Example to plot directly into latex

19-10-2019

1 Introduction

2 Genetic Algorithm Performance

To illustrate how the python code exports the figures directly into the report, this second "hw2" is included. Below are the pictures that are created by the code listed in Appendix A and Appendix B.

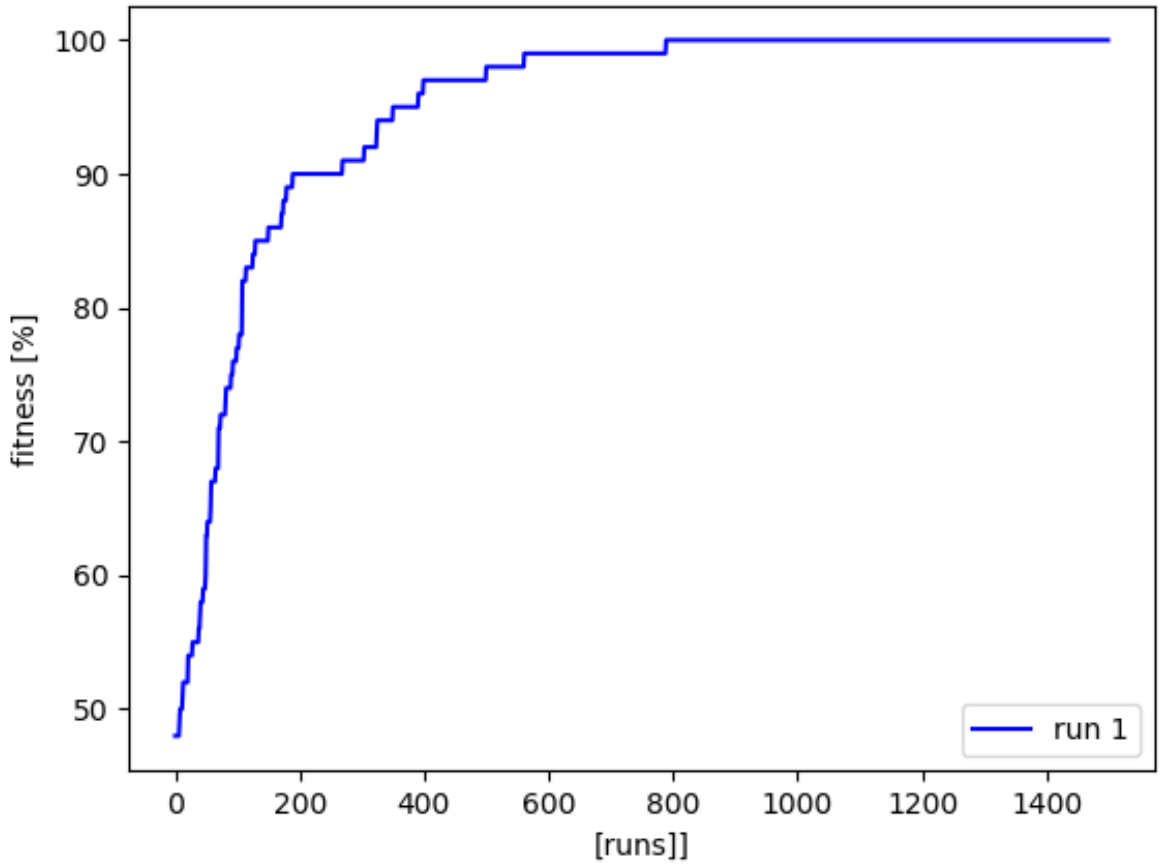

Figure 1: Performance of some genetic algorithm

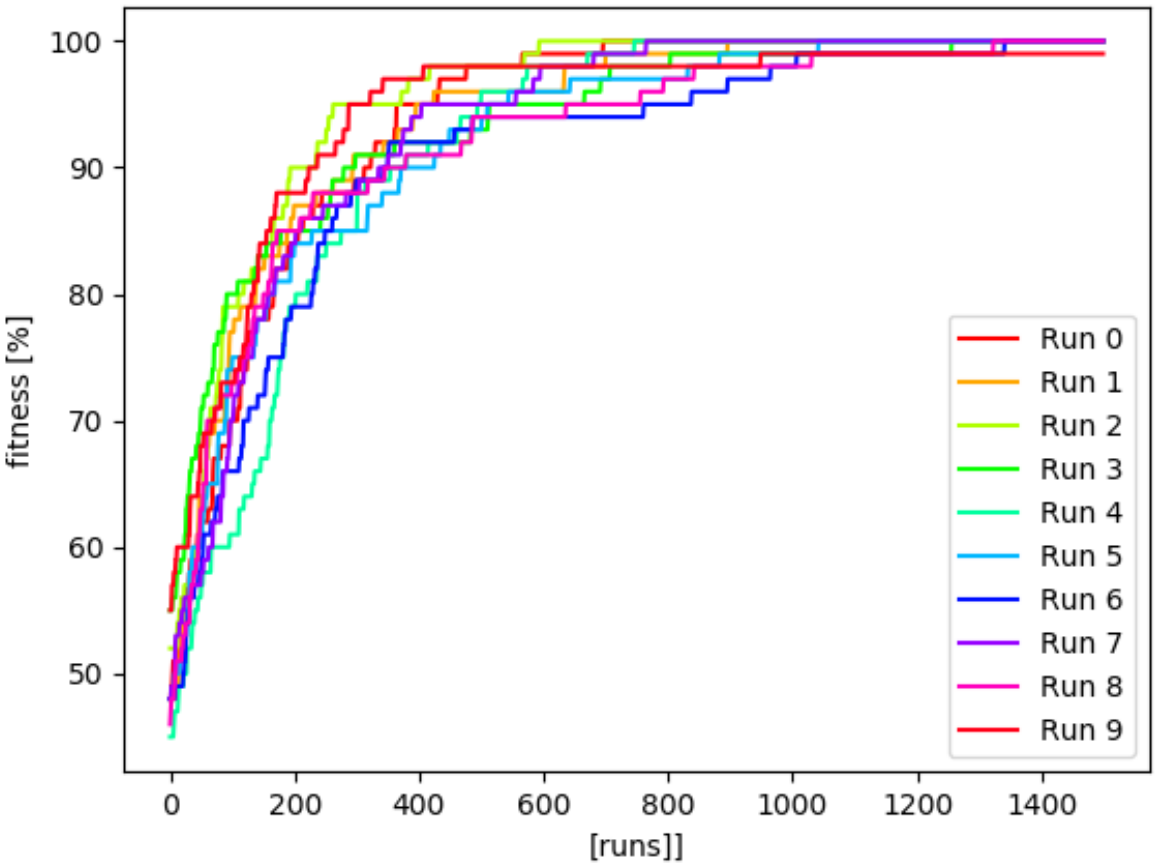

Figure 2: Performance of some genetic algorithm

A Appendix __main__.py

```
from .Main import Main

print(f'Hi, I\'ll be running the main code, and I\'ll let you know when I\'m done.')
project_nr = 1
main = Main()

# run a genetic algorithm to create some data for a plot.
print("now running a")
res = main.do_run_a()

# plot some graph with a single line, general form is:
# plt_tex.plotSingleLines(plt_tex,x,y,"x-axis label","y-axis label", lineLabels,"filename",legend_position,project_nr)
# main.plt_tex.plotSingleLine(plt_tex,range(0, len(res)),res,"[runs ]]","fitness [%]","run 1","4a",4,project_nr)

# run a genetic algorithm to create some data for another plot.
print("now running b")
main.do4b(project_nr)

# run a genetic algorithm to create some data for another plot.
print("now running 4c")
main.do4c(project_nr)

print(f'Done.')
```

B Appendix Main.py

```
# Example code that creates plots directly in report
# Code is an implementation of a genetic algorithm
import random
from matplotlib import pyplot as plt
from matplotlib import lines
import matplotlib.pyplot as plt
from .Plot_to_tex import Plot_to_tex as plt_tex

import numpy as np
string_length = 100
mutation_chance= 1.0/string_length
max_iterations = 1500
class Main:

    def __init__(self):
        pass

    def count(self,bits):
        count = 0
        for bit in bits:
            if bit:
                count = count + 1
        return count

    def gen_bit_sequence(self):
        bits = []
        for _ in range(string_length):
            bits.append(True if random.randint(0, 1) == 1 else False)
        return bits

    def mutate_bit_sequence(self,sequence):
        retval = []
        for bit in sequence :
            do_mutation = random.random() <= mutation_chance
            if(do_mutation):
                retval.append(not bit)
            else:
                retval.append(bit)
        return retval

    #execute a run a
    def do_run_a(self):

        seq = self.gen_bit_sequence()
        fitness = self.count(seq)
        results = [fitness]
        for run in range(max_iterations-1):
            new_seq = self.mutate_bit_sequence(seq)
            new_fitness = self.count(new_seq)
            if new_fitness > fitness:
                seq = new_seq
                fitness = new_fitness
            results.append(max(results[-1],fitness))
        return results


    #execute a run c
    def do_run_c(self):
        seq = self.gen_bit_sequence()
        fitness = self.count(seq)
```

```
61          results = [fitness]
62          for run in range(max_iterations):
63              new_seq = self.mutate_bit_sequence(seq)
64              new_fitness = self.count(new_seq)
65              seq = new_seq
66              fitness = new_fitness
67              results.append(max(results[-1], fitness))
68          return results
69
70      def do4b(self,project_nr):
71          optimum_found = 0
72
73          # generate plot data
74          plotResult = np.zeros((10,max_iterations), dtype=int);
75          lineLabels = []
76
77          # perform computation
78          for run in range(10):
79              res = self.do_run_a()
80              if res[-1] == string_length:
81                  optimum_found +=1
82
83              # store computation data for plotting
84              lineLabels.append(f'Run {run}')
85              plotResult[run,:]=res;
86
87          # plot multiple lines into report (res is an array of
                ↪ dataseries (representing the lines))
88          # plt_tex.plotMultipleLines(plt_tex,x,y,"x-axis label","y-
                ↪ axis label",lineLabels,"filename",legend_position,
                ↪ project_nr)
89          plt_tex.plotMultipleLines(plt_tex,range(0, len(res)),
                ↪ plotResult,"[runs]]","fitness [%]",lineLabels,"4b",4,
                ↪ project_nr)
90          print("total optimum found: {} out of {} runs".format(
                ↪ optimum_found,10))
91
92      def do4c(self,project_nr):
93          optimum_found = 0
94
95          # generate plot data
96          plotResult = np.zeros((10,max_iterations+1), dtype=int);
97          lineLabels = []
98
99          # perform computation
100         for run in range(10):
101             res = self.do_run_c()
102             if res[-1] == string_length:
103                 optimum_found +=1
104
105             # Store computation results for plot
106             lineLabels.append(f'Run {run}')
107             plotResult[run,:]=res;
108
109         # plot multiple lines into report (res is an array of
                ↪ dataseries (representing the lines))
110         # plt_tex.plotMultipleLines(plt_tex,x,y,"x-axis label","y-
                ↪ axis label",lineLabels,"filename",legend_position,
                ↪ project_nr)
111         plt_tex.plotMultipleLines(plt_tex,range(0, len(res)),
                ↪ plotResult,"[runs]]","fitness [%]",lineLabels,"4c",4,
                ↪ project_nr)
```

```
112
113         print("total optimum found: {} out of {} runs".format(
                ↪ optimum_found, 10))
114
115     def addTwo(self,x):
116         ''' adds two to the incoming integer and returns the result
                ↪ of the computation.'''
117         return x+2
118
119 if __name__ == '__main__':
120     # initialize main class
121     main = Main()
122
123     # set the project number folder since this is python code, the
            ↪ results are exported to the project2 report
124     project_nr = 1
125
126     # run a genetic algorithm to create some data for a plot.
127     print("now running a")
128     res = main.do_run_a()
129
130     # plot some graph with a single line, general form is:
131     # plt_tex.plotSingleLines(plt_tex,x,y,"x-axis label","y-axis
            ↪ label",lineLabels,"filename",legend_position,project_nr)
132     plt_tex.plotSingleLine(plt_tex,range(0, len(res)),res,"[runs]","
            ↪ fitness [%]","run 1","4a",4,project_nr)
133
134     # run a genetic algorithm to create some data for another plot.
135     print("now running b")
136     main.do4b(project_nr)
137
138     # run a genetic algorithm to create some data for another plot.
139     print("now running 4c")
140     main.do4c(project_nr)
```

Appendix python code that exports figures to latex

```
### Call this from another file, for project 11, question 3b:
### from Plot_to_tex import Plot_to_tex as plt_tex
### multiple_y_series = np.zeros((nrOfDataSeries,nrOfDataPoints), dtype=int); # actually fill with data
### lineLabels = [] # add a label for each dataseries
### plt_tex.plotMultipleLines(plt_tex,single_x_series, multiple_y_series,"x-axis label [units]","y-axis label [units]",lineLabels,"3b",4,11)
### 4b=filename
### 4 = position of legend, e.g. top right.
###
### For a single line, use:
### plt_tex.plotSingleLine(plt_tex,range(0, len(dataseries)), dataseries,"x-axis label [units]","y-axis label [units]", lineLabel,"3b",4,11)

### You can also plot a table directly into latex, see example_create_a_table(..)
###
### Then put it in latex with for example:
###\begin{table}[H]
###    \centering
###    \caption{Results some computation.}\label{tab:some_computation}
###    \begin{tabular}{|c|c|} % remember to update this to show all columns of table
###        \hline
###        \input{latex/project3/tables/q2.txt}
###    \end{tabular}
###\end{table}
import random
from matplotlib import lines
import matplotlib.pyplot as plt
import numpy as np
import os
class Plot_to_tex:

    def __init__(self):
        self.script_dir = self.get_script_dir()
        print("Created main")

    # plot graph (legendPosition = integer 1 to 4)
    def plotSingleLine(self,x_path,y_series,x_axis_label,y_axis_label,label,filename,legendPosition,project_nr):
        fig=plt.figure();
        ax=fig.add_subplot(111);
        ax.plot(x_path,y_series,c='b',ls='-',label=label,fillstyle='none');
        plt.legend(loc=legendPosition);
        plt.xlabel(x_axis_label);
        plt.ylabel(y_axis_label);
        plt.savefig(os.path.dirname(__file__)+'/../../../latex/project'+str(project_nr)+'/Images/'+filename+'.png');
#         plt.show();

    # plot graphs
    def plotMultipleLines(self,x,y_series,x_label,y_label,label,filename,legendPosition,project_nr):
        fig=plt.figure();
        ax=fig.add_subplot(111);
```

```
49
50          # generate colours
51          cmap = self.get_cmap(len(y_series[:,0]))
52
53          # generate line types
54          lineTypes = self.generateLineTypes(y_series)
55
56          for i in range(0,len(y_series)):
57              # overwrite linetypes to single type
58              lineTypes[i] = "-"
59              ax.plot(x,y_series[i,:],ls=lineTypes[i],label=label[i],
                    ↪ fillstyle='none',c=cmap(i)); # color
60
61          # configure plot layout
62          plt.legend(loc=legendPosition);
63          plt.xlabel(x_label);
64          plt.ylabel(y_label);
65          plt.savefig(os.path.dirname(__file__)+'/../../../latex/
                ↪ project'+str(project_nr)+'/Images/'+filename+'.png');
66
67          print(f'plotted lines')
68
69      # Generate random line colours
70      # Source: https://stackoverflow.com/questions/14720331/how-to-
            ↪ generate-random-colors-in-matplotlib
71      def get_cmap(n, name='hsv'):
72          '''Returns a function that maps each index in 0, 1, ..., n-1
                ↪ to a distinct
73          RGB color; the keyword argument name must be a standard mpl
                ↪ colormap name.'''
74          return plt.cm.get_cmap(name, n)
75
76      def generateLineTypes(y_series):
77          # generate varying linetypes
78          typeOfLines = list(lines.lineStyles.keys())
79
80          while(len(y_series)>len(typeOfLines)):
81              typeOfLines.append("-.");
82
83          # remove void lines
84          for i in range(0, len(y_series)):
85              if (typeOfLines[i]=='None'):
86                  typeOfLines[i]='-'
87              if (typeOfLines[i]==''):
88                  typeOfLines[i]=':'
89              if (typeOfLines[i]==' '):
90                  typeOfLines[i]='--'
91          return typeOfLines
92
93      # Create a table with: table_matrix = np.zeros((4,4),dtype=object
            ↪ ) and pass it to this object
94      def put_table_in_tex(self, table_matrix,filename,project_nr):
95          cols = np.shape(table_matrix)[1]
96          format = "%s"
97          for col in range(1,cols):
98              format = format+" & %s"
99          format = format+""
100         plt.savetxt(os.path.dirname(__file__)+"/../../../latex/
                ↪ project"+str(project_nr)+"/tables/"+filename+".txt",
                ↪ table_matrix, delimiter=' & ', fmt=format, newline='
                ↪ \\\\ \hline \n')
101
```

```
102     # replace this with your own table creation and then pass it to
          ↪ put_table_in_tex(..)
103     def example_create_a_table(self):
104         project_nr = "1"
105         table_name = "example_table_name"
106         rows = 2;
107         columns = 4;
108         table_matrix = np.zeros((rows,columns),dtype=object)
109         table_matrix[:,:]="" # replace the standard zeros with emtpy
              ↪ cell
110         print(table_matrix)
111         for column in range(0,columns):
112             for row in range(0,rows):
113                 table_matrix[row,column]=row+column
114         table_matrix[1,0]="example"
115         table_matrix[0,1]="grid sizes"
116 
117         self.put_table_in_tex(table_matrix,table_name,project_nr)
118 
119 
120     def get_script_dir(self):
121         ''' returns the directory of this script regardles of from
              ↪ which level the code is executed '''
122         return os.path.dirname(__file__)
123 
124 if __name__ == '__main__':
125     main = Plot_to_tex()
126     main.example_create_a_table()
```