An efficient discrimination discovery method for fairness testing

Shinya Sano
Dept. of Info. and Comp. Sci.
Keio University
Yokohama, Japan
sanoshin@doi.ics.keio.ac.jp

Takashi Kitamura
Nat. Inst. of Advanced Industrial Science and Technology
Tokyo, Japan
t.kitamura@aist.go.jp

Shingo Takada
Dept. of Info. and Comp. Sci.
Keio University
Yokohama, Japan
michigan@ics.keio.ac.jp

***Abstract*—With the increasing use of machine learning software in our daily life, software fairness has become a growing concern. In this paper, we propose an individual fairness testing technique called KOSEI. Individual fairness is one of the central concepts in software fairness. Testing individual fairness aims to detect individual discriminations included in the software. KOSEI is based on AEQUITAS by Udeshi et al., a landmark fairness testing technique featuring a two-step search strategy of global and local search. KOSEI improves the local search part of AEQUITAS, based on our insight to overcome the limitations of the local search of AEQUITAS. Our experiments show that KOSEI outperforms AEQUITAS by orders of magnitude. KOSEI, on average, detects 5,084.8% more discriminations than AEQUITAS, in just 7.5% of the execution time.**

***Index Terms*—software testing, algorithm fairness, machine learning**

I. INTRODUCTION

With the increasing use of machine learning (ML) software in our daily life, software fairness has become a growing concern. A famous example is the COMPAS software, which computes risk-assessment scores for recidivism of defendants. It assists in the sentencing process; however, the software makes biased and discriminatory mistakes [1]. For example, the software is likely to falsely rate more black defendants to be risky than white defendants.

Individual fairness is a key concept in software fairness. It is often referred to with the concept of individual discrimination. Individual discrimination occurs when ML software gives different results to two similar individuals that differ in protected attributes. Protected attributes represent attributes often tied to social bias, e. g., gender, race, or age.

Testing ML software for individual fairness is one approach to validate software fairness. It has recently been under active investigation, with various algorithms being proposed. For example, some use random sampling techniques [1], some use probabilistic searches [2], some apply the technique of gradient-based adversarial sampling [3], some apply symbolic execution [4], and some apply constraint solving [5].

In this paper, we tackle black-box individual fairness testing, proposing a technique called KOSEI[1] (Keeper Of Systematic Equality Investigation). KOSEI is based on AEQUITAS, a landmark technique for black-box individual fairness testing, developed by Udeshi et al. [2]. A characteristic feature of AEQUITAS is that its testing algorithm is structured into two phases of global and local search to leverage the robustness of ML algorithms for efficiently detecting discriminations. The two-phased structure of the algorithm has become the basis of many crucial algorithms (e. g., [3], [4]). Focusing on improving the second phase of the algorithm (called local search), KOSEI aims to obtain a higher detecting ability of individual discriminations. Our evaluation of KOSEI shows that KOSEI outperforms AEQUITAS by orders of magnitude. KOSEI detects, on average, 5,084.8% more discriminations than AEQUITAS in just 7.5% of the execution time. Our evaluation also confirms that our technical refinements realize the performance gain.

The paper is organized as follows. Section II reviews individual fairness testing and AEQUITAS by Udeshi et al. [2]. In Section III, we propose our method, KOSEI. In Section IV, we evaluate KOSEI through experiments. We discuss threats to validity in Section V. Section VI discusses related work. Finally, Section VII concludes this paper.

II. BACKGROUND

This section reviews individual fairness testing (referring to [6]) and AEQUITAS [2].

A. Individual fairness testing

Let $P = \{p_1, p_2, \cdots, p_n\}$ be a set of *attributes* (or *parameters*), for $n \in \mathbb{N}$. We use p_i to indicate the i-th attribute in P. Each attribute $p_i (\in P)$ is associated with a set of *values*, called the *domain* of p_i, denoted by $Dom(p_i)$, such that $(Dom(p_i))_{i \in n}$ is pairwise disjoint. The input space $\mathbb{I}$ of a set of attributes P is the Cartesian product of the domains of $p_1, p_2 \cdots p_n (\in P)$, i. e., $\mathbb{I} = Dom(p_1) \times Dom(p_2) \times \cdots \times Dom(p_n)$. An element I of $\mathbb{I}$ is a *data item*, which we may also call a *test case*. We use I_k as the value of the k-th attribute of input $I \in \mathbb{I}$. We also introduce $P_{protected} \subseteq P$ as the set of protected attributes (e. g., gender, race, age). We interpret a ML classifier, whose input space is $\mathbb{I}$, as function f; i. e., we use $f(I)$ to denote the result (i. e., decision) that the trained classifier f makes for input I.

DOI reference number: 10.18293/SEKE2022-064

[1]Japanese word which means fairness

Definition 1 (Discriminatory data and Fairness [2]): Let f be a classifier under test, γ be the pre-determined discrimination threshold (e.g. chosen by the user), and $I \in \mathbb{I}$. Assume $I' \in \mathbb{I}$ such that there exists a non-empty set $Q \subseteq P_{protected}$ and for all $q \in Q$, $I_q \neq I'_q$ and for all $p \in P \setminus Q$, $I_p = I'_p$. If $|f(I) - f(I')| > \gamma$, then I is called a discriminatory data item of the classifier f and is an instance that manifests the violation of (individual) fairness in f.

Example 1: We use the Census Income (aka, Adult) dataset [7] as the running example. Its task is to predict if the income of an adult exceeds $\$50,000$ per year. The dataset contains 32,561 training instances with 13 attributes each. The following example shows a numeric-represented data instance x:

$$x : [4, 0, 6, 6, 0, 1, 2, 1, \dot{1}, 0, 0, 40, 30]$$

The first attribute represents 'age', whose domain is $\{0..9\}$ (where value '4', for instance, means age from 40 to 49 years); the ninth represents 'gender', whose domain is 'male (0)' and 'female (1)'. The running example considers 'gender' (with a dot) as the protected attribute.

Classifier f inputs a data item of the Census data set and returns '1' if the income of an adult exceeds $\$50,000$ per year, and '0' otherwise. According to Definition 1, the data item x is discriminatory, for the classifier f, the protected (i. e., 'gender') attribute, and $\gamma = 0$, if $f(x) \neq f(x')$ with

$$x' : [4, 0, 6, 6, 0, 1, 2, 1, \dot{0}, 0, 0, 40, 30].$$

Observe that x' differs from x only in the 'gender' attribute.

Algorithm 1: Local Search Of Aequitas: $local\_search(D_{global}, f, limit)$

Data: discriminatory data (D_{global}), classifier(f), local iteration limit ($limit$)
Result: Discriminatory data (D_{local})

1 **Procedure** *local_search($D_{global}, f, limit$)*
2 $\quad \sigma_{pr}[p] \leftarrow \frac{1}{|P'|}$ for all $p \in P_{non\_protected}$
3 $\quad \sigma_v[p] \leftarrow 0.5$ for all $p \in P_{non\_protected}$
4 $\quad$ **for** $I \in D_{global}$ **do**
5 $\quad\quad$ **for** i *in* $(0, limit)$ **do**
6 $\quad\quad\quad$ // apply perturbation to I
7 $\quad\quad\quad$ Select $p \in P_{non\_protected}$ with $\sigma_{pr}[p]$
8 $\quad\quad\quad$ Select δ with $\sigma_v[p]$
9 $\quad\quad\quad$ $I[p] \leftarrow I[p] + \delta$
10 $\quad\quad\quad$ **if** *eval_disc(I)* **then** $D_{local} \leftarrow D_{local} \cup \{I\}$
11 $\quad\quad\quad$ update_prob$(I, p, D_{local}, \delta)$
12 $\quad$ **return** $D_{global} \cup D_{local}$

Algorithm 2: $eval\_disc(I)$

Data: A data item (I)
Result: Boolean

1 **Procedure** *eval_disc(I)*
2 $\quad \triangleright$ $\mathbb{I}^{(d)}$ extends I with all values of $P_{protected}$
3 $\quad \mathbb{I}^{(d)} \leftarrow \{I' | \forall p \in P_{non\_protected}. I_p = I'_p\}$
4 $\quad$ **if** $\exists I' \in \mathbb{I}^{(d)}, |f(I) - f(I')| > \gamma$ **then return** True
5 $\quad$ **else return** False

B. AEQUITAS

We review the algorithm of AEQUITAS, focusing on its two-phase-structured algorithm, perturbation, and an algorithm component called *local search*.

1) Two-phase-structured algorithm: The algorithm of AEQUITAS is structured into the two phases of *global* and *local search*. The algorithm starts with the global search, which randomly searches the input space of a ML classifier. Following the global search, the local search searches nearby the discriminatory data detected in the global search.

This two-phase structure of the algorithm is designed to detect discriminatory data effectively, leveraging the robustness of ML classifiers. The robustness of ML classifiers means that similar prediction is likely to be produced for similar data. So, first, the global search scans the whole input space widely to find different kinds of discriminatory data. Then, the local search searches the vicinities of discriminatory data found in the global search using perturbation to find more discriminatory data.

2) Perturbation : Given a data item, perturbation creates a similar data item by adding small changes to it. For example, Definition 2 gives the perturbation of AEQUITAS.

Definition 2 (perturbation): Perturbation g is a function $g : \mathbb{I} \times (P \setminus P_{protected}) \times \Gamma \rightarrow \mathbb{I}$ where $\Gamma = \{-1, +1\}$. If $I' = g(I, p, \delta)$ where $I \in \mathbb{I}$, $p \in P \setminus P_{protected}$ and $\delta \in \Gamma$, then $I'_p = I_p + \delta$, and $I'_q = I_q$ for all $q \in P \setminus \{p\}$.

Observe that the perturbation adds a change of only -1 or +1 to one attribute of a given data item. As the local search scans the vicinity of discriminatory data passed by the global search, the perturbation stipulates the 'vicinity' of data.

3) Local search : Algorithm 1 shows the local search algorithm of AEQUITAS. The algorithm takes three objects as the input (1) f: the ML classifier under test, (2) D_{global}: a set of discriminatory data (passed from the global search), and (3) $limit$: the number of local iterations. The output is discriminatory data found in the local search.

The algorithm begins with preparing lists σ_{pr} and σ_v. Given an attribute p, $\sigma_{pr}[p]$ shows the probability that p is selected to be perturbed. $\sigma_v[p]$ shows the probability that p is perturbed by $\delta = -1$, while $(1 - \sigma_v[p])$ shows the probability that p is perturbed by $\delta = +1$.

After the initialization of σ_{pr} and σ_v, the algorithm applies a perturbation (line 7 – line 9) to each data item (line 4) in the given discriminatory data (D_{global}) for the $limit$ times (line 5). A perturbation chooses an attribute and the direction of perturbation, respectively based on σ_{pr} (line 7) and σ_v (line 8). For each perturbed data item, the algorithm evaluates if it is discriminatory or not (line 10) using the evaluation function (Algorithm 2). Finally, the data item evaluated as discriminatory is added to D_{local} (line 10). Based on evaluation results, the algorithm updates both σ_{pr} and σ_v

for tuning the probabilities. Three strategies (*random*, *semi-directed*, and *fully-directed*) are provided for this probability update. We do not look into the details in this paper because our proposed technique is not very relevant to the strategies.

Example 2: Consider applying the local search of AEQUITAS to the following data item x:

$$x: \quad [7, 4, 26, 1, 4, 4, 0, 0, \dot{0}, 1, 5, 73, 1]$$

Suppose the local search algorithm, for the first iteration (i. e., for the first perturbation), chooses the third attribute and the direction of '$+1$' (respectively based on probabilities recorded in σ_{pr} and σ_v). It thus generates the following data item x', and check if it is discriminatory, where note that the value of the third attribute has changed from 26 to 27:

$$x'_1: \quad [7, 4, \underline{27}, 1, 4, 4, 0, 0, \dot{0}, 1, 5, 73, 1].$$

For the second iteration, the algorithm works based on this newly generated data item x'_1. Suppose it chooses the 12th attribute and direction of '$+1$'. It thus generates and evaluates the following perturbed data item x'_2:

$$x'_2: \quad [7, 4, \mathbf{27}, 1, 4, 4, 0, 0, \dot{0}, 1, 5, \underline{74}, 1]$$

For the third iteration, where it works on the new data x'_2, suppose it chooses the fifth attribute and direction of '-1'. It thus generates and evaluates the following perturbed data item x'_3:

$$x'_3: \quad [7, 4, \mathbf{27}, 1, \underline{3}, 4, 0, 0, \dot{1}, 1, 5, \mathbf{74}, 1]$$

The algorithm continues similarly until the number of iterations reaches the specified limit.

III. KEEPER OF SYSTEMATIC EQUALITY INVESTIGATION

This section proposes an individual fairness testing technique named Keeper Of Systematic Equality Investigation (KOSEI). KOSEI is based on AEQUITAS; it inherits the two-phase structured search strategy (global and local search) of AEQUITAS but improves the local search algorithm. We state the limitations of AEQUITAS's local search, explain technical details of KOSEI, and discuss its advantages.

A. Limitation of AEQUITAS's local search algorithm

1) AEQUITAS's local search algorithm searches space, where there may be little discriminatory data.: First, the algorithm equally searches vicinities of discriminatory data passed from the global search. Suppose, for example, discriminatory data passed by the global search contain two data d_1 and d_2, in the vicinities of which there are 10 and 1000 discriminatory data, respectively. Even though searching the vicinity of d_2 is likely to find more discriminatory data, the algorithm spends an equal amount of search resources (specified as a limit of local search) on d_1 and d_2.

Second, the algorithm may search not only the vicinities of discriminatory data but also those of non-discriminatory data. Note that the algorithm sequentially applies perturbation to generated data, regardless of whether that data is discriminatory or not. For example, suppose x'_1 (generated by the discriminatory data item x by perturbation) in Example 2 is not discriminatory; the algorithm searches the vicinity of x'_1, by applying perturbation and evaluates the perturbed data item.

2) AEQUITAS's local search may waste search resources by evaluating duplicated data.: First, the perturbation process is based on the *probabilistic choice* of parameter-values; thus, the algorithm may generate a data item that has previously been generated. However, AEQUITAS does not avoid such duplications, which causes its inefficiency.

Specifically, the second is more concerned with the algorithm implementation design. The AEQUITAS implementation of Algorithm 1 (coded in Python) is realized using the basin-hopping optimization function[2]. This implementation design seems to aim to detect discriminatory data efficiently. However, as we observe in our experiments of executing AEQUITAS, the implementation generates quite a few duplicated data during its executions (as is also pointed out in [2]-page 9).

B. Two key mechanisms of our local search

We explain the two key mechanisms in the local search of KOSEI: i. e., perturbation and dynamic update of search space.

1) Perturbation : The concept of perturbation of KOSEI inherits that of AEQUITAS (Definition 2); i. e., the perturbation of KOSEI changes the value of only one attribute of a given data item by $-1/+1$. However, KOSEI uses this perturbation concept differently from AEQUITAS. It applies the perturbation to all the unprotected attributes one by one instead of probabilistically choosing one attribute like in AEQUITAS.

Example 3: The following shows data obtained by applying the perturbation to the data item x:

$$\begin{aligned}
x: &\quad [7, 4, 26, 1, 4, 4, 0, 0, \dot{0}, 1, 5, 73, 1] \\
x'_1: &\quad [\mathbf{6}, 4, 26, 1, 4, 4, 0, 0, \dot{0}, 1, 5, 73, 1] \\
x'_2: &\quad [\mathbf{8}, 4, 26, 1, 4, 4, 0, 0, \dot{0}, 1, 5, 73, 1] \\
x'_3: &\quad [7, \mathbf{3}, 26, 1, 4, 4, 0, 0, \dot{0}, 1, 5, 73, 1] \\
x'_4: &\quad [7, \mathbf{5}, 26, 1, 4, 4, 0, 0, \dot{0}, 1, 5, 73, 1] \\
&\qquad\qquad \vdots \\
x'_{22}: &\quad [7, 4, 26, 1, 4, 4, 0, 0, \dot{0}, 1, 5, 73, \mathbf{2}]
\end{aligned}$$

The bold numerics indicate attributes to which the perturbation has been applied. For example, x'_1 is generated by perturbating the first (unprotected) attribute (i. e., 'age') of x to the direction of -1; x'_2 is generated by perturbating the first attribute of x to the direction of $+1$; x'_3 is generated by perturbating the second attribute ('work class') of x to the direction of -1. The number of data obtained from one data item x is at most $\#P' * |\delta|$, where $\#P'$ is the number of unprotected attributes and $|\delta| = |\{-1, +1\}| = 2$. Note that some data generated by perturbation will be invalid and hence will be excluded if the perturbed value in the perturbed data is out of its attribute domain.

[2]https://docs.scipy.org/doc/scipy-1.8.0/html-scipyorg/reference/

Algorithm 3: Local search of KOSEI

```
   Data: Same as Algorithm 1
   Result: Same as Algorithm 1
1  Procedure local_search(D_global, f, limit)
2      // initialize seed data D and D_total
3      // D_total is a global variable
4      D ← D_global
5      for i in (0, limit) do
6          d ← D.pop()
7          for p ∈ P_non_protected do
8              for δ ∈ {−1, +1} do
9                  d′ ← d;  d′[p] ← d[p] + δ
10                 if d′[p] ∈ Dom(p) then continue
11                 if d′ ∈ D_total then continue
12                 if eval_disc(d′) then
13                     // dynamic update of seed data D
14                     D.push(d′)
15                 D_total ← D_total ∪ {d′}
16     return D
```

2) Dynamic update of seed data : Naively, the number of local iterations would be the number of non-protected attributes times 2. In our running example, this iteration limit would be 24 (=12 x 2). However, this would severely limit the search space. Furthermore, in AEQUITAS, users can specify a local iteration limit (by variable *limit* in Algorithm 1), typically 1000 or 2000. These values are much higher than the naive value of 24 stated above. This suggests we need to consider cases where a user-specified local iteration limit is much higher than the naive number of local iterations.

KOSEI is designed to dynamically update the discriminatory data (called *seed data*) that the local search works on to bridge this gap. Although the local search of KOSEI starts working on discriminatory data passed by the global search as seed data, on detecting a new discriminatory data item, it appends that data item to the seed data so that the newly-detected data item can also be an object of the local search. KOSEI searches vicinities of discriminatory data detected in the global search and those of newly detected discriminatory data during the local search. The local search thus terminates when it reaches the iteration limit specified by users or when there is no more seed data.

C. Local search algorithm of KOSEI

Algorithm 3 shows the local search algorithm of KOSEI, which incorporates the two key mechanisms.

Note first that since the proposed local search offers an alternative to that of AEQUITAS, its input and output are designed the same as those of AEQUITAS's local search (in Algorithm 1).

The algorithm sets seed data D with the discriminatory data D_{global} (to be passed by the global search) at line 4. It next iterates the following procedure for the number of times specified by '*limit*': Dequeueing the first data item from the seed dataset D, it applies the perturbation (as explained in Section III-B1). That is, for each attribute of the dequeued data item, it perturbs the value of the attribute by '$-1/+1$'. In doing so, the perturbed data item is checked to see if it is valid (line 10) and if it has been evaluated previously (line 11, where D_{total} remembers the list of previously evaluated data). Next, the data item that passed these checks is evaluated for whether it is discriminatory or not (line 12). If it is evaluated discriminatory, it is appended to the list of seed data (line 14). Finally, the evaluated data (regardless of whether it is discriminatory or not) is added to D_{total} (line 15).

D. Advantages of KOSEI

KOSEI improves the limitations of the local search algorithm of AEQUITAS, discussed in Section III-A.

The first improvement is on the limitation that AEQUITAS's local search scans a search space, where there may be little discriminatory data (as stated in Section III-B1). The local search of KOSEI searches all the neighbors of a given data item, where all the neighbors of a data item mean all the perturbed data of a data item defined in Definition 2. Note, meanwhile, the local search of AEQUITAS does not necessarily work in this way according to its probabilistic search. This strategy of KOSEI aims to search data that are more likely to be discriminatory. The dynamic updating of seed data of KOSEI also can contribute to this aspect. Recall the two data d_1 and d_2 in Section III-A, around which there are respectively 10 and 1000 discriminatory data. In this situation, KOSEI may spend more search resources on d_2 than d_1 since its dynamic updating mechanism updates the seed data with more discriminatory data around d_2. KOSEI is also guaranteed to search the neighbors of discriminatory data only, which is not guaranteed in AEQUITAS.

The second is the limitation of wasting search resources to evaluate duplicated data. The local search of KOSEI carefully avoids evaluating duplicated data (as clarified in line 11 in Algorithm 3). Moreover, KOSEI is implemented without using the basin-hopping optimization function, which is heavily used in the AEQUITAS implementation. The optimization function's implementation demands a large amount of execution time since it evaluates many data evaluated in previous iterations, resulting in the inefficiency of detecting discriminatory data.

IV. EVALUATION

We conducted experiments to evaluate KOSEI. For the evaluation, we pose the two research questions (RQs).

- RQ1 Does KOSEI find more discriminatory data than AEQUITAS, at a reasonable execution cost?
- RQ2 Does KOSEI generate test cases likely to be discriminatory and avoid duplicated data, better than AEQUITAS?

RQ1 is the main RQ since the goal of this paper is to improve AEQUITAS. RQ2 investigates if and how much our improvement on local search improves the limitations of AEQUITAS, as we discussed in Section III-A.

TABLE I
COMPARISON OF KOSEI AND AEQUITAS (RQ1)

	Dataset	Classifier	#Discriminatory data			Time (s)			#Test cases		Precision (%)		Duplicated evaluation (%)
			Aequitas	KOSEI	Pct (%)	Aequitas	KOSEI	Pct (%)	Aequitas	KOSEI	Aequitas	KOSEI	Aequitas
1	Census Income	DT	2,147.5	**25,434.9**	1,184.4	57.6	**5.8**	10.0	26,970.5	35,205.3	8.1	**71.9**	95.01
2		MLPC	8,110.2	**369,629.5**	4,557.6	6,105.9	**609.4**	10.0	26,279	515,406	33.6	**71.7**	99.76
3		RF	7,506.9	**231,275.2**	3,080.8	4,465.1	**357.9**	8.0	48,838.8	333,604.5	16.6	**69.4**	99.06
4	Statlog	DT	2,135.6	**34,031.9**	1,593.6	109.4	**9.9**	9.0	23,986.9	42,405.2	9.2	**80.3**	97.75
5		MLPC	1,498.2	**101,092.7**	6,747.6	1,371.6	**52.3**	3.8	17,522.9	316,411	9.0	**32.0**	99.84
6		RF	8,727.8	**379,656.3**	4,350.0	14,442.0	**820.8**	5.7	25,202.4	707,607.1	34.6	**53.6**	99.85
7	Bank Marketing	DT	8,374.2	**206,718.3**	2,468.5	440.2	**36.5**	8.3	79,605.3	276,798.6	11.8	**74.7**	98.68
8		MLPC	1,654.8	**187,076.4**	11,305.1	746.9	**53.8**	7.2	8,001.3	302,790.9	21.4	**61.8**	99.88
9		RF	7,481.0	**783,700.6**	10,475.9	18,380.9	**1,010.1**	5.5	16,273.9	1,209,985.7	46.1	**64.8**	99.94
	Average				5,084.8			7.5			21.2	64.5	98.86

A. Experimental environment and settings

We implemented KOSEI with Python 2.7.18, extending AEQUITAS and using the scikit-learn library [8]. The code for KOSEI is available at: "https://github.com/sskeiouk/KOSEI".

For a fair comparison, we use the same settings used in [2] [6], as follows: three datasets (Census Income[3], Statlog[4], Bank Marketing[5]), three classifiers (Decision Tree (DT), MLPC, Random Forest (RF)), and protected attributes ('gender' for Census Income and Statlog, and 'age' for Bank Marketing).

For AEQUITAS, we use the fully-directed search variant since it is shown that the variant performs best among the three variants [2]. The iteration limits for global and local search (i. e., global and local iteration limit) are set to 2000, which are also the settings used in [2]. We ran ten executions for each configuration of experiments and took their average. All experiments were executed on a laptop machine with Apple M1, 16GB of RAM, running macOS Big Sur 11.4.

B. [RQ1] Does KOSEI find more discriminatory data than AEQUITAS, at a reasonable execution cost?

Table I shows the results of experiments to compare KOSEI and AEQUITAS. The rows represent configurations of datasets and classifiers. The columns represent the number of detected discriminatory data ('#Discriminatory data') and execution time ('Time'). The result shows that KOSEI's performance was orders of magnitude better than AEQUITAS. KOSEI detects more discriminatory data than AEQUITAS, by 5084.8% on average, for all nine configurations, and up to 11305.1% (for the eighth configuration).

The results also show that KOSEI requires less execution time than AEQUITAS. KOSEI runs faster than AEQUITAS, by 13.3 times on average, and KOSEI is faster by up to 26.3 times (for the fifth configuration).

Answer for RQ1

Yes. The discrimination detecting ability of KOSEI is 5084.8% stronger than AEQUITAS. Also, KOSEI runs 13.3 times faster than AEQUITAS.

[3]https://archive.ics.uci.edu/ml/datasets/adult

[4]https://archive.ics.uci.edu/ml/datasets/statlog+(german+credit+data)

[5]https://archive.ics.uci.edu/ml/datasets/bank+marketing

C. [RQ2] Does KOSEI generate test cases likely to be discriminatory and avoid duplicated data, better than AEQUITAS?

We measure the number of test cases generated by KOSEI and AEQUITAS and calculate hit ratios (i. e., precisions) of discriminatory data over generated test cases. The '#Test cases' and 'precision (%)' columns in Table I respectively show them. As stated in [3], [4], AEQUITAS generates and evaluates many duplicated data, which may cause inefficiency of the technique. Therefore, we measured the ratio of duplicated data over all the data evaluated in AEQUITAS, shown in the 'Duplicated evaluation (%)' column.

The results confirm that the precisions of KOSEI are higher than those of AEQUITAS by 3.04 (= 64.5/21.2) times on average. From the 'Duplicated evaluation (%)' column, we also observe that 98.8 % of evaluated data are duplicated. Note that KOSEI avoided evaluations of duplicated data due to the algorithm design (line 11 in Algorithm 3).

Answer for RQ2

Yes. KOSEI generates test cases more likely to be discriminatory than AEQUITAS, by 3.04 times on average. In addition, in AEQUITAS, 98.8 % of evaluated data are duplicated, while in KOSEI, all duplicated evaluations are avoided.

V. THREATS TO VALIDITY

This section discusses the main validity threats of our study.

a) Datasets, classifiers, and protected attributes used in experiments: Our evaluation experiments use the same settings used in [2] [6]: three datasets (Census, Statlog, and Bank), three classifiers (DT, MLPC, RF), and one protected attribute ('gender', 'age'). As with any experiments, the number of configurations is a threat, so additional experiments with more datasets, classifiers, and different protected attributes (e.g., 'Race') would strengthen the generalization. Note, however, that it is not easy to consider all configurations in experiments since the number of configurations increases exponentially due to combinations. Experiments in other fairness testing studies thus also pick several datasets, classifiers, and protected attributes instead of thorough configurations.

b) Comparison with other techniques: Our study evaluates KOSEI by comparing it against only AEQUITAS; that is, we did not compare it with other black-box individual fairness testing techniques, such as SG [4] and VBT [5]. The focus of the study is on improving AEQUITAS, and thus evaluation experiments were designed accordingly. Also, fair comparison with SG and VBT is not easy since their implementation details are not equivalent; e. g., KOSEI (based on AEQUITAS) uses Scikit-Learn, while VBT and SG use TensorFlow for ML library.

c) Construct validity: We implemented KOSEI by extending the AEQUITAS code, obtained from "https://github.com/sakshiudeshi/Aequitas". Despite our best efforts to pursue the code quality of KOSEI, it cannot be guaranteed that the code is free of bugs, as always. Therefore, we make the KOSEI code available online so that anyone can inspect its validity.

VI. Related work

Galhotra et al. [1] developed an individual fairness testing technique based on this abstract concept of individual fairness introduced by Dwork et al. [9]. They realize it by treating two individuals as similar if they are identical except for protected attributes. This simple treatment of individual similarity has been widely accepted, as most studies on individual fairness testing (discussed below) are based on this concept of individual similarity. This work also proposed algorithms for the individual fairness testing, called THEMIS, which are basically based on simple random testing.

Aggarwal et al. [4] proposed an individual fairness testing technique, called SG, that uses a symbolic execution technique [10]. Symbolic execution was initially developed for program analysis to systematically search the input space in order to cover input space efficiently. SG applies this technique to individual fairness testing to gain its efficacy. It is also noteworthy that SG also structures its algorithm with the global and local search phases, inspired by AEQUITAS.

Morales et al. [6] proposed an individual fairness testing technique named CGFT, which focused on the global search of AEQUITAS. While AEQUITAS used random testing in the global search, CGFT proposed using combinatorial testing (CT) [11], which has a diverse sampling ability, resulting in an improvement in the discrimination detecting ability.

Sharma and Wehrheim [5] proposed Verification-Based Testing (VBT). Its key idea is to detect discriminatory data by encoding the property of individual fairness and the approximated ML classifier under test into a logical formula and solving the encoded constraint with a constraint solver.

While the techniques mentioned above (i. e., [1], [2], [4]–[6]) are all featured with a black-box testing approach, Zhang et al. [3] proposed a white-box approach to individual fairness testing. Specifically, it targets ML classifiers based on Deep Neural Networks (DNN). Their algorithm, called ADF, is inspired by a gradient technique to detect adversarial examples. However, this technique differs from ours in that it is only applicable to DNN based ML classifiers.

VII. Conclusion and Future Work

This paper proposed a black-box individual fairness testing technique called KOSEI. KOSEI is based on AEQUITAS and is realized by improving the local search of AEQUITAS based on its limitations identified by our insight. Experiments show that the performance gain of KOSEI compared with AEQUITAS is orders of magnitude; KOSEI, on average, detects 50.8 times more discriminations than AEQUITAS. Experiments also show that the performance gain is due to our technical improvement based on our insight.

There are many directions to extend this work. The first is to extend experiments with more datasets, more classifiers, and different kinds of protected attributes (e. g., race) for further generalization. The second direction is to evaluate KOSEI, comparing it with other techniques, such as SG [4] and VBT [5]. Another direction is to evaluate the use of detected discriminatory data for re-training classifiers to improve its fairness, as attempted in [2]. Finally, we also plan to combine the improvement of the local search of KOSEI with other techniques, such as with CGFT [6], which improves the global search of AEQUITAS.

Acknowledgement

This paper is based on results obtained from a project, JPNP20006, commissioned by the New Energy and Industrial Technology Development Organization (NEDO).

References

[1] S. Galhotra, Y. Brun, and A. Meliou, "Fairness testing: testing software for discrimination," in *Proceedings of ESEC/FSE'17*, 2017, pp. 498–510.
[2] S. Udeshi, P. Arora, and S. Chattopadhyay, "Automated directed fairness testing," in *Proceedings of ASE'18*, 2018, pp. 98–108.
[3] P. Zhang, J. Wang, J. Sun, G. Dong, X. Wang, X. Wang, J. S. Dong, and T. Dai, "White-box fairness testing through adversarial sampling," in *Proceedings of ICSE'20*, 2020, pp. 949–960.
[4] A. Aggarwal, P. Lohia, S. Nagar, K. Dey, and D. Saha, "Black box fairness testing of machine learning models," in *Proceedings of ESEC/SIGSOFT FSE'19*, 2019, pp. 625–635.
[5] A. Sharma and H. Wehrheim, "Automatic fairness testing of machine learning models," in *Proceedings of ICTSS'20*, 2020, pp. 255–271.
[6] D. P. Morales, T. Kitamura, and S. Takada, "Coverage-guided fairness testing," in *Proceesing of ICIS'21*, 2021, pp. 183–199.
[7] D. Dua and C. Graff, "UCI machine learning repository," 2017. [Online]. Available: http://archive.ics.uci.edu/ml
[8] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.
[9] C. Dwork, M. Hardt, T. Pitassi, O. Reingold, and R. Zemel, "Fairness through awareness," in *Proceedings of ITCS'12*, 2012, pp. 214–226.
[10] K. Sen, D. Marinov, and G. Agha, "CUTE: a concolic unit testing engine for C," in *Proceedings of FSE'05*, 2005, pp. 263–272.
[11] R. Kuhn and R. Kacker, *Introduction to Combinatorial Testing*. Chapman & HallCRC, 2013.