Evolving Counterfactual Explanations with Particle Swarm Optimization and Differential Evolution

Hayden Andersen
Victoria University of Wellington
Wellington, New Zealand
andershayd@ecs.vuw.ac.nz

Andrew Lensen
Victoria University of Wellington
Wellington, New Zealand
andrew.lensen@ecs.vuw.ac.nz

Will N. Browne
Queensland University of Technology
Brisbane, Australia
will.browne@qut.edu.au

Yi Mei
Victoria University of Wellington
Wellington, New Zealand
yi.mei@ecs.vuw.ac.nz

***Abstract*—Counterfactual explanations are a popular eXplainable AI technique, used to provide contrastive answers to "what-if" questions. These explanations are consistent with the way that an everyday person will explain an event, and have been shown to satisfy the 'right to explanation' of the European data regulations. Despite this, current work to generate counterfactual explanations either makes assumptions about the model being explained or utlises algorithms that perform suboptimally on continuous data. This work presents two novel algorithms to generate counterfactual explanations using Particle Swarm Optimization (PSO) and Differential Evolution (DE). These are shown to provide effective post-hoc explanations that make no assumptions about the underlying model or data structure. In particular, PSO is shown to generate counterfactual explanations that utilise significantly fewer features to generate sparser explanations when compared to previous related work.**

***Index Terms*—explainable AI, counterfactual explanation, particle swarm optimization, differential evolution**

I. Introduction

Explainable AI [1] (XAI) is a field of AI that focuses on explaining the predictions made by AI systems. These explanations can be intrinsic to the model or post-hoc, they can be tailored to explain specific models or model agnostic, and they can explain the entire model or a local area of the model. Counterfactual Explanations (Counterfactuals) are a post-hoc XAI technique that aims to produce an explanation for how a given input would need to be different to produce a different, more desired, output by a given AI system [2].

As a post-hoc method, a desired property of systems to generate counterfactual explanations is that they should make as few assumptions as possible about the underlying model and data being explained. Despite this, many existing studies to produce these explanations expect either a specific underlying model [3][4] or rely on the data itself to be differentiable [5], which is often not the case.

Previous research [6][7] has shown that population-based evolutionary algorithms have strong promise in generating entirely model-agnostic counterfactual explanations. However, these papers only explore the use of Genetic Algorithms (GAs). This carries the issue that counterfactual generation is an inherently continuous problem on many datasets, and GAs have been shown to be unable to find optimal solutions for many continuous problems [8][9]. This means that these algorithms are unlikely to generate counterfactual explanations that are as similar as possible to the original instance, which is the goal of counterfactual optimisation.

Particle Swarm Optimisation (PSO) [10] is an Evolutionary Computation (EC) method that utilises a number of particles representing solutions, exploring the search space with these particles. PSO is suited for use in continuous domains as each particle will travel directly through the data space, the data domain implicitly specified by the training data.

Differential Evolution (DE) [11] is another EC method that utilises a population of solutions, iteratively improving the population by combining different solutions. In opposition to GA and PSO, DE will only create a new solution in the population if it is an improvement on previous solutions. As with PSO, DE has been developed for use over continuous domains.

This paper explores the use of PSO and DE to generate counterfactual explanations for given class predictions in a black-box model. These are both algorithms that are developed for a continuous space, and make no assumptions about the structure of the underlying search space. The specific goals of this research are to:

- propose new approaches to generate counterfactual explanations utilising PSO and DE;
- evaluate the PSO-based and DE-based algorithms against existing GA work on several datasets of varying complexity;
- explore counterfactual explanations produced for a simple, real-world, dataset, which will be used to qualitatively evaluate the accuracy (where accuracy refers to the ability of the algorithm to produce a sensible explanation for unseen test data) of the explanations.

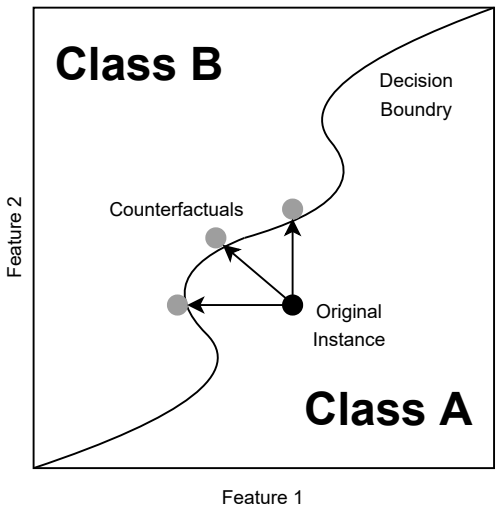

Fig. 1. Counterfactual explanation example. A counterfactual can be produced by modifying one of each of the features, or by modifying both of them.

II. Background

A. Counterfactual Explanations

Counterfactual explanations, introduced by Watcher et al. [2], are artificial data points that are as close as possible to an original instance but produce a different output in a black-box prediction model. Figure 1 shows an example of different counterfactuals produced for a data point, changing the predicted class from A to B. This illustrates that a counterfactual can modify a subset of all the possible features (dimensions in the search space), usually modifying each feature by a lesser magnitude the more features are modified.

Counterfactual explanations are a useful post-hoc XAI method for ML models as they have been shown to fulfil the 'right to explanation' of the European data regulations [2]. Counterfactual explanations are also known for their ability to provide a contrastive explanation, which has been shown to be more effective and understandable to users [12].

An important consideration for counterfactual generation is the number of features that are modified to create the counterfactual. Fewer features changed means that the counterfactual is more likely to be accepted by users due to higher simplicity [12]. To account for this, the distance between the original instance and the new instance is often calculated using the Manhattan distance, also known as the L_1 norm. This is due to the sparsity inducing properties of the L_1 norm [13], [14], which will push the individual differences of as many features as possible to be zero. Equation (1) gives the Manhattan distance between two points F_1 and F_2.

$$D = \sum_{i=1}^{dim(F)} |F_{1i} - F_{2i}| \tag{1}$$

In order to induce more explicit control over the generated explanations, a number of other evaluation metrics are proposed. Most notably, Dandl et al. [7] proposed a multi-objective algorithm incorporating four objectives: the similarity to the desired output, the distance from the original instance, a count of features modified, and an approximation of how well the counterfactual fits into the original data distribution. However, these fail to work well as a multi-objective problem as these objectives are not all conflicting.

Counterfactual explanations can be taken as a full explanation on their own [7][15], or they can be used as a tool to learn and explain more information about the data or black-box model[6] [16].

B. Particle Swarm Optimisation

Particle Swarm Optimisation (PSO) [10] is an EC method that utilises a number of particles representing solutions, exploring the search space with these particles. PSO is optimised to be used over continuous domains, as each particle will travel directly through the data space. A particle is represented as a vector of values, where each value represents the position of the particle in that particular dimension. Similarly, each particle has a given velocity through each dimension of the data. The full representation of a single particle is therefore given as $([x_1, x_2, x_3, ..., x_D], [v_1, v_2, v_3, ..., v_D])$ where x_i and v_i give the position and velocity along dimension i, and D is the total number of dimensions in the data.

At each step of the algorithm, each particle will move through the search space, tending towards both the best position that particular particle has found (denoted as *pbest*) and the best position the full swarm has found (denoted as *gbest*). For a given particle the position and velocity updates can be defined according to Equations (2) and (3) respectively, where t represents the tth step in the search process, i represents a single dimension of the data, r_1 and r_2 are random values sampled from U(0, 1), and w, c_1, and c_2 are given constants.

$$x_i^{t+1} = x_i^t + v_i^{t+1} \tag{2}$$

$$v_i^{t+1} = w \cdot v_i^t + c_1 r_1 (pbest_i - x_i^t) + c_2 r_2 (gbest_i - x_i^t) \tag{3}$$

The constant w refers to the inertia weight of the algorithm, used to balance the exploration and exploitation of the particles to improve convergence. The constants c_1 and c_2 refer to acceleration constants, used to balance of contributions of pbest and gbest. Through numerical analysis, it has been found that for a general case the best values for these constants are $w = 0.7298$, $c_1 = 1.4962$, and $c_2 = 1.4962$ [17].

C. Differential Evolution

DE [11] is an EC method that maintains a population of solutions, represented similar to the solutions in PSO. The difference is in how this population is modified: in DE, a solution is mutated and then combined with another solution in the population in order to create a new solution. If this defined solution performs better than the original solutions, then it is added to the population. There are many variations of DE, but this paper will focus on the DE best/1/bin schedule.

DE is designed to perform well on nonlinear and non-differentiable problems, as it does not directly travel through

the search space. In addition to this, it does not make any specific assumptions about the shape of that search space.

A solution in DE is represented in a similar way to the spatial position in PSO. Given D dimensions, a solution is given as the vector $[x_1, x_2, x_3, ..., x_D]$. At each stage of the algorithm, each solution s will undergo combination. For combination, the best solution in the population x_{best} is combined with two other random solutions in the population x_1 and x_2 according to Equation (4) to form a *donor* vector v. The constant F is known as the scale factor, controlling the step size of the search.

$$v_i = x_{best} + F(x_1 - x_2) \tag{4}$$

Once the donor vector has been generated, a new solution y is created according to Equation (5). At each index i, a random value $r_i \sim U(0, 1)$, is sampled. The constant CR is known as the crossover probability, controlling how often the new solution uses the donor or original solutions.

$$y_i = \begin{cases} v_i & r_i < CR \\ s_i & otherwise \end{cases} \tag{5}$$

Once y has been found, the fitness is compared to the original solution s. If it has better fitness, it replaces s in the population. [18]

Later work found that varying the value of F throughout the optimisation improved the convergence of the optimisation [19]. Known as dithering, this samples F uniformly from a user-defined range at each iteration of the algorithm.

D. Related Work

While not all EC methods, there exists a variety of methods for generating counterfactual explanations for black-box models. Karimi et al. [15] proposed an algorithm utilising Satisfiability Modulo Theories (SMT) solvers, treating counterfactual production as a formal verification problem. Grath et al. [20] proposed a method to generate counterfactuals using the Nelder-Mead algorithm, a simplex-based direct search algorithm. Poyiadzi et al. [21] proposed a graph-based algorithm for counterfactual generation. An issue with many of these algorithms, along with others, is that they will only produce a single counterfactual for the instance. Population-based methods such as EC can address this issue, allowing for counterfactual explanations to be explored in multiple areas of the problem space at the same time.

An important population-based precursor to later EC-based algorithms for counterfactual generation is the Growing Spheres algorithm proposed by Laugel et al. [4]. This searches around the original instance by placing points along an increasingly growing sphere centred on the instance until a point is found that gives the desired prediction. Greedy feature selection is then used to set as many features to the original values as possible to produce the most sparse, and therefore simplest, counterfactual. One issue with this method is that if the sphere sits at a diagonal edge of the decision boundary then the algorithm will terminate before discovering further away but sparser counterfactuals. Figure 2 demonstrates this issue.

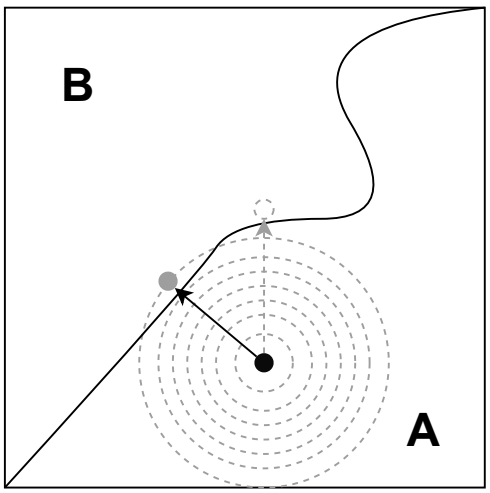

Fig. 2. Potential non-optimality of the Growing Spheres algorithm. The algorithm will terminate having found the counterfactual shown in solid grey, when there is another potential explanation that only modifies a single feature. Sparser explanations that modify those features by a higher amount are known to be preferred over dense explanations [22]

Genetic algorithms (GA) have been used to generate counterfactual explanations, producing feasible and interpretable populations of explanations. Sharma et al. [6] first proposed a GA based method that follows a relatively simple GA optimisation method, evaluating each counterfactual in the population using Manhattan distance. Dandl et al. [7] then expanded on this idea with an Evolutionary Multiobjective algorithm on the four objectives described above. This algorithm produces an approximate Pareto front of counterfactuals that give different trade-offs between the four objectives, however, as not all the objectives are conflicting many of the solutions in the front will have poor performance.

An issue with these methods is that counterfactual production is a problem over a continuous range of data, which can result in a simple GA algorithm struggling to converge to a solution [8][9] without additional local optimisation. Other population-based methods, such as PSO or DE, are more appropriate to create counterfactuals as similar as possible to the original data as they are optimised to work over continuous domains.

III. Proposed Method

The proposed approaches are to utilise custom PSO and DE algorithms respectively in order to generate counterfactual explanations for a provided black-box model f. The use of population-based algorithms that are designed to work with continuous data is hypothesised to be able to generate explanations that are closer to the decision boundary than previous EC methods, resulting in a minimal change from the original instance.

Distance between generated counterfactuals and the original point is measured as discussed in Section II-A, using

the Manhattan distance d. This encourages the algorithm to change as few features as possible for each particle during the evolutionary process.

For both algorithms, each solution is represented as a feature vector of floating-point values. These vectors have the same dimensions as the points in the original data. Given an original instance i, the optimisation task for the algorithms is to find a counterfactual c that optimises Equation (6), where $f(c)$ results in the desired class.

$$c = \min_{c} d(i, c) \tag{6}$$

In order to treat each feature as equally as possible, both algorithms require that the data is scaled to a common range before they begin. For simplicities sake, this should generally be taken as the range [0, 1]. In addition to simplifying the calculations of the data, this scaling also allows for a more fair evaluation of the algorithms, as the evaluation will also treat each feature equally.

Because of the fact that the data is generally scaled from [0,1], the set of valid counterfactual solutions targeting a specific class t is defined according to Equation (7).

$$C = \{c_{1..n} | n = dim(i), f(c) = t, \forall c_i : c_i \in [0, 1]\} \tag{7}$$

Algorithm 1 is used to generate counterfactuals using PSO. This algorithm requires a pretrained model f, a target prediction result t, and a point to be explained i. For the desired number of iterations, each particle will be updated based on both the best counterfactual they have found so far and the best counterfactual the entire population has found so far. This algorithm can either return just the fittest counterfactual produced or the entire population of counterfactuals.

Algorithm 2 is used to generate counterfactuals using DE. As with PSO, this algorithm requires a pretrained model f, a target prediction result t, and a point to be explained i. For the desired number of iterations, each solution will be updated based on a combination of that solution and multiple other solutions in the population. As DE eventually converges on a single point, this algorithm returns only the fittest counterfactual produced.

Algorithm 1: PSO algorithm for counterfactual generation

Parameters: w, c_1, c_2
Data: f model to be explained, i point to be explained, t desired prediction

generate population randomly, sampling initial values from U(0, 1);
$gbest \leftarrow \infty$;
for *particle in population* **do**
  $pbest \leftarrow \infty$;
  $v \leftarrow$ random values from U(0, 0.5);
end
for *desired number of iterations* **do**
  for *particle in population* **do**
    if $f(particle)$ *is* t **then**
      $fitness \leftarrow d(particle, i)$ (Equation (1));
    else
      $fitness \leftarrow \infty$;
    end
    if *fitness* $<$ $pbest$ **then**
      $pbest \leftarrow$ particle;
    end
    if *fitness* $<$ $gbest$ **then**
      $gbest \leftarrow$ particle;
    end
  end
  for *particle in population* **do**
    update v according to Equation (3);
    update particle according to Equation (2);
  end
end
return full population or fittest particle;

IV. Experimental Design

A. Datasets

Six datasets were selected to evaluate the proposed methods. Four datasets were sourced from the UCI machine learning repository [23], one dataset (kc2) was sourced from the OpenML platform[24], and the final dataset (Penguins) was sourced from an online R repository [25]. These datasets were chosen as they are all classification problems, over a continuous domain. They provide a steady increase in difficulty for the counterfactual problems. Table I shows the datasets chosen, ordered approximately in order of complexity.

The Penguins dataset is chosen as it is a simple toy problem that can be used to easily visually inspect the results. It is used as an alternative to the well-known iris dataset, keeping the same property of being a useful toy problem, but is easier to visualise for a qualitative evaluation [26].

B. Experiment Setup

To evaluate the proposed algorithms they are compared against the GA method proposed by Sharma et al. [6]. This gives a good comparison to a similar EC-based algorithm for counterfactual explanation production, with the main difference between the three algorithms being only how the population is updated. [1]

For the PSO algorithm, standard parameters of $w = 0.7298$, $c_1 = 1.4962$, and $c_2 = 1.4962$ are used [17]. While more fitting parameters could be found for each individual problem, the proposed method is intended to fit any data without too much modification and an exploration of perfectly optimal parameters is beyond the scope of this paper. A range of population sizes and iterations for PSO were explored, and from these explorations, a population size of 200 was used for Penguins, Breast Cancer Wisconsin, and Wine; a population size of 500 was used for Steel Plates Faults and Dermatology. For all experiments, 500 generations were utilised. The parameters used for GA are the same as those defined by Sharma et al.,

[1] Implementation code for the proposed methods can be found at https://github.com/HaydenAndersen/ECCounterfactuals

Algorithm 2: DE algorithm for counterfactual generation

Parameters: $Fmin$, $Fmax$, CR
Data: f model to be explained, i point to be explained, t desired prediction

generate population randomly, sampling initial values from U(0, 1);
$xbest \leftarrow \infty$;
for *desired number of iterations* **do**
  for *solution in population* **do**
    if $f(solution)$ *is* t **then**
      $fitness \leftarrow d(particle, i)$ (Equation (1));
    else
      $fitness \leftarrow \infty$;
    end
    if *fitness* $< xbest$ **then**
      $xbest \leftarrow$ solution;
    end
  end
  for *solution in population* **do**
    select random individuals x_1, x_2 from population;
    $F \leftarrow U(F_{min}, F_{max})$;
    generate v according to Equation (4);
    generate y according to Equation (5);
    if $f(y)$ *is* t *and distance(y, i)* $<$ *fitness* **then**
      replace solution with y in population;
    end
  end
end
return fittest solution;

TABLE I
CHOSEN DATASETS

Name	Features	Instances	Classes
Penguins	4	333	3
Breast Cancer Wisconsin (BCW)	10	699	2
Wine	13	178	3
kc2	21	522	2
Steel Plates Faults (SPF)	27	1941	2
Dermatology	34	366	6

with the exception of the population size. Initial experiments found that the equation used in the paper produced far too large a population size, which was not providing any visible performance improvement. For the sake of fair comparison, the same population size and number of iterations were used as the PSO. This results in the same number of evaluations between algorithms. The standard parameters of DE as defined in the SciPy Python library at the time of writing are used [27]. These are a population size of 15, 1000 iterations, $CR = 0.7$, $F_{min} = 0.5$, and $F_{max} = 1.0$.

For each of the three algorithms, a single experimental run takes the following steps:

1) Scale the input data so that each feature is in the range [0, 1].
2) Train a black-box model on the provided training data. For the sake of these experiments, a random forest classifier was used as the black-box model. However, any classifier model can take the place of the random forest with no changes to the rest of the algorithm.
3) Select a single random instance from the data.
4) Use the black-box model to predict the class of the instance.
5) For each class other than the predicted one, use the chosen algorithm to find a single counterfactual that the black-box predicts as that class.
6) Record the discovered counterfactual, as well as both the scaled change in each feature and the change in each feature in the original data ranges.
7) Repeat from Step 3 20 times, using the same black-box model but a different chosen instance.

Of note is that each instance will have a counterfactual explanation generated for each possible class and that multiple instances are explained for the same model. This is to remove potential false-positive results, where good performance is reported simply because of an instance being randomly selected near a decision boundary. In addition to this, the random selection of instances and the training of the black-box model is provided with a randomisation seed so that they are consistent between testing the three algorithms. This means that for the same seeds, each algorithm will be tested on the same black-box model and counterfactuals will be generated from the same initial instances.

As steps 2, 3, and 5 are all non-deterministic, each algorithm has 30 experimental runs performed. This means that, for a dataset with 3 classes, each algorithm will be used to generate a total of $(3 - 1) \times 20 \times 30 = 1200$ counterfactuals.

V. Results and Discussion

A. Results

The results of the initial experiments on the three datasets are shown in Table II. For each combination of dataset and algorithm, the table shows the mean number of features that are modified in generated counterfactual explanations. In addition to this, it shows both the mean Manhattan distance (L1) and the mean Euclidean distance (L2) from the original points to the counterfactuals. The Euclidean distance is displayed as it gives a measurement of exactly how similar the counterfactuals are to the original instance, not worrying about how many features are modified.

A Friedman test followed by post-hoc Nemenyi tests are performed for the features changed, L1, and L2 metrics, with a α of 0.05. For each metric, a $\uparrow$ is shown if the result is statistically significantly better than both other algorithms, and a $\downarrow$ is shown if the result is statistically significantly worse than both other algorithms.

B. Initial Analysis

For all six datasets, the PSO algorithm modifies a statistically significant fewer number of features than the other

TABLE II
PERFORMANCE OF BEST COUNTERFACTUALS

Dataset	Method	Features changed	L1	L2
BCW	PSO	4.15↑	0.87	0.55
	DE	9.16	0.81↑	0.48↑
	GA	9.67↓	0.9↓	0.51
Dermatology	PSO	13.82↑	5.48↑	1.46↑
	DE	33.48	7.8	1.72
	GA	33.74	8.1↓	1.79↓
Penguins	PSO	2.05↑	0.31↑	0.25↑
	DE	3.34	0.31	0.25
	GA	3.74↓	0.32↓	0.26↓
SPF	PSO	10.22↑	0.6↓	0.34↓
	DE	13.85	0.13↑	0.08↑
	GA	25.86↓	0.24	0.12
Wine	PSO	3.95↑	0.54	0.36↓
	DE	11.61	0.5↑	0.33↑
	GA	12.37↓	0.56↓	0.34
kc2	PSO	9.49↑	0.19	0.09
	DE	14.22	0.59↓	0.17↓
	GA	19.77↓	0.14↑	0.06

algorithms. For the Breast Cancer Wisconsin, Wine, and Dermatology datasets PSO modifies less than half as many features as the other two algorithms. This behaviour is likely due to the behaviour of PSO compared to the other two algorithms. As discussed above, GA struggles to converge for continuous problems. DE will only accept a new solution if it is an improvement on previous solutions, which means if it begins to converge on a solution that modifies a large number of features then it will simply continue that convergence. In contrast, PSO allows for more exploration of different solutions, allowing for more sparse counterfactuals to be found.

The DE algorithm shows a statistically significant improvement over the other algorithms on the L1 and L2 metrics for the Breast Cancer Wisconsin, Steel Plates Fault and Wine datasets. The PSO algorithm shows a statistically significant improvement over the other algorithms on the L1 and L2 metrics for the Dermatology and Penguins datasets. The only time the GA algorithm outperforms both other algorithms is on the L1 metric in the kc2 dataset. This performance spread is consistent with the idea discussed previously that GA is unsuited for optimising across the continuous domain that is required for counterfactual production. Given that the PSO algorithm produces considerably sparser solutions than the other algorithms and that it generally produces either the most similar or close to the most similar explanations to the original point, it is in general the most effective method.

An interesting outlier in the results is the Steel Plate Fault dataset. Despite changing fewer features than both other algorithms, the PSO algorithm creates counterfactual explanations that are significantly further away from the original point. This is shown by the fact that both the L1 and L2 metrics are significantly higher for the PSO. This is likely due to the shape of the data space of the dataset, as a solution that does not result in the current class will be set to a very poor fitness. Due to the fact that GA and DE both work by combining existing

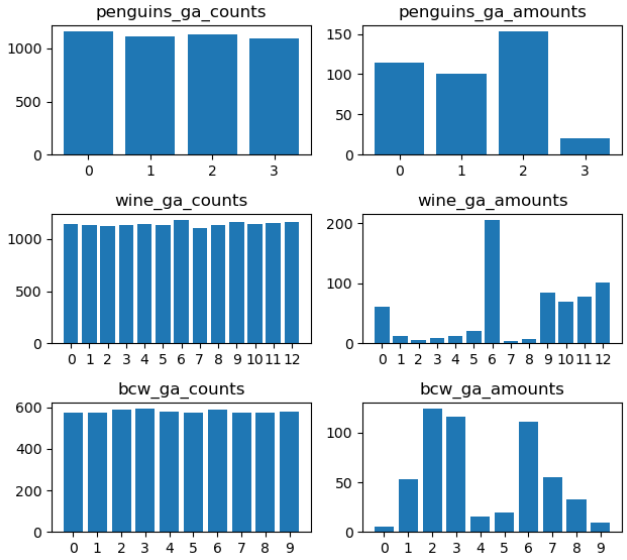

Fig. 3. Feature-wise analysis of GA

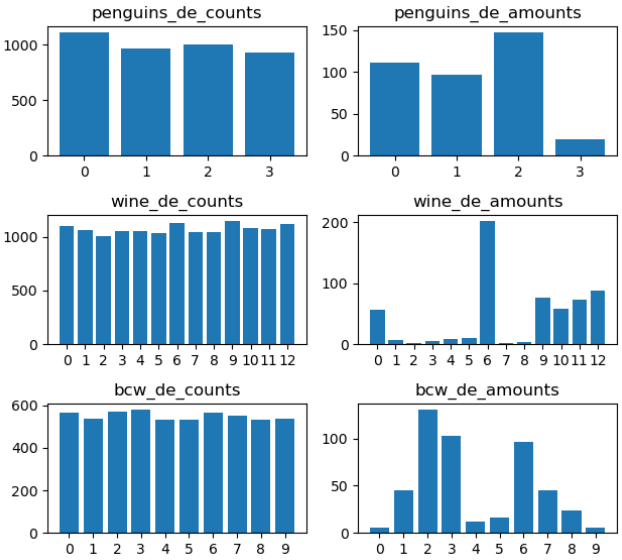

Fig. 4. Feature-wise analysis of DE

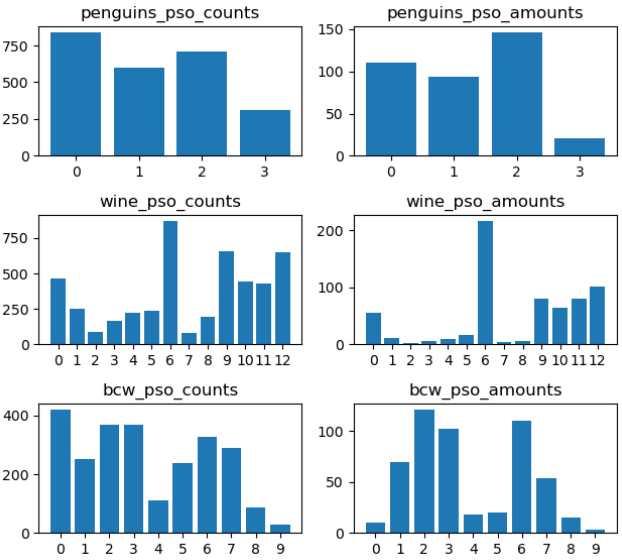

Fig. 5. Feature-wise analysis of PSO

solutions, PSO will be spending a lot more of the computation time with solutions resulting in these poor fitnesses. Future work will explore the filtering or re-generation of particles for which this occurs.

C. Further Analysis

A useful consideration that arose during the analysis of the produced counterfactuals was that some features could be more important than others for the purpose of generating counterfactual explanations. To explore this, the three datasets with the fewest number of features were chosen for analysis. The counterfactuals produced were analysed, with the number of times each feature was modified among the different experimental runs and the total scaled size of these changes being recorded. Graphs of this analysis are shown in Figures 3-5.

The strength of the PSO method in reducing the number of features modified can be further observed in these graphs. While both DE and GA modify each feature an almost uniform number of times, in both the Penguins and Breast Cancer Wisconsin datasets the PSO algorithm shows clear variation in how much each feature is used in the counterfactual explanations. In addition to this, the amount each feature is used does not necessarily correlate to how much that feature was modified. For example, despite all algorithms modifying all features in the Wine dataset an almost uniform number of times, the amount each was modified by shows strong inter-feature variance. When reporting counterfactual explanations to users, a basic metric of $\frac{\text{amount modified}}{\text{times modified}}$ should be considered to determine how important of a consideration each feature should be. This balances the importance of the feature and the amount of change required with the abnormality of that feature appearing in an explanation.

D. Qualitative Analysis

To give a visual idea of the quality of counterfactuals produced by the PSO algorithm, the Penguins dataset is used. The purpose of this dataset is to differentiate between three species of penguin found in Antarctica. These are Adelie, Chinstrap, and Gentoo penguins. Figure 6 shows an example of each of the three penguin species, sourced from an online bird-watching directory [28]. The features in the dataset are bill length, bill depth, flipper length, and body mass.

For this analysis five instances from the data are chosen at random, and a counterfactual to each other type of penguin is generated. Table III shows the best counterfactual explanations produced in each of these runs.

Comparing the produced counterfactuals to the visual traits of the three species, it can be observed that the produced counterfactual explanations are reasonable. Adelie penguins have short bills and flippers, so increasing the bill length causes one to be classified as a Chinstrap and increasing the flipper length causes one to be classified as a Gentoo. Likewise, the Chinstrap can be classified as an Adelie by reducing the size of the bill, or as a Gentoo by increasing the length of the flippers. As the species with the largest features, the Gentoo penguins can be classified as Adelie by reducing the size of both the bill and flippers or as Chinstrap by increasing the bill depth but decreasing the length of the flippers and the body mass. The final counterfactual is the most abnormal one, as it is the only one of the 10 generated explanations that considers the body mass. However, this is still consistent with the visual appearance of the different species, where the Chinstrap looks a lot leaner than the Gentoo.

TABLE III
PENGUIN COUNTERFACTUALS

From	To	B. length	B. depth	F. length	Mass
Adelie	Chinstrap	+5.95	-	-	-
	Gentoo	-	-1.15	+14.0	-
Adelie	Chinstrap	+5.40	-	-	-
	Gentoo	-	-2.45	+21.0	-
Chinstrap	Adelie	-5.05	-0.05	-	-
	Gentoo	-	-	+8.50	-
Gentoo	Adelie	-5.80	-	-2.50	-
	Chinstrap	+0.75	+1.35	-3.00	-
Gentoo	Adelie	-8.15	-	-9.50	-
	Chinstrap	-	+2.55	-1.50	-25.0

While this is a simple task, this analysis can give us confidence that the algorithm works as intended. This can be used to motivate the use of the algorithm for more complex tasks that are more difficult for a non-expert to reliably qualitatively verify.

VI. CONCLUSIONS

This paper introduced new algorithms to produce counterfactual explanations utilising PSO and DE. To our knowledge, this is the first work to explore the use of these algorithms for the task of counterfactual production. These explanations are shown to modify fewer features from the original instances and to require changes of a lesser magnitude than previous evolutionary work to generate counterfactuals. In addition to this, graphical analysis of individual features has shown that the PSO algorithm is better at utilising only important features compared to other similar work. Finally, a simple qualitative analysis showed that the generated counterfactuals are reasonable, and reflect real-world causal relationships in the data.

Future work will focus on two main areas of the method: a more complex fitness function and niching to produce a more diverse population of explanations. The more complex fitness function will consider the abnormality of feature values in instances in order to produce more human-friendly explanations that are more likely to be accepted by users. This is in line with existing research on how a human would produce a counterfactual explanation [29]. In addition to this, more explicit penalisation will be explored for the number of features used in order to encourage sparser explanations.

Early experiments in applying niching to the PSO algorithm have shown promise to produce counterfactual explanations utilising a range of different distinct feature subsets. This will be further explored to allow the presentation of multiple explanations to users, that they can then select from.

Fig. 6. Adelie, Chinstrap, and Gentoo penguins [28]

REFERENCES

[1] C. Molnar, *Interpretable Machine Learning: A Guide for Making Black Box Models Explainable*. Leanpub, 2019.

[2] S. Wachter, B. Mittelstadt, and C. Russell, "Counterfactual Explanations without Opening the Black Box: Automated Decisions and the GDPR," *SSRN Electronic Journal*, Nov. 2017.

[3] S. Liu, B. Kailkhura, D. Loveland, and Y. Han, "Generative Counterfactual Introspection for Explainable Deep Learning," in *Proceedings of the IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, IEEE, Jul. 2019, pp. 1–5.

[4] T. Laugel, M.-J. Lesot, C. Marsala, X. Renard, and M. Detyniecki, "Inverse Classification for Comparison-based Interpretability in Machine Learning," in *Information Processing and Management of Uncertainty in Knowledge-Based Systems. Theory and Foundations*, Springer International Publishing, Dec. 2017.

[5] A. Dhurandhar, T. Pedapati, A. Balakrishnan, P.-Y. Chen, K. Shanmugam, and R. Puri, *Model agnostic contrastive explanations for structured data*, 2019. arXiv: 1906.00117 [cs.LG].

[6] S. Sharma, J. Henderson, and J. Ghosh, "CERTIFAI: Counterfactual Explanations for Robustness, Transparency, Interpretability, and Fairness of Artificial Intelligence models," in *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society (AIES)*, ACM, May 2019, pp. 166–172.

[7] S. Dandl, C. Molnar, M. Binder, and B. Bischl, "Multi-Objective Counterfactual Explanations," *Lecture Notes in Computer Science*, vol. 12269 LNCS, pp. 448–469, Apr. 2020.

[8] K. Rasheed, H. Hirsh, and A. Gelsey, "A genetic algorithm for continuous design space search," *Artificial Intelligence in Engineering*, vol. 11, no. 3, pp. 295–305, 1997.

[9] R. Hassan, B. Cohanim, O. de Weck, and G. Venter, "A Comparison of Particle Swarm Optimization and the Genetic Algorithm," in *Proceedings of the 46th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference*, American Institute of Aeronautics and Astronautics, Apr. 2005.

[10] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of the International Conference on Neural Networks (ICNN)*, vol. 4, IEEE, 1995, pp. 1942–1948.

[11] R. Storn and K. Price, "Differential Evolution – A Simple and Efficient Heuristic for global Optimization over Continuous Spaces," *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[12] T. Miller, "Explanation in artificial intelligence: Insights from the social sciences," *Artificial Intelligence*, vol. 267, pp. 1–38, Feb. 2019.

[13] E. J. Candès, J. K. Romberg, and T. Tao, "Stable signal recovery from incomplete and inaccurate measurements," *Communications on Pure and Applied Mathematics*, vol. 59, no. 8, pp. 1207–1223, Aug. 2006.

[14] D. L. Donoho, "For most large underdetermined systems of linear equations the minimal l-norm solution is also the sparsest solution," *Communications on Pure and Applied Mathematics*, vol. 59, no. 6, pp. 797–829, Jun. 2006.

[15] A.-H. Karimi, G. Barthe, B. Balle, and I. Valera, *Model-agnostic counterfactual explanations for consequential decisions*, 2020. arXiv: 1905.11190 [cs.LG].

[16] A. White and A. S. d'Avila Garcez, "Measurable Counterfactual Local Explanations for Any Classifier," in *Proceedings of the 24th European Conference on Artificial Intelligence (ECAI)*, vol. 325, IOS Press, 2020, pp. 2529–2535.

[17] F. Van Den Bergh and A. P. Engelbrecht, "An Analysis of Particle Swarm Optimizers," Ph.D. dissertation, University of Pretoria, ZAF, 2002.

[18] M. Georgioudakis and V. Plevris, "A Comparative Study of Differential Evolution Variants in Constrained Structural Optimization," *Frontiers in Built Environment*, vol. 6, 2020.

[19] K. Price, R. Storn, and J. Lampinen, *Differential Evolution-A Practical Approach to Global Optimization*. Springer Verlag, Jan. 2005, vol. 141, p. 539.

[20] R. M. Grath, L. Costabello, C. L. Van, *et al.*, *Interpretable credit application predictions with counterfactual explanations*, 2018. arXiv: 1811.05245 [cs.AI].

[21] R. Poyiadzi, K. Sokol, R. Santos-Rodríguez, T. D. Bie, and P. A. Flach, "FACE: Feasible and Actionable Counterfactual Explanations," in *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, 2020.

[22] T. Miller, P. Howe, and L. Sonenberg, *Explainable AI: Beware of Inmates Running the Asylum Or: How I Learnt to Stop Worrying and Love the Social and Behavioural Sciences*, 2017. arXiv: 1712.00547 [cs.AI].

[23] D. Dua and C. Graff. "UCI Machine Learning Repository." (2017), [Online]. Available: http://archive.ics.uci.edu/ml.

[24] J. Vanschoren, J. N. van Rijn, B. Bischl, and L. Torgo, "Openml: Networked science in machine learning," *SIGKDD Explorations*, vol. 15, no. 2, pp. 49–60, 2013.

[25] A. M. Horst, A. P. Hill, and K. B. Gorman. "palmerpenguins: Palmer Archipelago (Antarctica) penguin data." (2020), [Online]. Available: https://allisonhorst.github.io/palmerpenguins/.

[26] T. Poisot. "It's time to retire the iris dataset." (2020), [Online]. Available: https://armchairecology.blog/iris-dataset/.

[27] P. Virtanen, R. Gommers, T. E. Oliphant, *et al.*, "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python," *Nature Methods*, vol. 17, pp. 261–272, 2020.

[28] B. Sullivan, C. Wood, M. Iliff, R. Bonney, D. Fink, and S. Kelling, "eBird: a citizen-based bird observation network in the biological sciences," *Biological Conservation*, vol. 142, pp. 2282–2292, 2009.

[29] R. M. Byrne, "Counterfactuals in Explainable Artificial Intelligence (XAI): Evidence from Human Reasoning," in *Proceedings of the IJCAI International Joint Conference on Artificial Intelligence*, 2019, pp. 6276–6282.