```python
import global_random_search
import lib
import numpy as np
import sgd
import matplotlib.pyplot as plt
from matplotlib.lines import Line2D
import pandas as pd
import time
import json

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.perf_counter()
    current_time = 0
    while current_time < max_time:
        current_time = time.perf_counter() - start_time
        g.step()
        yield {
                "f(x)": g._function(g._x_value),
                "x": g._x_value,
                "time": time.perf_counter() - start_time,
        }


custom_lines = [
        Line2D([0], [0], color='purple', lw=2),
        Line2D([0], [0], color='blue', lw=2),
        Line2D([0], [0], color='orange', lw=2),
        Line2D([0], [0], color='black', lw=2),
        ]
custom_labels = ['rnd search b_mod', 'rnd search b', 'rnd search a', 'gradient descent']

def thin(array, step = 30):
    return [array[i] for i in range(0, len(array), step)]

def vis_results(results):
    def f(x, y):
        return 3 * (x - 5)**4 + 10 * (y - 9)**2
    def g(x, y):
        return np.maximum(x - 5, 0) + 10 * np.abs(y - 9)

    x = np.linspace(0, 10, 400)
    y = np.linspace(0, 18, 400)
    X, Y = np.meshgrid(x, y)
    Z_f = f(X, Y)
    Z_g = g(X, Y)

    fig = plt.figure(figsize=(12, 6))

    axf = fig.add_subplot(1, 2, 1)
    axf.contourf(X, Y, Z_f, levels=30, cmap='viridis')
    axf.set_title('$f(x, y)$')
    axf.set_xlabel('$x$')
    axf.set_ylabel('$y$')

    axg = fig.add_subplot(1, 2, 2)
    axg.contourf(X, Y, Z_g, levels=30, cmap='viridis')
    axg.set_title('$g(x, y)$')
    axg.set_xlabel('$x$')
    axg.set_ylabel('$y$')

    cmap = plt.cm.Oranges
    for a_results in results['f']['a']:
        x_coords = thin([point[0] for point in a_results['stats']['it_best_params']])
        y_coords = thin([point[1] for point in a_results['stats']['it_best_params']])
        color = [cmap(i / len(x_coords)) for i in range(len(x_coords))]
        axf.scatter(x_coords, y_coords, linestyle='-', label="rndsearch a", color=color)
    for a_results in results['g']['a']:
        x_coords = thin([point[0] for point in a_results['stats']['it_best_params']])
        y_coords = thin([point[1] for point in a_results['stats']['it_best_params']])
        color = [cmap(i / len(x_coords)) for i in range(len(x_coords))]
        axg.scatter(x_coords, y_coords, linestyle='-', label="rndsearch a", color=color)
    plt.tight_layout()
    plt.savefig("fig/bii-contours-a.pdf")
```

```
101:
102:     x = np.linspace(0, 10, 400)
103:     y = np.linspace(0, 18, 400)
104:     X, Y = np.meshgrid(x, y)
105:     Z_f = f(X, Y)
106:     Z_g = g(X, Y)
107:
108:     fig = plt.figure(figsize=(12, 6))
109:
110:     axf = fig.add_subplot(1, 2, 1)
111:     axf.contourf(X, Y, Z_f, levels=30, cmap='viridis')
112:     axf.set_title('$f(x, y)$')
113:     axf.set_xlabel('$x$')
114:     axf.set_ylabel('$y$')
115:
116:     axg = fig.add_subplot(1, 2, 2)
117:     axg.contourf(X, Y, Z_g, levels=30, cmap='viridis')
118:     axg.set_title('$g(x, y)$')
119:     axg.set_xlabel('$x$')
120:     axg.set_ylabel('$y$')
121:
122:     cmap = plt.cm.Blues
123:     for b_results in results['f']['b']:
124:         x_coords = thin([point[0] for point in b_results['stats']['it_best_params']])
125:         y_coords = thin([point[1] for point in b_results['stats']['it_best_params']])
126:         color = [cmap(i / len(x_coords)) for i in range(len(x_coords))]
127:         axf.plot(x_coords, y_coords, linestyle='-', label="rndsearch b", color=color)
128:     for b_results in results['g']['b']:
129:         x_coords = thin([point[0] for point in b_results['stats']['it_best_params']])
130:         y_coords = thin([point[1] for point in b_results['stats']['it_best_params']])
131:         color = [cmap(i / len(x_coords)) for i in range(len(x_coords))]
132:         axg.plot(x_coords, y_coords, linestyle='-', label="rndsearch b", color=color)
133:     plt.tight_layout()
134:     plt.savefig("fig/bii-contours-b.pdf")
135:
136:     # axf.legend(custom_lines[1:3], custom_labels[1:3])
137:     # axg.legend(custom_lines[1:3], custom_labels[1:3])
138:
139:     return axf, axg
140:
141: max_time=0.1
142: if __name__ == "__main__":
143:     all_results = {}
144:     for funcs in f, g:
145:         res = list(gradient_descent_constant(max_time=max_time, funcs=funcs, step_size=funcs["alpha"]))
146:
147:         plt.figure()
148:         results = {
149:                 "b_mod": [],
150:                 "b": [],
151:                 "a": [],
152:         }
153:         res = pd.DataFrame(res)
154:
155:         for i in range(5):
156:             # ps = [{"min": 0, "max": 10}, {"min": 0, "max": 18}]
157:             # grs = global_random_search.b_mod(
158:             #     costf=funcs["function"], iterations=100, parameters=ps, N=1000, M=100, max_time=max_time)
159:             # costs = grs['stats']['it_best_costs']
160:             # plt.plot(grs['stats']['time'], costs, label="rnd search b_mod")
161:             # print(funcs["name"], "total iterations global random search b_mod: ", len(grs['stats']['time']))
162:             ps = [{"min": 0, "max": 10}, {"min": 0, "max": 18}]
163:             grs = global_random_search.b_mod(
164:                 costf=funcs["function"], iterations=8, parameters=ps, N=100, M=50)
165:             costs = grs['stats']['it_best_costs']
166:             plt.plot(grs['stats']['time'], costs, label="rnd search b_mod", color="purple")
167:             results["b_mod"].append(grs)
168:             print(funcs["name"], "total iterations global random search b_mod: ", len(grs['stats']['time']))
169:
170:             ps = [{"min": 0, "max": 10}, {"min": 0, "max": 18}]
171:             grs = global_random_search.b(
172:                 costf=funcs["function"], iterations=8, parameters=ps, perturb_pc=0.01, N=100, M=50)
173:             costs = grs['stats']['it_best_costs']
174:             plt.plot(grs['stats']['time'], costs, label="rnd search b", color="blue")
175:             results["b"].append(grs)
176:             print(funcs["name"], "total iterations global random search b: ", len(grs['stats']['time']))
177:
178:             ps = [{"min": 0, "max": 10}, {"min": 0, "max": 18}]
179:             grs = global_random_search.a(
180:                 costf=funcs["function"], parameters=ps, N=300)
181:             costs = grs['stats']['it_best_costs']
182:             plt.plot(grs['stats']['time'], costs, label="rnd search a", color="orange")
183:             results["a"].append(grs)
184:             print(funcs["name"], "total iterations global random search a: ", len(grs['stats']['time']))
185:
186:
187:         plt.plot(res["time"], res["f(x)"], label="gradient descent", color="black")
188:         plt.title(f"Global Random Search vs Gradient Descent on {funcs['dname']}")
189:         plt.legend(custom_lines, custom_labels, loc='lower right')
190:         plt.yscale('log')
191:         plt.xlabel("time (seconds)")
192:         plt.ylabel(funcs['dname'])
193:         plt.tight_layout()
194:         print(funcs["name"], "total iterations gradient descent: ", len(res))
195:
196:         all_results[funcs['name']] = results
197:
198:     with open("data/bii-time.json", "w") as f:
199:         json.dump(all_results, f)
```