that consists of 9 weather variables and a target load factor value.
- When using a 60-40 train-test split of the data set, the prediction model (trained on the training set) will achieve a mean-squared error of less than 0.025.
- Once the model is initialised, a prediction can be calculated in a constant time < 0.05 seconds.

Allocation Optimisation

- Upon termination the program will return an allocation that is strictly better than any other in the search space it has covered.
- The simulated load factor of the resulting allocation will be equal or greater than the real world load factor of 26.58% (average over 2012-2021 (9))
- The search should be parameterised by variables allowing the user to adjust the length of the search to their needs. However shorter search lengths should be expected to return worse results.

3.2 Implementation

3.2.1 Data Collection

As we stated in the background research section 2 many of our choices rely on the availability of data. This is one of the major requirements for a machine learning approach, and therefore the data collection process was paramount to the success of the rest of the project. We had two types of data we needed to collect, historic meteorological data and past records of the electricity generated by wind farms. The second where most of the restrictions we faced were introduced in terms of locations.

In order to obtain this data we made use of API access to two data services. For weather we used a service called VisualCrossing.com (10),

which allows for a detailed record of weather events across the country at a 15 minute resolution. This allowed us to build a comprehensive data set comprised of a range of times of day, months and years. The other API we made use of was the Elexon Balancing Mechanism Reporting Service (11), here we could request a range of different data about each wind farm, including the actual generation amount again at a 15 minute resolution. Paired together these two APIs gave us access to everything we would need to build a sufficiently large training data set for our regression model.

The locations we were requesting this weather data for, was dependent on the availability of past generation records. Our candidate locations were sampled from a data set of future wind turbine availability from on data.nationalgrideso.com (12), this gave us a list of ID codes of wind turbine generators across the UK. With the help of an online tool (13) we created a list of the wind farms associated with these IDs along with their locations, and generation capacity. This initial list consisted of 194 wind power generators, however when we removed entries where we could not find record of a farm name or location and any offshore wind farm entries this dropped to 99. When we began the data collection process however we ran into an unexpected problem where some generation amount requests were returning errors. We discovered that some of our location set did not have records for actual generation amount, after rechecking the API for any other data points that would give us the information we required we decided that a smaller location set would be acceptable as after cutting out the problematic locations we still had 56 Locations across Scotland (seen in Figure 2.2).

One limitation with our approach is the restrictions that come with using a free account on VisualCrossing.com. This restricts the number of requests per day meaning time was a limiting factor in the size of our data set. By automating the request process however, we were able to continuously add to our data set while progressing onto other work. Our final data set consists of 7450 entries distributed across 56 locations. By building our training set across different locations we hoped to get a wider variation in weather events, with the hope that this will allow our model to more generalised. This should allow it to train on a variety of values for each of our weather features. In future extensions of this work building a larger data set across even more locations could work to further generalise the model and improve accuracy in unknown scenarios.

3.2.2 Predictor Model

Regression

To make generation predictions as described in section 3.1.1 we will be using a machine learning technique called regression. This is a category of techniques that attempt to model the relationship between some input features X and an outcome y. A regression model is put simply some function that can accurately calculate y from that input X. Different types of regression will be more suited to different tasks and in order to evaluate which approach is most suited to a given problem we use performance metrics such as Mean Squared Error (MSE) and Root Mean Squared Error (RMSE).

The process of finding a function that can accurately describe the relationship between the inputs and outputs is the 'learning' part of 'machine learning'. Most implementations use a technique called gradient descent to find the function that will have the lowest 'loss' values, a value similar to the performance metric that tells the learning algorithm how well the regression model is performing. Each type of regression will have different parameters it can tweak to try and find a better representation of the input-output relationship. A simple linear regression model which represents the relationship as

$$y = mX + c$$

can only change the m and c values, meaning it cannot accurately represent a relationship more complex than a simple straight line graph. Luckily there are more complex models that allow us to model trends with inputs of high dimensions and relationships that are non-linear. For example, we discussed in our related works (Section 1.2) an ensemble method called extreme gradient boosting approach (XGBoost) to solar radiation prediction (4), and we can also combine different pre and post processing techniques into a pipeline to improve our prediction accuracy even further.

Feature Engineering

Before beginning to design the model we needed to ensure our data set was preprocessed so that there were no outlying data points that might cause errors. Making use of the python library Pandas, we imported our data set into a data frame to allow for more powerful data manipulation tools. The first area requiring attention was the 'WindGust' feature, we found that the feature contained less entries than

every other as we found that in some instances the API would return an empty response if there was a 0mph gust recorded if not set up correctly. This would have caused issues with the sckit-learn library we used for our regression so we made sure that empty entries were replaced with 0 where needed, and also removed any remaining entries containing empty variables afterwards.

Next we noticed some inexplicable values for the load factor where it was above 100% suggesting it had managed to generate more than its maximum capacity. We examined these cases to ensure we edit our data collection method to avoid these anomalies. We found it was a small subset of locations causing these issues which was likely due to a misreporting of maximum generation capacity causing issues in our calculation of the load factor. Therefore, we decided the best course of action was to remove the locations from consideration this could cause issues in the training of the model if left unresolved.

Next one of the important steps of preprocessing, is to normalise the features so every feature is scaled to a $[0, 1]$ range which is important for the gradient descent algorithm.

Constructing our model

Our initial approach to choosing our model was to compare the Mean Squared Error of common regression model types, from simple linear models to more complex ensemble approaches. This would allow us to compare the performance of each on our data set and see which we should evaluate further. We made use of a few open source libraries for this, scikit-learn (14) providing a large selection of different models, and we also made use of the popular XGBoost library (15). This gave us a wide range of options to test.
Our evaluation process was as follows:

1. Import Data Set
2. Preprocess & Standarise features
3. Initialise Model with default settings
4. Perform 5-fold cross validation on the data set
5. Use scikit-learn.metrics.mean_square_error to evaluate performance

From these initial results we were able to see that our best performing models were the two gradient boosting approaches, the MLP

Table 3.1: Linear Model Performance

Model Name	Average MSE Over 5 Folds (4 d.p)
Linear	0.0518
Lasso	0.0518
ElasticNet	0.0823
SGDRegressor	0.0519
SVM	0.0493

Table 3.2: Non-Linear Model Performance

Model Name	Average MSE Over 5 Folds (4 d.p)
Decision Tree Regressor	0.0792
KNN-Regressor	0.0512
GradBoost Regressor	0.0485
AdaBoost Regressor	0.0534
MLP Regressor	0.0489
XGBoost	0.0495

regressor and SVM all achieveing a mean squared error of less than 0.05. As these were all tested with default hyperparameters our next step was to tune these hyperparameters to improve the accuracy. To do this we initially used a simple grid search on a set of potential parameters. We began with the GradBoost Regressor as it was the most promising from our initial results. The models have a range of options to configure, we decided to do a grid search across a set of learning rates, max depths and different loss and criterion functions. Doing this we found that by increasing the maximum search depth and using the 'huber' loss function instead of the default 'squared_error' function allowed for improved performance. These new parameters did not make a massive leap in performance, however they did improve upon the default parameters so we adopted them until we found a better option.

The next model we looked at was the MLP regressor. This model had a lot more flexibility in its configuration and if it showed potential to out perform the others could open the door for us to examine designing a more complex neural network structure using TensorFlow or pyTorch. Within the MLP Regressor model we can describe the layer structure, learning rate, activation functions and regularisation factors. We designed a range of structures to see if applying an auto-encoder like structure to the data set would be able to discover any latent features in the data that may improve the models ability to find trends. Testing a range of different structures for the MLPRegressor however did not uncover a new direction to take the model. Instead we

Table 3.3: MLPRegressor Shape Performance

Hidden Layer Shape	Average MSE Over 5 Folds (4 d.p)	ScikitLearn Score
(25,)	0.0489	0.4023
(50,)	0.0478	0.4227
(100,)	0.0469	0.4262
(50,100)	0.0499	0.4584
(25,50,100)	0.0519	0.4101
(50, 100, 250)	0.0577	0.3696
(100, 50, 100)	0.0534	0.3792
(100,25,25,100)	0.0540	0.3775
(100, 50, 25, 50, 100)	0.0569	0.3850
(25, 50, 100, 50, 25)	0.0569	0.2805

found that a standard single hidden layer structure outperformed any more complex structures we tested. The auto encoder structures we mentioned could hold some promise also performed worse than a standard layout and some structures performed worse than default models we had examined previously. Evaluating these results against the improved scored we found when tuning the hyper-parameters of the gradient boosting approach we decided that the increased complexity of MLPRegressors did not hold a significant performance benefit over the gradient boosting approaches we had found and decided to continue tuning the gradient boosting approach.

While investigating meta-learning techniques to improve our model performance, we discovered a library called 'TPOT' (16). This is an autoML algorithm that makes use of genetic algorithms to search for an optimal configuration and pipeline to use our model in. Up to this point we had just been using a simple scaling preprocessing step to normalise the features, but by making use of the TPOT library we can find a more complex pipeline. After running the search for 100 generations, our best pipeline consisted of a feature selection step using AdaBoost and ExtraTrees Regressors before feeding the best results into a combination of GradBoostRegressors, SVM Regressors and KNN Regressor models. Each of these have performed well individually, so the combination of the into an ensemble prediction makes sense that together they would continue to perform well. With all these steps combined, we observed a much more accurate prediction and by combining the results of multiple models we expect to observe less overfitting in the final model.

3.2.3 Allocation Optimisation

With our prediction model designed and trained we could move onto designing an algorithm to find optimal allocations. We will be using the model designed in the previous section to determine the performance of a given allocation by using the model to make predictions of the total outputs of every farm included in the allocation. Using these predictions we can evaluate the performance of any given allocation. The problem then becomes how do we optimally traverse this large search space in an efficient manner.

Multi-objective optimisation

The unique challenge of this problem is that there are a few different objectives we want to achieve. Obviously the main objective of this project is to find an allocation that generates the most electricity, but how we define that is not straight-forward. As we have observed in training our model the energy generated from a location is very variable due to the inconsistency of weather. Therefore, while we want to maximise the average generation amount, we also want to minimise the amount of variation as if our solution cannot consistently contribute electricity to the grid it is losing out to non-renewable sources which can.

One approach to using multiple objectives as a heuristic in our search method, would be to use a simple weighted sum.

$$f(obj_1, obj_2, ..., obj_n) = w_1 obj_1 + w_2 obj_2 + ... + w_n obj_n$$

This would allow us to choose how important each objective is by changing the values of the corresponding weight value w_i. However this raises the challenge of how to you choose the importance of each objective, this is a common issue found in many multi-objective resource allocation problems (MORAPs). This problem has a large assortment of papers addressing possible approaches, with one of the most common being variations on evolutionary or genetic algorithms.

Genetic Algorithm Approach

Genetic algorithms, are not deterministic meaning they do not guarantee us the global optimal solution. However, as described before the search space for this problem grows exponentially meaning there is no efficient way to find a deterministic global optimal solution. The benefit of heuristic driven approaches like a genetic algorithm is that given a

long enough run time they will find a local minimum and by including random restart factors such as mutation we increase the possibility of exploring more of the search space. This approach will not be a fast real time implementation, instead the longer we let it run theoretically the better the result should be. This will make it a flexible for different applications allowing for quicker estimates or long term searches for more certain results. We believe this is not a problem in the use case of infrastructure planning as the planning process is a time consuming one where this algorithm can be running in parallel of other tasks before being returning its results.

The benefits of a genetic algorithm for our allocation problem, is that the search method is based up ranking a large set of solutions rather than one allocation at a time moving towards a maximum reward value. This opens up the possibility of more complex ranking functions to allow the algorithm find a "pareto front" of solutions which are all viewed as equally good by balancing the importance of different objectives. This means that rather than the user defining importance at the beginning of run time, they are presented at the end a selection of allocations which have different priorities. This approach is well suited to our problem as it will allow us to make choices between solutions that maximise average output or minimise variance, or any other metrics we decide would be relevant to the task.

NGSA-II

A very proven approach to this multi-objective optimisation problem is NGSA-II, an algorithm proposed in the 2002 paper "A fast and elitist multiobjective genetic algorithm: NSGA-II" (17). This approach has been expanded and improved in many follow up papers such as Carvalho and Araujo's 2009 paper proposing improvements to the algorithm using an adaptive mutation rate that can achieve better diversity in its solutions (18). As such we believe it is a good groundwork to base our initial investigations on, and if this genetic approach proves itself in this report performance improvements from more cutting edge papers in future works.

The algorithm works by making use of two ranking metrics to select the best candidates from each generation: non-dominated sort; and crowding distance assignment. Non-dominated sort groups all the candidate locations into "fronts", where the best solutions are in the first front and each front is worse than the one before it as can be seen in Figure

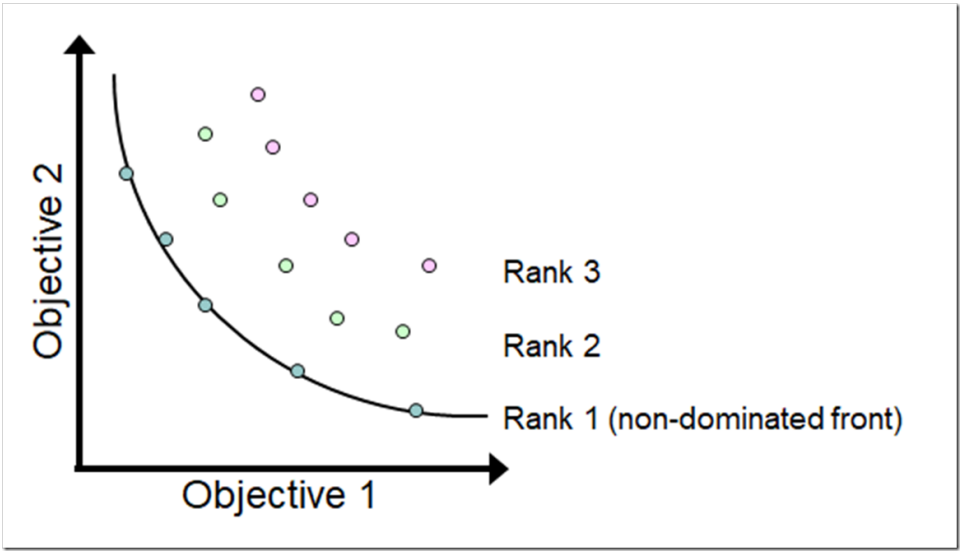

Figure 3.1: Multi-Objective Front visualisation.(19)

3.1. These fronts are determined by a comparison called "domination". A point p dominates another point q if it meets the following conditions (where F is the set of objective functions):

$$\forall f_i \in F : f_i(p) \leq f_i(q)$$

The non-dominated sort ranks points based on the number of other points they are dominated by. The best front consists of all points not dominated by anything else, ensuring it consists of the best solutions found so far. As such these are then highly desirable in the next generation and should be ranked highly for selection. As this ranking only ranks by group it does not provide any ranking between members of the same front. This is where crowding distance is used. The concept behind this metric is that to reduce the algorithm converging to one points and encourage it to explore, the more isolated a point is the higher it should be ranked.

The algorithm uses these rankings to select which members of the population are carried on to the next generation, but it also includes the offspring of the generation in this ranking. Figure 3.2 shows the selection of a generation P_{t+1} from the combination of generation P_t and it's offspring Q_t. These are combined into R_t which we then apply Non-dominated sort to to find which fronts are rejected and which need to be split by crowding distance sorting. Using the fast sorting algorithms defined in the paper we can apply this process in $O(MN^2)$ where M is the number of objectives and N is the population size.

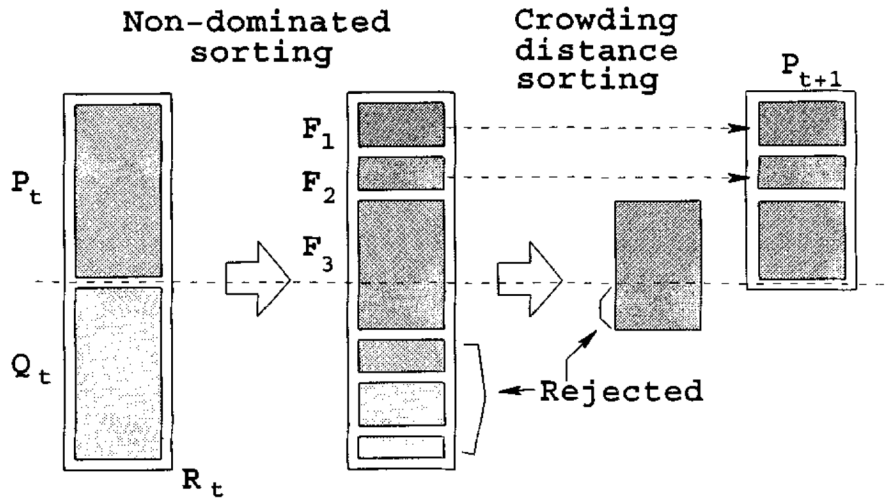

Figure 3.2: NGSA-II Selection Process.(17)

Problem Encoding & Decoding

A genetic algorithm works by simulating the "evolution" of structure called a chromosome. These chromosomes are using represented as a list of often binary values called genes, where each value refers to some aspect of the problem. The first step of applying a genetic algorithm to a problem is to determine a method of encoding a potential solution into a chromosome format.

$$C = \{g_1, g_2, ..., g_n\}$$

Our problem fortunately has quite an intuitive encoding, where we have chromosomes of length n where n is the number of candidate locations, and each gene in the chromosome takes an integer value in range $[0, B]$ where B is max budget and these gene values each correspond to the number of "units" being allocated to that location.

This representation makes our results easily human readable, and decoding an allocation to the real locations chosen only requires us to define a constant order of locations. This allows us to use a dictionary to lookup information about the a gene's corresponding location using the index number.

To start the genetic algorithm we need to initialise a population of chromosomes, we have a few options: sample random distribution; set

all locations to 1; or use a human decided pattern of locations. This is quite an important decision as the starting point of the algorithm will determine where in the search space it is able to investigate, as such we will likely want to run it at multiple starting points to increase the breadth of our search. As such we decided on initialising our population with a random distribution, having a human picked starting distribution could allow for insight into specific areas but due to lack of knowledge in the field any human suggestions we would feed the algorithm would likely be as insightful as random generation. We made use of the dirichlet distribution to generate a set of fractional values that sum to 1.0 representing the proportion of the budget each location would get.

Fitness Function

In order to evaluate the rankings of each generation set out in the NGSA-II paper, we need to define our set of objective/fitness functions. As we discussed at the start of this section the a successful allocation should balance the mean of the predicted generation amounts and the variance in the predicted generation amounts. This is where we make use of the model described in section 3.2.2. For each of our candidate locations we modelled the expected weather patterns based on daily samples over a two year period (again sampled from VisualCrossing.com), from this data we evaluated the mean and standard deviation. Using these two data points for each location we could sample a Gaussian distribution for 1000 "expected weather days" which should represent a range of scenarios the location would be subject to. We next applied our trained model to these "expected weather" samples to generate a large set of generation predictions. By storing these predictions, we were able to design our fitness functions to lookup the predictions for each location and use them to evaluate the performance a given allocation.

When evaluating an allocations performance we decided on using 4 objective values: Mean Predicted Output; Negative Variance of Predicted Outputs; Minimum Predicted Output and Maximum Predicted Output. As most of our objectives are based around maximising values, we are considering this a maximisation problem hence the use of negative variance as variance is the sole objective we want to minimise. To evaluate these values, we take an input of an allocation and loop over its entries. If the allocation has assigned a wind turbine to that location (value is non 0) we lookup the predictions for that location and add the values to a predictions list. To account for multiple wind

turbines being allocated we add the predictions once for each turbine allocated to that location. This process is shown in algorithm 1.

Algorithm 1 Objective Value Calculation

$A \leftarrow$ Allocation Input
$P \leftarrow \emptyset$
for $l \in A$ **do**
  if $val(l) > 0$ **then**
    for $i \in [1, val(l)]$ **do**
      $P \leftarrow P \cup \{PredictLookup(location(x))\}$
    end for
  end if
end for
return $[P.mean(), P.var(), P.min(), P.max()]$

One further consideration we had to make in this function was how to handle illegal allocations. It is possible due to the crossover, for some of the offspring of a generation to have allocated more units than the budget allows. Rather than attempting to retroactively fit these offspring to the budget, we instead give them a zero value for every objective such that they will never be introduced into the next generation. This works for all cases except where you have a value you want to minimise such as variance in our case as a 0 value will outperform all other allocations - meaning illegal allocations are viewed as the best. This can be fixed simply by ensuring that in illegal or erroneous cases you set the minimised objective to -1 instead of 0.

Reproduction

The selection process defined by NGSA-II requires that we produce an offspring generation before applying the rankings, and we use a tournament selection approach to select the half the population to be parent chromosomes. After selecting the parents we produce two offspring per parent pair made up of some of each parents genes in a process called crossover. We define a hyperparmeter $c \in [0, 1]$ called crossover rate determining the split of each parent in the two offspring. For example if $c = 0.2$ one offspring would consist of 20% of parent 1's genes and 80% of parent 2's genes, and the other offspring would inherit the opposite proportions to ensure equal representation of parents in the next generation.

The tournament selection process (seen in algorithm 2) is a way of mak-

Algorithm 2 Tournament Selection Process

```
P ← Population
S ← ∅
while |S| < |P| do
  s1 ←Random Member of P
  s2 ←Random Member of P
  if domRank[s1] < domRank[s2] then
    S ← S ∪ {s1}
  else if domRank[s1] = domRank[s2] then
    if crowdDist[s1] < crowdDist[s2] then
      S ← S ∪ {s1}
    else
      S ← S ∪ {s2}
    end if
  else
    S ← S ∪ {s2}
  end if
end while
return S
```

ing random selections from a population weighted by performance of the chromosome in some function. By introducing a random aspect to the selection process we allow non-optimal solutions to be occasionally picked and introduce a random walk possibility to explore and reduce the risk of getting stuck in a local-optimal solution. This stochastic element is further introduced in the mutation stage of our reproduction.

Mutation

Mutation is an important part of genetic algorithms, it increases the chances of finding a global optima over a long search time by allowing random jumps steps away from local optima. It works after the reproduction stage where each member of the new population is subjected to a chance to mutate. We set the chance of mutation in a hyper-parameter $m \in [0, 1]$, and we found that the best balance was setting it to 0.1 (10% chance to mutate) so that in a size 100 population means we expect 10 allocations to mutate and diverge from the crowd. This means that is the results start to converge to a local maximum, 10 of the generation mutating each generation gives ample opportunity for a random occurrence to find a better result, while not hindering the improvements found generation to generation by randomly restarting a large amount of the population. This is an area that likely could be improved in future implementations by using adaptive mutation rates

to increase mutation when the generational improvements reduce and results start to converge.

Traditionally genetic algorithms encode there problems in bit strings, this allows for mutation to occur at a gene level and a commonly used approach is to apply a bit flip to mutated genes. This simple approach is very fast to apply and in a well thought out encoding can make a large impact in the results allowing for good discovery of new solutions. However in our encoding we are using integer values which cannot be mutated by a simple bit flip. As such we looked for a different solution and came across a couple of alternative approaches. One option was to swap values between genes, however we decided this was not appropriate for our approach due to the likely sparsity of the allocations over a large number of candidate locations. As we have a relatively low mutation rate we want our mutations to make discover new solutions and if we are swapping gene values it is likely a majority of mutations will be wasted on swapping genes set to 0.

This led us to consider another approach which is to simply regenerate a random allocation. With such a large search space a full random restart would likely allow us to cover a larger area that is very unlikely to overlap with our currently searched path. We made us of the same distribution discussed in section 3.2.3 to regenerate our mutated chromosomes. This is also quite a quick operation to perform as it consists of sampling a random distribution, and has a large impact making it a great option for our implementation.

Our Search Approach

With our algorithm defined we could start searching for an optimal solution to our problem. As we repeatedly mention, the search space is very large and our algorithm has a range of hyper-parameters to set all of which will affect the solutions found.

When setting up our approach we have 4 hyper-parameters we can set to define the run time operation: generation size; number of generations; mutation rate and crossover rate. The first two will affect the run time of the search and by setting them to large values are more likely to find us an optimal result. The second two parameters affect the generation to generation reproduction process, meaning we can tweak them to change how much variation is introduced generation to generation. The majority of searches we ran used the following configuration: Generation Count = 100, Generation Size = 150, Mu-

tation rate = 0.1, Crossover rate = 0.65. We found this was produced a productive search that often tailed in improvements by the 150th generation while still introducing varied allocations each generation to potentially find improvements.

One problem with this approach is results are often dependent on the initial starting population, as this sets up where the algorithm is able to go. Therefore, we have a couple of options. We can run an extremely long search period with a high mutation rate and large population size to ensure that one search travels as much of the space as possible, or we can run smaller searches repeatedly on a large range of initial seeds cover more ground that way. We have opted to take the second approach and as such will be searching over a set of seed values that will initialise the algorithm in different locations. This does introduce an uncertainty to our search results and we cannot claim they are the best possible as there may be unconsidered options that far outperform what we have seen. We will be considering a selection of potential improvements to this method in our future works section as we have explored some ideas but have not had the time to fully investigate the improvements that some approaches may provide.

Chapter 4

Evaluation

4.1 Predictor Model

4.1.1 Data Set

At the start of this project we set some objectives to measure our solutions effectiveness against what we expected to be achievable. Our first objective was to ensure that we had a sufficiently large data set to work with. We aimed to have more than 5000 entries in our data set and we achieved this even after some of the preprocessing steps to remove erroneous data points with currently 6908 data entries being fed into the training step of our model. This goal was set before we started to modelling process and after growing the data set throughout we have seen that we may benefit from a bigger target amount of data entries. Intuitively with a machine learning task, the more data the better as long as appropriate measures are taken to include regularisation factors to stop overfitting.

We can simulate the improvements given by data set size, by testing the model on a fixed test set size but a varying train set size using the train_test_split function we use in the process of training our model. As we can see in figure 4.1 by two metrics, the mean squared error and scikit_learn "score" function (a perfect model would score 1, and a random model with no correlation would score -1), the model improves as the training set size increases. Although we cannot confidently extrapolate these trends to find a training set size that would be sufficient to meet target accuracy and scores as we do not know the point at which this will tail off, it is safe to say here our data set is not sufficient to reach the full potential of our model.

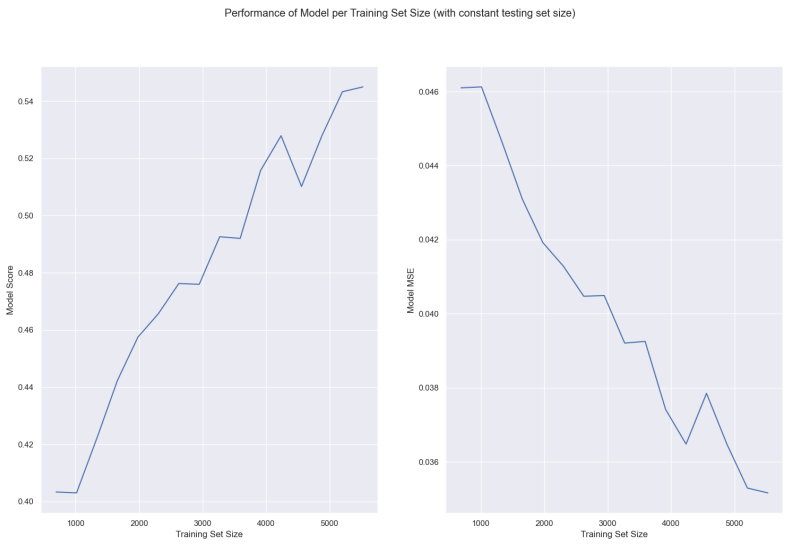

Figure 4.1: Model performance per training set size.

4.1.2 Model Performance

Our second objective was our that using a sufficiently large test set we could achieve a mean squared error of less than 0.025. Unfortunately this was not something were able to achieve, and while we found that increasing the ratio of training to testing set size did yield performance improvements we were not confident that the size of the training set would allow us to be confident in the performance estimates it provided. This is more evidence to suggest that we needed a larger training set as we outlined previously. We were only able to achieve a mean squared error of 0.0419 when using a 60-40 train-test split, however after considering what an acceptable size of testing set would be we settled on an 80-20 split which netted us an error of 0.0373. We believed this was an acceptable compromise as it ensured we were using as much data as possible to train the model on while still retaining over 1000 entries to evaluate it upon. The reason we view the testing set size as important is that there are a lot variation in the weather in the data set, therefore we want to ensure we test, record and evaluate the performance on as many of these as possible to increase confidence in our results.

As discussed in the prior section, the data set appears to be the fairly limiting factor in our model. Because of this despite not meeting our original objectives, we are happy with the results and the model we produced. While the accuracy leaves much room for improvement, as

long as we take these errors into consideration when evaluating the results of the allocation algorithm that uses the predictions it is a sufficient model for this initial investigation. As we will discuss later, due to restrictions in time for this project there are some areas where we have not been able to reach the end result we would like and as such we will be proposing a range of improvements to the project to be looked at in future work.

The last objective we defined in relation to the prediction model was its prediction speed. Here we can clearly say we have met this objective by benchmarking our the prediction function using the python time library. We time the time taken to make a prediction for a large set of values, in order to find a mean as at a small timescale there will likely variations. Looking at table 4.1.2 we can see that across a large range prediction input sizes, the time per prediction doesn't vary a large amount and always stays below 0.5 ms, which is a factor of 10 below the objective time of 5 ms.

Mean Individual Prediction Time (ms)	Size of Prediction Input
0.086	690
0.093	1013
0.116	1335
0.109	1657
0.139	1980
0.104	2302
0.097	2625
0.102	2947
0.117	3269
0.148	3592
0.147	3914
0.149	4236
0.115	4559
0.357	4881
0.240	5204
0.188	5526

Table 4.1: Speed of individual predictions at different input volumes

4.2 Allocation Optimiser

As with the predictor model we will be evaluating the allocation aspect of this project against the objectives we set out at the start of

the project. However in those comparisons we have to make sure we consider any error carried through from the predictor model as it is what we base our fitness values on.

4.2.1 Allocation Performance

The objective we set out to measure performance was that we should aim to achieve a better predicted load factor than the average of wind turbines across the UK currently. As we are using mean predicted load factor as one of our objective values this is simple to look at our catalogue of result allocations and there corresponding fitness values. We found that across 710 allocations that our algorithm had found to be optimal (allocations in the non-dominating front of the final generation of a search) the lowest predicted load factor was 27.79% and that the average load factor was 29.32% with the best performing allocation achieving a load factor of 30.82%.

These values however need to be considered in context, as on the surface it appears that these results show that we can consistently improve upon the performance of existing infrastructure by a 3-4% margin (equating to a 10% performance improvement). In reality we cannot make these conclusions confidently as we have seen previously that the performance of our predictions are quite variable and the average absolute error is around 13%. As such we cannot confidently say if our results are improvements from just looking at this metric. We can however define an allocation in reference to the current proportion of capacity at each location. By running this allocation through the same fitness function as our other results use we can compare the performance of our algorithm on a level playing field.

To do this we calculated the lowest common multiple of the capacity of all locations as lcm and recorded $\lfloor \frac{lcm}{capcity} \rfloor$ for each location. Due to the large number of locations these values were extremely large causing our fitness function to have issues computing the results, so we scaled each of these values down by a factor of 10×10^{18} so that the lowest allocation was 1. This gave us an allocation that approximates the distribution of wind turbine capacity across the UK today and can be seen in figure 4.2. This allocation gave us a predicted mean load factor of 27.18%. This leads us to two conclusions: our approach, while not as accurate as we would like, models current UK wind power infrastructure within 1% of the actual recorded load factor; and our genetic algorithm approach does discover allocations of wind resources that consistently more efficient than the current system with improvements

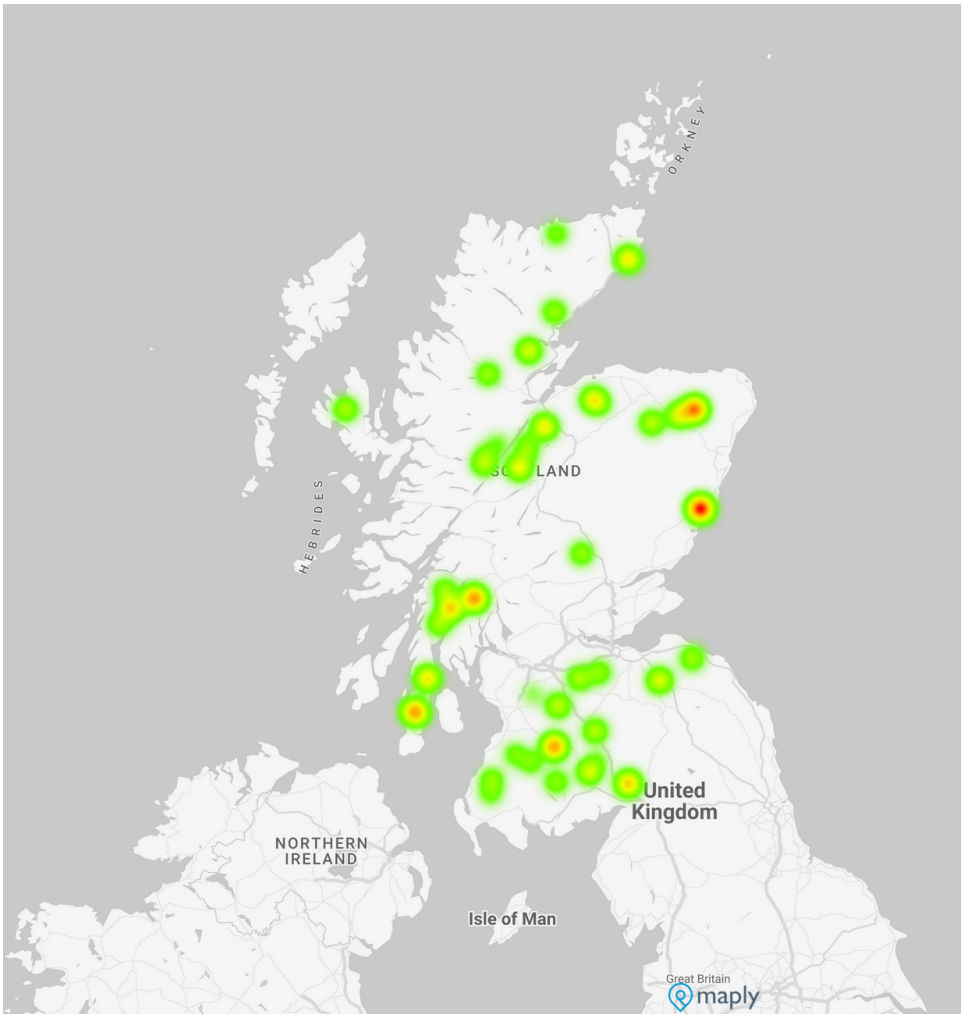

Figure 4.2: Our modelled distribution of existing wind turbines

of up to 13% over the existing system.

In our objectives we only set our an objective for the performance relative to mean load factor, however the benefit of our approach is the range of solutions we get in regards to balancing variance and maximum and minimum predicted values to choose an allocation that allows for consistent generation. Using the strategy we define above we will compare our selection of results to existing systems results to choose a result that performs most consistently while still on average creating high outputs. We found that the most all-around performing results achieved a mean load factor of 30.17% and were at least equal to the existing system in every other metric: a variance of 0.00993 compared

to the existing 0.0122, the same maximum existing load factor (likely a limitation of our fitness function as we found many results with identical maximum load factors) of 50.33% and a higher minimum predicted load factor of 5.72%. This result achieves the exact balance we set out to achieve in this project and examining our results there are many like it coming from different starting seeds.

The main thing we notice throughout our results is how sparse the resulting allocations are. We ran the algorithm examining all 56 candidate locations shown in figure 2.2 and the majority of our results ended up only allocating to 10 - 20 of these locations. This suggests to us that while a large range of locations is necessary due to space constraints of each site, it is not required to create a consistent generation amount. This is potentially due to the set of locations chosen not being large enough to have large variation in weather conditions over an extended time period. We may have seen different results if the candidate locations were sampled from more southern regions of the UK where weather trends are different. We also see that although supplied with no geographical knowledge in its decision making process our algorithm has generally shown preference to locations closer to the sea, which likely are subject to less shielding from winds. This may be coincidence in the allocations that we have visualised as the locations chosen in each allocation do vary, and there is likely selection bias introduced by the fact that we are using existing wind turbines and choices made in their location would likely take these factors into account. However is we look at the figure 4.3 we can see a "V" shape of choices even though there are candidate locations in the more land locked areas they avoid. If we applied this approach to a more landlocked area it would be interesting to see if these trends continue and the algorithm continues to prefer locations closer to the coast. If we compare this to figure 4.2 we can see the areas of maximum capacity in the current system do not correspond with the optimal allocations found by our approach, in fact our one concentrated spot in figure 4.3 is much less represented in the real world currently suggesting this is a potential area for new investment.

4.2.2 Algorithm design

The other two objectives we detailed in this report were related to the design of the algorithm itself, and certain behaviours it should possess. We can easily confirm we have met our third objective here, as we have described our implementation has 4 hyper-parameters and that generation size m and generation count n will impact the run time. We can

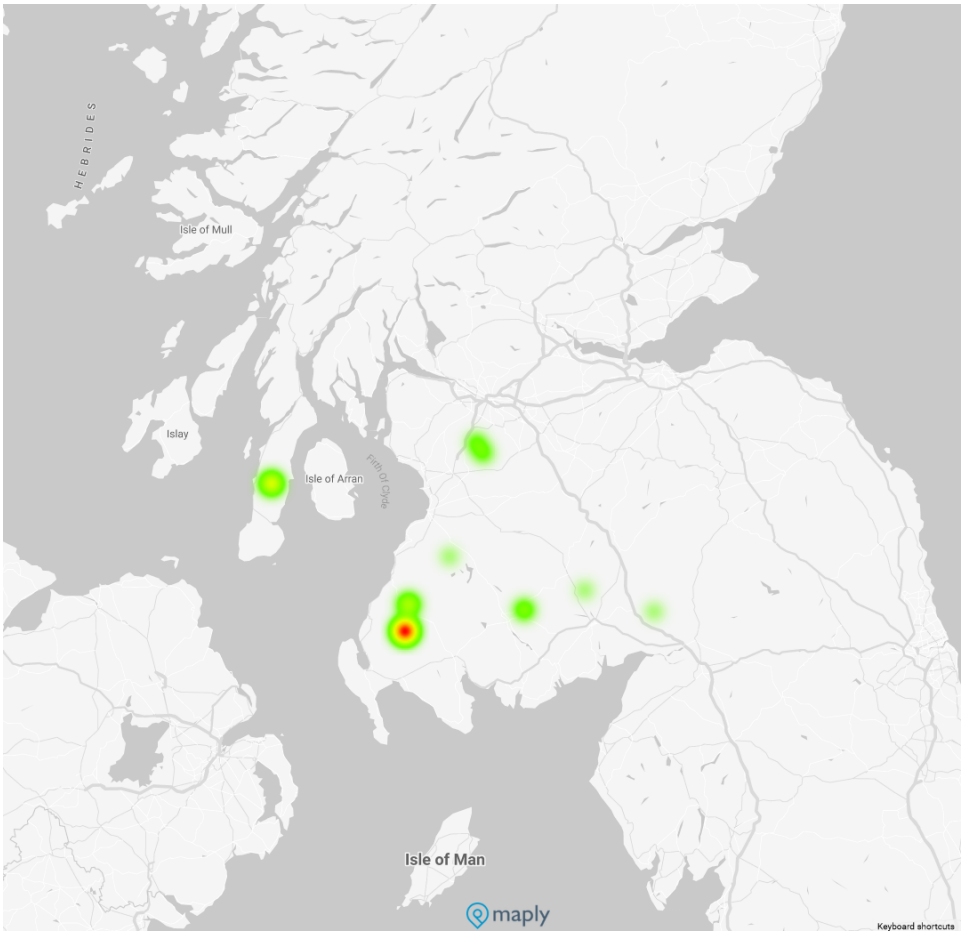

Figure 4.3: Heat map of the allocation described.

also say intuitively that by reducing the generation count and size we will decrease the run time as it simply reduces the number of iterations of we have to carry out. In a simplified view, the number of times the algorithm has to evaluate the performance of an allocation is $O(nm)$.

As to whether upon termination the best result is returned, due to the design of the NGSA-II algorithm the population of the next generation is selected by ranking the top half of both the offspring and the original population (as seen in figure 4.3). This means the top ranking allocations of any generation must be at least as good as the generation preceding them. This means that at any point of termination, by the elitist property of the NGSA-II algorithm the best performing allocation throughout the search space will still be present. This objective has to be considered with some flexibility due to the multi-objective design of our problem meaning we do not have a strictly best performing result but our choice of algorithm to tackle this approach does ensure that we preserve all best performing allocations throughout the search time.

4.3 Project Management

At the beginning of this project we proposed a gantt chart that we would follow to keep us on track and give us consistent milestones to aim to meet (see figure 4.4). For the first half of the project we managed to follow this well each week focusing on a different area of research to build our understanding of how the project would work. However, this planning style made many assumptions about the process of completing this project that did not hold up after the initial research stage. Up to the completion of the progress report, we had managed to stick to the time scale however after that point we had planned to specify a set of tests. This however proved challenging and made us realise that we did not know how the end result would look as a large part of the development process would be made up of testing different approaches to find the best one which would not suit the strict timeline of defining tests and developing around those tests we had set out. Therefore from this point we reevaluated our plan to set a maximum time to spend on each core section: data set creation; prediction model; and allocation optimiser. With this new more loose plan in place we were able to spend as much time as possible focusing on exactly what would lead us to our best results by splitting the time between research, implementation, and testing as needed.

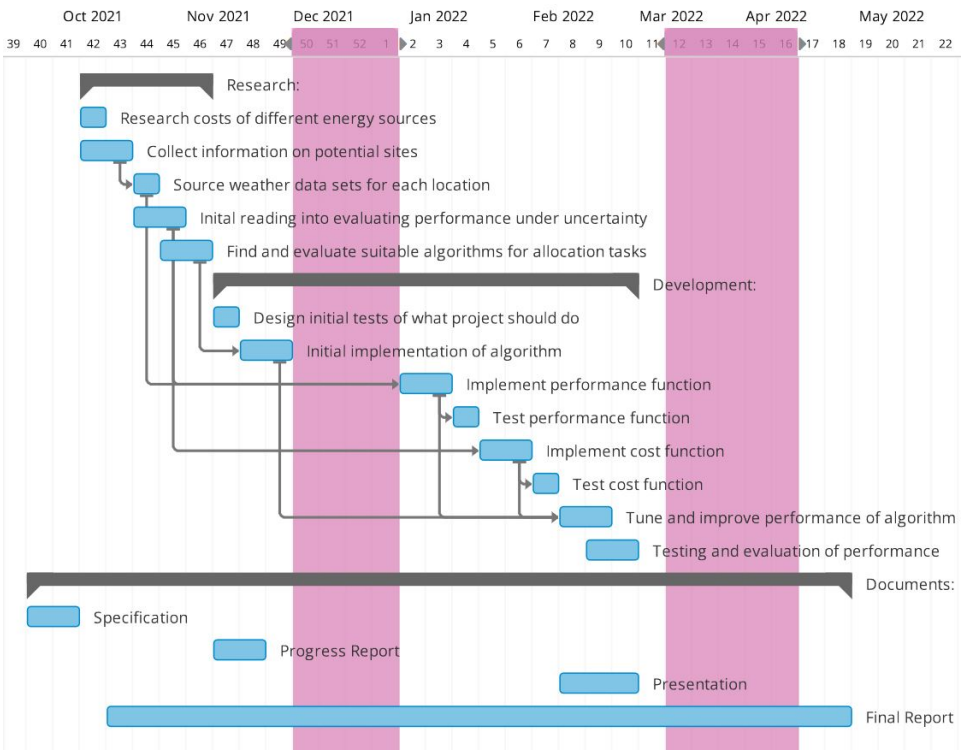

Figure 4.4: Initially proposed Gantt Chart

While this approach did not follow our initial proposed method of managing the project it let us be more flexible with the development of the project allowing us to learn what worked best as the project went on. We believe this change in approach allowed us to be more successful in our development, and was more appropriate for a single person task that only had one strict deadline. Our failure to follow our initial plan was a symptom of issues with the plan itself. As while we left flexibility in timelines to account for overrunning, it was made with a linear development path in mind rather than the cyclical research and trial/error based approach that suited the task better. In future a better planning approach would have been to split the task into the three core focus areas and plan each of them out in detail, accounting for time to research, implement and test, multiple times before arriving at a final result.

Chapter 5

Future Work

Throughout this report we have discussed a variety of limitations or areas of improvement that have not come to fruition. In this section we will outline some of the areas that we believe this project could be improved with further work and areas of interest that we have discovered from our implementation that have not been explored in research much and we would like to see further work done.

5.1 Performance Improvements

When we initially discussed the use of the NGSA-II algorithm one of the strengths we saw was the fact that it had a large assortment of follow up papers detailing performance improvements that we could capitalise upon if the base algorithm showed promise. In this report we have shown the promise of this approach in the use case, and as such we would be interested to see if this performance could be furthered if a more cutting edge approach to multi-objective optimisation approach was implemented. We have seen papers highlighting more cutting edge approaches for example: modifying NGSA-II with an adaptive mutation rate (18); a more cutting edge extremal optimisation (EO) approach with polynomial mutation (20); and many more approaches to multi-objective optimisation that are not evolution based such as swarm and animal based algorithms ((21), (22)).

5.2 Generalisation

One potential limitation of this report is the focus it has on the UK or more specifically Scotland as an example location. While subject to varied weather conditions, our training data set formed across our candidate locations is specific to weather in this country and others with

similar climates. One strength of the approach described in this report is its ability to be generalised fully if the prediction model is trained suitably. Therefore we believe that a part of expanding the training data set that we discussed in section 4.1.1 we believe there is room for to follow on from this report with a more global data set leading to a more generalised model that should allow this technique to be applied almost anywhere where there exists records of past weather. We believe that with a generalised prediction model this approach could lead to an improvement in the roll out of renewable energy globally, potentially improving adoption in developing areas and allowing for greater efficiency in large investments of existing large adopters of renewable technology.

Another area that this approach could be more generalised is the type of energy considered. In this paper we have very strictly focus on just on-shore wind power, however there are many places where solar energy is a more viable option and having a method of making location decisions for this could be useful. This would likely require a reworking of the prediction model entirely as it is unlikely that the same factors affecting wind power would hold much relevance in regards to solar energy.

5.3 Trend Extrapolation

The climate crisis is what initially brought attention to the importance of this topic and is highlighting the importance of renewable energy. However, one thing that this report does not consider is the long term weather trends associated with climate change and how these changes may mean that choices made today on past data is not optimal in 10 years from now. We believe that an investigation on how these trends can be incorporated into the objective values could yield an insight into investments policies that stand to be consistent over a multi-year time frame.

5.4 Genetic Algorithm Improvements

One of the limitations of genetic algorithms and heuristic approaches is that the best solution the algorithm finds in a relatively short time span is dependent on its starting state. As such a method of predetermining the qualities of a good starting location would immensely improve the ability to find globally optimal solutions. The approach used in this paper is to run searches on as large of a set of starting states as possible,

to get a wide range of solutions to improve confidence in our best results being the best possible. This approach gives us no guarantees however, and in fact we believe there are likely a large number of solutions we did not find that could be superior to our results. We theorised a couple of improved approaches to this which we propose here.

5.4.1 Approximate Algorithm Approach

First we considered a running a brute force search across all starting locations on simplified view of the problem, possibly with a very restricted budget or with locations clustered based on similarity. Whatever this simplified problem statement, we could use these to construct starting starting states that are applicable to the full problem under the assumption that they are will produce us the optimal solutions in the full program. This would be similar to the concept of linear & integer programming in the study of approximate algorithms, where an optimisation problem is defined with relaxed constraints and the solution is interpreted into an approximate solution to the true problem.

5.4.2 Deep Learning Guided Approach

Another concept we briefly considered was the possibility of was making use of a deep neural network to make predictions on the best initial states to start on. From a brief exploration of existing literature on this area, we found that this is not a concept that has been explored. The one paper we found exploring this idea was a 2021 paper from the NeurIPS conference which detailed the use of using a deep reinforcement learning agent to aide the seeding of it's genetic algorithm population (23). They claim this combined approach is novel and by combining genetic programming and reinforcement learning approaches outperforms existing approaches. The problem they tackle is a very different domain to our problem, however it gives a basis for our claims that a similar approach work in our problem space. We only briefly considered this approach due to time constraints in this project, however we strongly believe this could be a very interesting area of future research for many types of evolutionary algorithms.

Chapter 6

Conclusion

In this project we aimed to find a method to streamline the process of finding efficient allocations of renewable energy across the UK, and over the course of this report we have showed the development of and thought process behind our solution. The end result focused on a narrower scope of renewable energy than we initially planned, however this narrow scope allowed us to design a specialised system that consistently finds results that outperform the average output of current UK wind infrastructure.

Overall this project has been successful in producing a new approach to the an important problem in renewable energy. This is not to say however, that there are no short comings of this report. While we have concluded that our results do generally show an out performance of the existing infrastructure by evaluating both our results and the existing distribution of wind turbines under the same fitness function, the error rate of the prediction model powering this fitness function still has a lot of room for improvement. Therefore, we believe a future investigation should be conducted with a more accurate prediction model before any of our performance claims should be properly considered.

With these drawbacks considered, we believe that if our performance claims can be further supported this is an extremely important result. The opportunity to see up to an 11% increase over the existing generation amounts with future installations is incredibly exciting as increasing the value proposition of renewable energies would like draw more much needed investment interest into the area.

Chapter 7

Author's Assessment of the Project

This project does not show any breakthrough in techniques and technologies, instead it applies existing well supported approaches to the increasingly important area of renewable energy and proposes a new application of these techniques. As shown in the related works section of this report both regression techniques and genetic algorithms have been used in the renewable energy field in select works however being used together to model a larger scale problem like grid layout planning is not something we saw evidence of being done before.

This contribution is important to the field of computer science as it highlights a range of future works that could lead to new techniques in genetic algorithms, while showing that existing algorithms can still be applied to new fields. Most importantly the results of this report show that these techniques are viable for "social good" problems and that theoretical computer science can be applied in many ways to propose improvements to the real world.

The hope is that the work of this project is applicable to a more generalised version of the problem we presented. I believe this sets the groundwork for a more thorough implementation of our work to be made use of to help guide the decision-making process of future renewable energy infrastructure as we have shown our algorithm can find allocations that improve upon what exists. I also hope that the ideas set out in our future works discussion can lead to more theoretical improvements in genetic algorithms as we believe with improvements genetic algorithms could be applicable to even more areas.

The work presented in this project should be considered an achievement as we have shown that our approach can design allocations that are more efficient than those of the existing in the UK grid today. This is a sign of possibilities for improvement in the world of renewable energy which is growing ever more important.

Throughout this project we have discovered and discussed a range of limitations that we would have liked to improve on given the time. Mostly the size of our training data set is a limitation to our accuracy that would improve the confidence we have in our claims. Another limitation is that we have only shown this proof of concept on a limited selection of candidate locations for reasons we described in the report, however this does give us any evidence to claim that this approach would extend to a more generalised case which we originally set out with the hopes of achieving.

Bibliography

[1] Serban AC, Lytras MD. Artificial Intelligence for Smart Renewable Energy sector in Europe – Smart Energy Infrastructures for next generation Smart Cities. IEEE Access. 2020;8:1-1.

[2] Yousif M, Ai Q, Wattoo WA, Jiang Z, Hao R, Gao Y. Least cost combinations of solar power, wind power, and energy storage system for powering large-scale grid. Journal of Power Sources. 2019 02;412:710-6.

[3] Wu YK, Lee CY, Chen CR, Hsu KW, Tseng HT. Optimization of the Wind Turbine Layout and Transmission System Planning for a Large-Scale Offshore WindFarm by AI Technology. IEEE Transactions on Industry Applications. 2014 05;50:2071-80.

[4] Qiu R, Liu C, Cui N, Gao Y, Li L, Wu Z, et al. Generalized Extreme Gradient Boosting model for predicting daily global solar radiation for locations without historical data. Energy Conversion and Management. 2022 04;258:115488. Available from: https://www.sciencedirect.com/science/article/pii/S0196890422002849.

[5] Dreher A, Bexten T, Sieker T, Lehna M, Schütt J, Scholz C, et al. AI agents envisioning the future: Forecast-based operation of renewable energy storage systems using hydrogen with Deep Reinforcement Learning. Energy Conversion and Management. 2022 03;258:115401.

[6] Department for Business EIS. Energy Trends About this release Information on energy production, trade, and consumption in the UK for total energy and by specific fuels; 2021. Available from: https://assets.publishing.service.gov.uk/government/

uploads/system/uploads/attachment_data/file/1043279/Energy_Trends_December_2021.pdf.

[7] Agency IRE. Renewable Power Generation Costs in 2020. Internation Renewable Energy Agency; 2021. Available from: https://www.irena.org/publications/2021/Jun/Renewable-Power-Costs-in-2020.

[8] GreenMatch. How Much Do Solar Panels Cost (2022 Prices)?; 2022. Available from: https://www.greenmatch.co.uk/blog/2014/08/what-is-the-installation-cost-for-solar-panels#labour-costs.

[9] Department for Business EIS. Energy Trends: UK renewables; 2022. Available from: https://www.gov.uk/government/statistics/energy-trends-section-6-renewables.

[10] Corporation VC. Weather Data Weather API — Visual Crossing;. Available from: https://www.visualcrossing.com/.

[11] ELEXON. Balancing Mechanism Reporting Service;. Available from: https://www.bmreports.com/bmrs/?q=help/about-us.

[12] Grid N. ESO Data Portal: Home — National Grid Electricity System Operator;. Available from: https://data.nationalgrideso.com/.

[13] Elexon BM Unit Details;. Available from: http://www.netareports.com/data/elexon/bmu.jsp.

[14] Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, et al. Scikit-learn: Machine learning in Python. Journal of Machine Learning Research. 2011;12:2825–2830.

[15] XGBoost: A scalable tree boosting system. ACM; 2016. Available from: http://doi.acm.org/10.1145/2939672.2939785.

[16] Fu W, Olson R, Nathan, Jena G, PGijsbers, Augspurger T, et al. EpistasisLab/tpot: v0.11.5. 2020 06. Available from: https://doi.org/10.5281/zenodo.3872281.

[17] Deb K, Pratap A, Agarwal S, Meyarivan T. A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation. 2002 04;6:182-97.

[18] Carvalho AG, Araujo AFR. Improving NSGA-II with an adaptive mutation operator. Proceedings of the 11th annual conference

companion on Genetic and evolutionary computation conference - GECCO '09. 2009.

[19] Chichakly K. Multiobjective Optimization — Making Connections; 2018. Available from: https://blog.iseesystems.com/modeling-tips/multiobjective-optimization/.

[20] Zeng GQ, Chen J, Li LM, Chen MR, Wu L, Dai YX, et al. An improved multi-objective population-based extremal optimization algorithm with polynomial mutation. Information Sciences. 2016 02;330:49–73. Available from: https://www.sciencedirect.com/science/article/pii/S0020025515007276.

[21] Sedarous S, El-Gokhy SM, Sallam E. Multi-swarm multi-objective optimization based on a hybrid strategy. Alexandria Engineering Journal. 2018 09;57:1619–1629. Available from: https://www.sciencedirect.com/science/article/pii/S1110016817302284.

[22] Got A, Moussaoui A, Zouache D. A guided population archive whale optimization algorithm for solving multiobjective optimization problems. Expert Systems with Applications. 2020 03;141:112972.

[23] Terrell M, Mikel L, Ruben G, P C Santiago, Daniel f, K B Petersen. Symbolic Regression via Deep Reinforcement Learning Enhanced Genetic Programming Seeding. Advances in Neural Information Processing Systems. 2021 12;34. Available from: https://proceedings.neurips.cc/paper/2021/hash/d073bb8d0c47f317dd39de9c9f004e9d-Abstract.html.

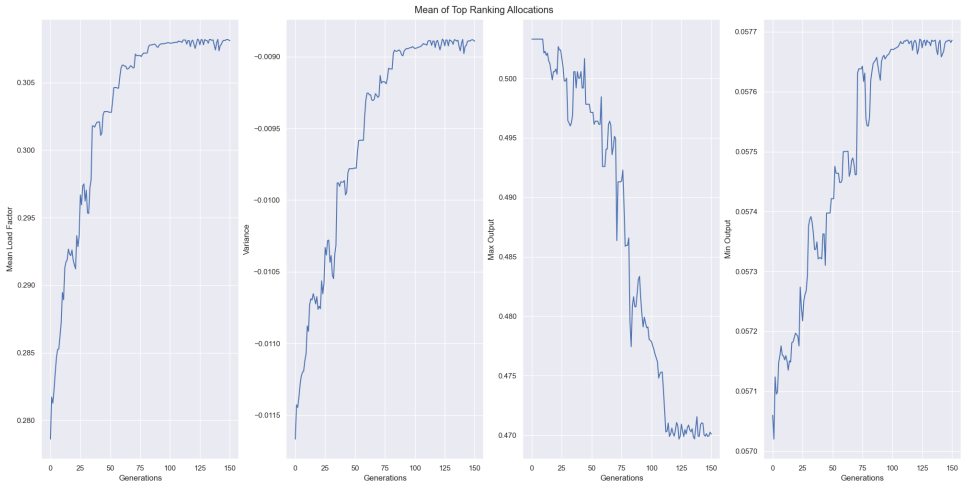

Figure 1: Trace of mean objective values of top performing members of population over the search time from initial seed 1.

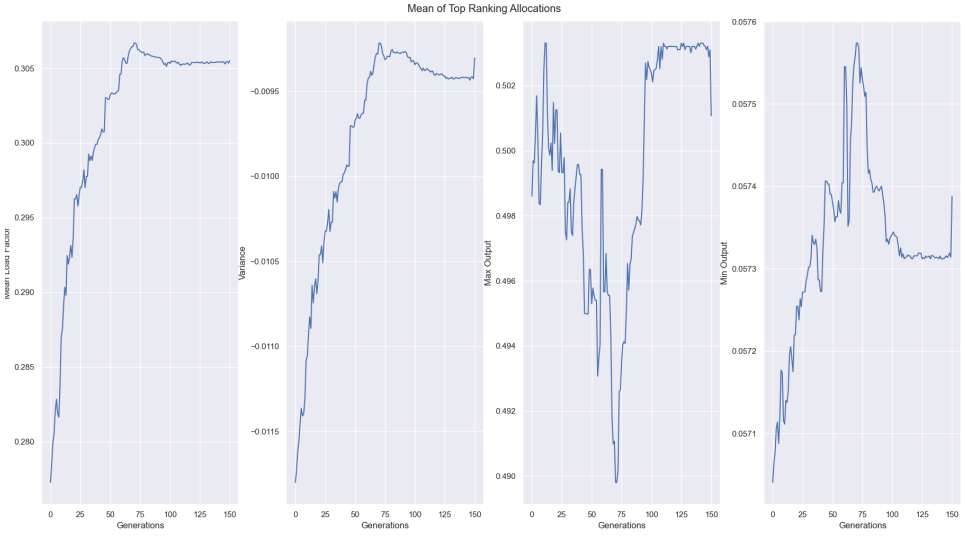

Figure 2: Trace of mean objective values of top performing members of population over the search time from initial seed 2.