Exam Solution

Course: SOW-MKI96 Neuromorphic Computing
Assignment Source:brightspace
Assignment Due Date: 20-09-14 10-30-00

Author:
collaborative effort

06-09-2019

0 Introduction

This contains an exam solution. If you wish to contribute to this exam solution:

1. Create a github account, (you can create an "anonymous" one).
2. git clone ...
3. edit your changes in the document.
4. open cmd, and browse to inside the folder you downloaded and edited
5. git pull (updates your local repository=copy of folder, to the latest version in github cloud)
6. git status shows which files you changed.
7. git add "/some folder with a space/someFileYouChanged.tex"
8. git commit -m "Included solution to question 1c."
9. git push

It can be a bit initimidating at first, so feel free to click on "issue" in the github browser of this repository and ask :) (You can also use that to say "Hi, I'm having a bit of help with this particular equation, can someone help me out?")

If you don't know how to edit a latex file on your own pc iso on overleaf, look at the "How to use" section of `https://github.com/a-t-0/AE4872-Satellite-Orbit-Determination`.

0.1 Consistency

To make everything nice and structured, please use very clear citations:

1. If you copy/use an equation of some slide or document, please add the following data:
   (a) Url (e.g. if simple wiki or some site)
   (b) Name of document
   (c) (Author)
   (d) PAGE/SLIDE number so people can easily find it again
   (e) equation number (so people can easily find it again)
2. If you use an equation from the slides/a book that already has an equation number, then hardcode that equation number in this solution manual so people directly see which equation in the lecture material it is, this facilitates remembering the equations.
3. Here is an example is given in **??** (See file references.bib [**?**]).

$$R_n^E = \sum_{t=1}^{n} l_m(p_t, z_t) - \min_i \sum_{t=1}^{n} {z_t}^i \tag{10.32[?]}$$

1 Turing Tumble

Conclusion

A 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)
```

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

    def do4b(self,hw_nr):
        optimum_found = 0

        # generate plot data
        plotResult = np.zeros((10,max_iterations), dtype=int);
        lineLabels = []

        # perform computation
        for run in range(10):
            res = self.do_run_a()
            if res[-1] == string_length:
                optimum_found +=1

            # store computation data for plotting
            lineLabels.append(f'Run {run}')
            plotResult[run,:]=res;

        # plot multiple lines into report (res is an array of
            ↪ dataseries (representing the lines))
        # plt_tex.plotMultipleLines(plt_tex,x,y,"x-axis label","y-
            ↪ axis label",lineLabels,"filename",legend_position,hw_nr
            ↪ )
        plt_tex.plotMultipleLines(plt_tex,range(0, len(res)),
            ↪ plotResult,"[runs]]","fitness [%]",lineLabels,"4b",4,
            ↪ hw_nr)
        print("total optimum found: {} out of {} runs".format(
            ↪ optimum_found,10))

    def do4c(self,hw_nr):
        optimum_found = 0

        # generate plot data
        plotResult = np.zeros((10,max_iterations+1), dtype=int);
        lineLabels = []

        # perform computation
        for run in range(10):
            res = self.do_run_c()
            if res[-1] == string_length:
                optimum_found +=1

            # Store computation results for plot
            lineLabels.append(f'Run {run}')
            plotResult[run,:]=res;

        # plot multiple lines into report (res is an array of
            ↪ dataseries (representing the lines))
        # plt_tex.plotMultipleLines(plt_tex,x,y,"x-axis label","y-
            ↪ axis label",lineLabels,"filename",legend_position,hw_nr
            ↪ )
        plt_tex.plotMultipleLines(plt_tex,range(0, len(res)),
            ↪ plotResult,"[runs]]","fitness [%]",lineLabels,"4c",4,
            ↪ hw_nr)
```

```
        print("total optimum found: {} out of {} runs".format(
            ↪ optimum_found, 10))

if __name__ == '__main__':
    # initialize main class
    main = Main()

    # set the hw number folder since this is python code, the results
        ↪ are exported to the hw2 report
    hw_nr = 2

    # 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,hw_nr)
    plt_tex.plotSingleLine(plt_tex,range(0, len(res)),res,"[runs]]","
        ↪ fitness [%]","run 1","4a",4,hw_nr)

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

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