```
  1: import global_random_search
  2: import lib
  3: import numpy as np
  4: import sgd
  5: import matplotlib.pyplot as plt
  6: from matplotlib.lines import Line2D
  7: import pandas as pd
  8: import time
  9: import json
 10:
 11: f = {
 12:     "function": lib.f_real,
 13:     "gradient": lib.f_grad,
 14:     "dname": "$f(x)$",
 15:     "name": "f",
 16:     "alpha": 0.0065,
 17: }
 18:
 19: g = {
 20:     "function": lib.g_real,
 21:     "gradient": lib.g_grad,
 22:     "dname": "$g(x)$",
 23:     "name": "g",
 24:     "alpha": 0.003,
 25: }
 26:
 27:
 28: def gradient_descent_constant(step_size=0.0065, start=[0, 0], funcs=f, max_time=1):
 29:     start = np.array(start)
 30:     g = sgd.StochasticGradientDescent()
 31:     g.step_size(step_size)
 32:     g.start(start)
 33:     def function_generator():
 34:         while True:
 35:             yield funcs["function"], funcs["gradient"]
 36:     g.function_generator(function_generator())
 37:     g.debug(True)
 38:     g.alg("constant")
 39:     start_time = time.perf_counter()
 40:     current_time = 0
 41:     while current_time < max_time:
 42:         current_time = time.perf_counter() - start_time
 43:         g.step()
 44:         yield {
 45:                 "f(x)": g._function(g._x_value),
 46:                 "x": g._x_value,
 47:                 "time": time.perf_counter() - start_time,
 48:         }
 49:
 50:
 51: custom_lines = [
 52:         Line2D([0], [0], color='purple', lw=2),
 53:         Line2D([0], [0], color='blue', lw=2),
 54:         Line2D([0], [0], color='orange', lw=2),
 55:         Line2D([0], [0], color='black', lw=2),
 56:         ]
 57: custom_labels = ['rnd search b_mod', 'rnd search b', 'rnd search a', 'gradient descent']
 58:
 59: def thin(array, step = 5):
 60:     return [array[i] for i in range(0, len(array), step)]
 61:
 62: def vis_results(results):
 63:     print("starting vis")
 64:     def f(x, y):
 65:         return 3 * (x - 5)**4 + 10 * (y - 9)**2
 66:     def g(x, y):
 67:         return np.maximum(x - 5, 0) + 10 * np.abs(y - 9)
 68:
 69:     x = np.linspace(0, 10, 400)
 70:     y = np.linspace(0, 18, 400)
 71:     X, Y = np.meshgrid(x, y)
 72:     Z_f = f(X, Y)
 73:     Z_g = g(X, Y)
 74:
 75:     fig = plt.figure(figsize=(12, 6))
 76:
 77:     axf = fig.add_subplot(1, 2, 1)
 78:     axf.contourf(X, Y, Z_f, levels=30, cmap='viridis')
 79:     axf.set_title('$f(x, y)$')
 80:     axf.set_xlabel('$x$')
 81:     axf.set_ylabel('$y$')
 82:
 83:     axg = fig.add_subplot(1, 2, 2)
 84:     axg.contourf(X, Y, Z_g, levels=30, cmap='viridis')
 85:     axg.set_title('$g(x, y)$')
 86:     axg.set_xlabel('$x$')
 87:     axg.set_ylabel('$y$')
 88:
 89:     cmap = plt.cm.Oranges
 90:     for a_results in results['f']['a'][:1]:
 91:         x_coords, y_coords = zip(*thin(a_results['stats']['it_best_params']))
 92:         # y_coords = thin([point[1] for point in a_results['stats']['it_best_params']])
 93:         color = [cmap(i / len(x_coords)) for i in range(len(x_coords))]
 94:         for x,y,c in zip(x_coords, y_coords, color):
 95:             axf.scatter(x, y, color=c)
 96:     for a_results in results['g']['a'][:1]:
 97:         x_coords = thin([point[0] for point in a_results['stats']['it_best_params']])
 98:         y_coords = thin([point[1] for point in a_results['stats']['it_best_params']])
 99:         color = [cmap(i / len(x_coords)) for i in range(len(x_coords))]
100:         for x,y,c in zip(x_coords, y_coords, color):
```

```
101:             axf.scatter(x, y, color=c)
102:     plt.tight_layout()
103:     plt.savefig("fig/bii-contours-a.pdf")
104:
105:     x = np.linspace(0, 10, 400)
106:     y = np.linspace(0, 18, 400)
107:     X, Y = np.meshgrid(x, y)
108:     Z_f = f(X, Y)
109:     Z_g = g(X, Y)
110:
111:     fig = plt.figure(figsize=(12, 6))
112:
113:     axf = fig.add_subplot(1, 2, 1)
114:     axf.contourf(X, Y, Z_f, levels=30, cmap='viridis')
115:     axf.set_title('$f(x, y)$')
116:     axf.set_xlabel('$x$')
117:     axf.set_ylabel('$y$')
118:
119:     axg = fig.add_subplot(1, 2, 2)
120:     axg.contourf(X, Y, Z_g, levels=30, cmap='viridis')
121:     axg.set_title('$g(x, y)$')
122:     axg.set_xlabel('$x$')
123:     axg.set_ylabel('$y$')
124:
125:     cmap = plt.cm.Blues
126:     for b_results in results['f']['b'][:1]:
127:         x_coords = thin([point[0] for point in b_results['stats']['it_best_params']])
128:         y_coords = thin([point[1] for point in b_results['stats']['it_best_params']])
129:         color = [cmap(i / len(x_coords)) for i in range(len(x_coords))]
130:         for x,y,c in zip(x_coords, y_coords, color):
131:             axf.scatter(x, y, color=c)
132:     for b_results in results['g']['b'][:1]:
133:         x_coords = thin([point[0] for point in b_results['stats']['it_best_params']])
134:         y_coords = thin([point[1] for point in b_results['stats']['it_best_params']])
135:         color = [cmap(i / len(x_coords)) for i in range(len(x_coords))]
136:         for x,y,c in zip(x_coords, y_coords, color):
137:             axf.scatter(x, y, color=c)
138:     plt.tight_layout()
139:     plt.savefig("fig/bii-contours-b.pdf")
140:
141:     # axf.legend(custom_lines[1:3], custom_labels[1:3])
142:     # axg.legend(custom_lines[1:3], custom_labels[1:3])
143:
144:     return axf, axg
145:
146:
147: if __name__ == "__main__":
148:     results = None
149:     with open("data/bii-time.json", "r") as f:
150:         results = json.load(f)
151:     vis_results(results)
```