Algorithm 1: Single Point 2-sample Fast HHG Test

input : (1) category matrix X of size N where $x_i \in \{1, 2....K\}$, (2) observation matrix $Y \in \mathbb{R}^q$ of size N, (3) distance metric

output: test statistic T and p-value P

function T, P = SINGLEPOINTTEST(X, Y)
    $z_y \leftarrow \frac{\sum_{i=1}^{N} y_i}{N}$;   /* z_y = center point of Y $\in \mathbb{R}^q$ */
    $D_y \leftarrow$ POINTDISTANCE$(Y, z_y, metric)$
    /* D_y = 1D collection of distances of size N */
    $T, P \leftarrow$ minP(X, D_y)
    OR
    $T, P \leftarrow$ ADTEST(X, D_y)
end

Algorithm 2: Point Distance Function

input : (1) category matrix $X \in \mathbb{R}^p$ of size N, $Y \in \mathbb{R}^q$ of size M (2) single point $z_x \in \mathbb{R}^p$, $z_y \in \mathbb{R}^q$ (3) distance metric

output: 1D collection of distances D_y of size N

function $D_x, D_y \leftarrow$ POINTDISTANCE$(Y, z_y, metric)$
    for i *in 1:N* **do**
        $D_y[i] \leftarrow D(Y_i, z_y)$; /* where D = valid distance metric */
    end
end

Algorithm 3: minP Test

input : (1) category matrix X of length N where $x_i \in \{1, 2....K\}$, (2) 1D collection of distances D_y of length N

output: test statistic T and p-value P

function T, P = `minP(`X, D_y`)`

  $Y_{ranked} = \text{rank}(D_y)$ **for** *m in 2:N* **do** `/* m = partition size */`

    Π_m = the set of all possible partitions of (X, Y_ranked) into m cells

    for *each partition L in Π_m* **do**

      for *each cell C in partition L* **do**

        for *each category $g \in \{1, 2....K\}$* **do**

          $e_c(g) \leftarrow \text{len}(C) * \frac{N_g}{N}$

          $o_c(g) \leftarrow$ number of observations of category g in C

        end

        $t_c \leftarrow \sum_{g=1}^{K} o_c(g) * log(\frac{o_c(g)}{e_c(g)}$

      end

      $T^L \leftarrow \sum_{allCinL} t_c$

    end

    $S_m \leftarrow \sum_{L \in \Pi_m} T^L$ $P_m \leftarrow$ `minPVALUE(`S_m, m, Y_ranked, X`)`

  end

  $T \leftarrow \min_{m \in [2,...N]} P_m$ $P \leftarrow$ p-value based on null distribution of test statistics of fixed m

end

Algorithm 4: minP P-Value

input : (1) S_m = aggregated per-partition test statistic for a given partition size m, (2) partition size m, (3) sample ranks Y_{ranked} of length N, (4) category matrix X of length N, (5) number of runs B

output: P_m = p-value of given S_m

```
function P_m = minP(S_m, m, Y_ranked, X)
    for b in 1:B do
        XY_null ← random reassignment of Y_ranked to X
        Π_null ← the set of all possible partitions of XY_null into m cells
        for each partition L in Π_null do
            for each cell C in partition L do
                for each category g ∈ {1, 2....K} do
                    e_c(g) ← len(C) * N_g / N
                    o_c(g) ← number of observations of category g in C
                end
                t_c ← Σ_{g=1}^{K} o_c(g) * log(o_c(g) / e_c(g))
            end
            T^L ← Σ_{allCinL} t_c
        end
        S_b ← Σ_{L∈Π_null} T^L
    end
    P_m = Σ(S_b ≥ S_m) / (B+1)
end
```

Algorithm 5: Anderson-Darling K-Sample Test

input : (1) category matrix X of length N where $x_i \in \{1, 2....K\}$, (2) 1D collection of distances D_y of length N

output: test statistic T and p-value P

```
function T, P = ADTEST(X, D_y)
    T ← test-statistic from scipy.stats.anderson_ksamp((X, D_y))
    P ← significance level from scipy.stats.anderson_ksamp((X, D_y))
end
```