

```
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=0.4):
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 = 30):
60:     return [array[i] for i in range(0, len(array), step)]
61:
62: def vis_results(results):
63:     def f(x, y):
64:         return 3 * (x - 5)**4 + 10 * (y - 9)**2
65:     def g(x, y):
66:         return np.maximum(x - 5, 0) + 10 * np.abs(y - 9)
67:
68:     x = np.linspace(0, 10, 400)
69:     y = np.linspace(0, 18, 400)
70:     X, Y = np.meshgrid(x, y)
71:     Z_f = f(X, Y)
72:     Z_g = g(X, Y)
73:
74:     fig = plt.figure(figsize=(12, 6))
75:
76:     axf = fig.add_subplot(1, 2, 1)
77:     axf.contourf(X, Y, Z_f, levels=30, cmap='viridis')
78:     axf.set_title('$f(x, y)$')
79:     axf.set_xlabel('$x$')
80:     axf.set_ylabel('$y$')
81:
82:     axg = fig.add_subplot(1, 2, 2)
83:     axg.contourf(X, Y, Z_g, levels=30, cmap='viridis')
84:     axg.set_title('$g(x, y)$')
85:     axg.set_xlabel('$x$')
86:     axg.set_ylabel('$y$')
87:
88:     cmap = plt.cm.Oranges
89:     for a_results in results['f']['a']:
90:         x_coords = thin([point[0] for point in a_results['stats']['it_best_params']])
91:         y_coords = thin([point[1] for point in a_results['stats']['it_best_params']])
92:         color = [cmap(i / len(x_coords)) for i in range(len(x_coords))]
93:         axf.scatter(x_coords, y_coords, linestyle='-', label="rndsearch a", color=color)
94:     for a_results in results['g']['a']:
95:         x_coords = thin([point[0] for point in a_results['stats']['it_best_params']])
96:         y_coords = thin([point[1] for point in a_results['stats']['it_best_params']])
97:         color = [cmap(i / len(x_coords)) for i in range(len(x_coords))]
98:         axg.scatter(x_coords, y_coords, linestyle='-', label="rndsearch a", color=color)
99:     plt.tight_layout()
100:     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=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=200, parameters=ps, N=400, M=100, max_time=max_time)
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=200, parameters=ps, perturb_pc=0.001, N=400, M=100, max_time=max_time)
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, max_time=max_time)
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:         plt.savefig(f"fig/bii-time-{funcs['name']}.pdf")
195:         print(funcs["name"], "total iterations gradient descent: ", len(res))
196:
197:         all_results[funcs['name']] = results
198:
199:     # with open("data/bii-time.json", "w") as f:
200:     #     json.dump(all_results, f)

```

src/bii-time.py Wed Apr 10 21:50:57 2024 3

201: