```python
import global_random_search
import lib
import numpy as np
import sgd
import matplotlib.pyplot as plt
import pandas as pd
import time
import cifar_costf
import json
import argparse
import c_vis
import even_samples
import math
import cps
ps = cps.ps

ap = argparse.ArgumentParser()
# ap.add_argument("--exp", type=str, required=True)
ap.add_argument("--M", type=int, required=True)
ap.add_argument("--N", type=int, required=True)
ap.add_argument("--n", type=int, required=True)
ap.add_argument("--iterations", type=int, required=True)
args = ap.parse_args()

f = {
    "function": lib.f_real,
    "gradient": lib.f_grad,
    "dname": "$f(x)$",
    "name": "f",
    "alpha": 0.0065,
}

g = {
    "function": lib.g_real,
    "gradient": lib.g_grad,
    "dname": "$g(x)$",
    "name": "g",
    "alpha": 0.003,
}


def gradient_descent_constant(step_size=0.0065, start=[0, 0], funcs=f, max_time=1):
    start = np.array(start)
    g = sgd.StochasticGradientDescent()
    g.step_size(step_size)
    g.start(start)
    def function_generator():
        while True:
            yield funcs["function"], funcs["gradient"]
    g.function_generator(function_generator())
    g.debug(True)
    g.alg("constant")
    start_time = time.time()
    current_time = 0
    while current_time < max_time:
        current_time = time.time() - start_time
        g.step()
        yield {
                "f(x)": g._function(g._x_value),
                "x": g._x_value,
                "time": time.time() - start_time,
        }

if __name__ == "__main__":
    train, test = even_samples.even_sample_categories(math.floor(args.n))

    def costf(x):
        return cifar_costf.costf(x, train, test)

    grs = global_random_search.b_mod(
        debug=True,
        costf=costf, parameters=ps, N=args.N, M=args.M, iterations=args.iterations)

    fname = f"data/c-b_mod-N{args.N}-M{args.M}-n{args.n}-it{args.iterations}.json"
    save = {
        'results': grs,
        'param-limits': ps,
        'args': vars(args),
        'name': None,
    }
    with open(fname, "w") as f:
        json.dump(save, f)
```