Algorithm 3 `Facilities_generator (nb_locations, seed, min_demand, max_demand)`

country_codes ← [“EU”, “CHN”, “IND”, “ZAF”, “USA”, “TUR”, “IRN”]
Shapefile ← “TM_WORLD_BORDERS-0.3.shp”
Boundaries ← filtered_locations_in_Shapefile
Facilities ← empty_list
locations ← empty_list
for each i in country_codes **do**
    Boundaries.add(Shapefile[i]) ▷ Getting country borders.
end for
while |locations| < nb_locations **do**
    location ← $(U(-180, 180), U(-90, 90))$
    if location is in Boundaries **then**
        locations.add(location)
        Boundaries.remove(location)
    end if
end while
for each i in locations **do**
    TTR[i] ← $U(2, 11)$
    SI[i] ← $U(1, 11)$
    CAP[i] ← $U(\text{min\_demand} \times 5, \text{max\_demand} \times 10)$
    lon[i] ← locations[i].longitude
    lat[i] ← locations[i].latitude
    Facilities.add(TTR[i], SI[i], CAP[i], lon[i], lat[i]) ▷ Assigning facility attributes.
end for
for each i in facilities **do** ▷ Haversine distance computation
    for each j in facilities where index[j] ¡ index[i] **do**
        lon1, lat1, lon2, lat2 ← radians(lon[i], lat[i], lon[j], lat[j])
        dlon ← lon2 - lon1 ▷ Difference in longitude between the two points.
        dlat ← lat2 - lat1 ▷ Difference in latitude between the two points.
        a ← sin(dlat / 2)^2 + cos(lat1) × cos(lat2) × sin(dlon / 2)^2 ▷ Haversine formula’s component which calculates the square of half the chord length between the points.
        c ← 2× atan2($\sqrt{a}$, $\sqrt{1-a}$) ▷ Angular distance in radians between the two points on the sphere.
        r ← 6371 ▷ Radius of Earth in kilometers.
        distance[i][j] ← r × c
        distance[j][i] ← distance[i][j] ▷ Distance between facilities.
        TGHG[i][j] ← distance[i][j] × 1.05 ▷ Multiplied with 1.05 kg/km for trucks. (Source: 8billiontrees)
        TGHG[j][i] ← TGHG[i][j]
    end for
end for
return Facilities