UNIVERSITY OF WATERLOO

Department of Electrical and Computer Engineering

ECE 457A Adaptive and Cooperative Algorithms

Instructor: Dr. Otman A. Basir

Assignment 3
Due Date: November 10, 2024 11:59PM

Instructions

- This assignment can be approached in groups. Each group consists of upto three students.
- You are expected to do all problems of the assignment even if you are not doing it in a group.
- You should upload your answers as a PDF file on learn before 11:59pm of the deadline date
- Attach any codes used for the assignment separately as a compressed file to the same dropbox
- You can use any programming language (Matlab and Python are preferred)
- Submitted work with more than 30% similarity will not be marked
- Communicate any issues or concerns with the TAs.
- Deliverables: provided for each question
- The three problems are equally weighted

Problem 1

GA can be used to design a PID (Proportional-Integral-Differential) controller for closed loop plant control systems. A common configuration is shown in the following Figure 1. Notice that the output of this system has an explicit impact on the input of the controller, which explains the "closed-loop" term. In this loop, the output of the system is what we would like to control to the set-point. So, the controller is being fed by the error.
The PID controller is specified by a transfer function of the form:

$$G_C(s) = K_p \left(1 + \frac{1}{T_I s} + T_D s\right)$$

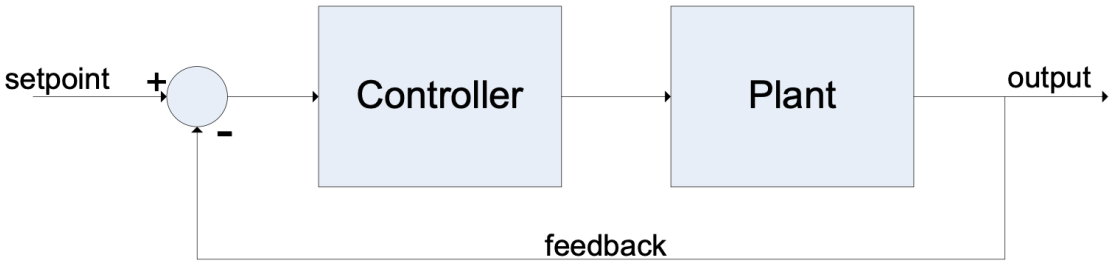

Figure 1. Closed-loop control system

Figure 1:

Where K_p, T_I, and T_D are the controller parameters. The objective of the design is to obtain the values of these parameters that optimize the performance of the system. For a given plant transfer function, the performance of the system is typically evaluated by the step response of the system which is of the form shown in figure 2 .

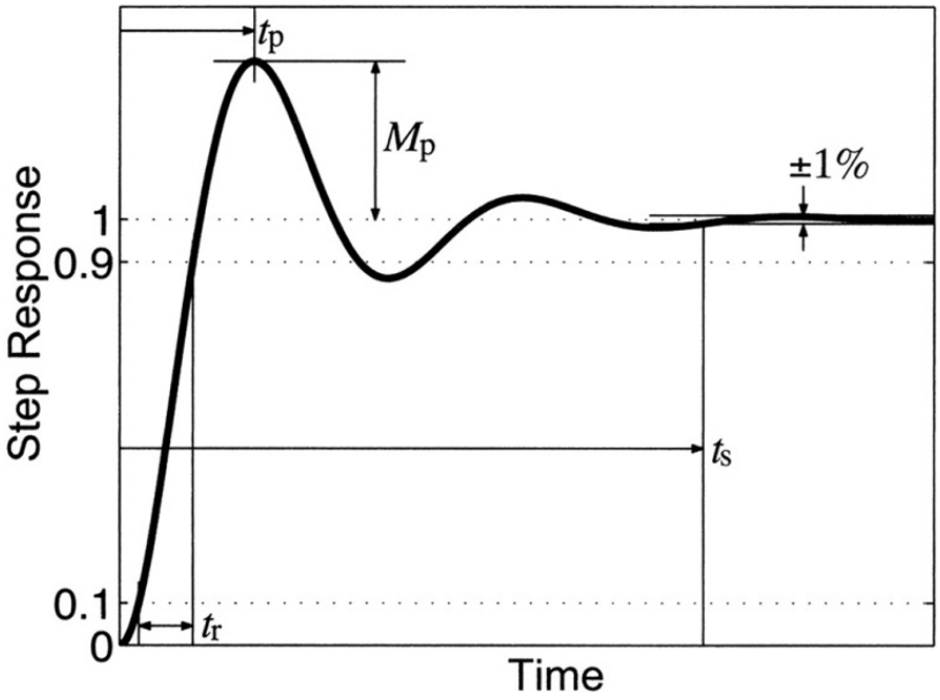

Figure 2. Step response parameters

The performance is measured in terms of integral squared error (ISE),

$$ISE = \int_0^T (e(t))^2 dt \tag{1}$$

which is the integral of the square of the difference between the output and the steady state value (1 in this case), along with the values of the step response parameters: t_r, the rise-time; t_s, the settling time, and M_p, the maximum overshoot magnitude, which can be addressed as the percentage of the steady-state output. t_r and t_s are important, as they demonstrate how fast a system can work. M_p is of importance as well, as we may abide by the plant restrictions. So it is important to minimize all these performance measures.

In this problem you are provided a function that gives you the performance measures if you provide it with values for the parameters K_p, T_I, and T_D. The function is functions of Matlab control toolbox.
Assume that the values of the controller parameters are in the ranges: $K_p \in (2, 18), T_I \in (1.05, 9.42), T_D \in (0.26, 2.37)$

Deliverables

1. Develop a suitable representation for the solutions with precision of 2 decimal points.
2. Formulate a fitness function that you can use to evaluate a solution.
3. Implement GA algorithm to solve this problem, use a population of 50 individuals, number of generations of 150 , crossover probability of 0.6 and mutation probability of 0.25. Use FPS parent selection strategy and an elitism survival selection strategy keeping the best two individuals across generations. Select a crossover and mutation operators and solve the problem.
4. Plot the fitness of best solution in each generation across the generations. In this part we would like to study the effect of the choice of the GA parameters:
5. Experiment with 3 different values for the number of generations and report the progression of the solutions.
6. Repeat 5 for 3 different population sizes.
7. Experiment with different crossover and permutation probabilities (within the guidelines given in class) and report the results.

Problem 2

The below pseudo-code outlines the adaptive (1+1) evolution strategy where n is the problem dimension. x_1 is equal to x_0 but with each feature mutated, ϕ is the proportion of mutations during the past G generations that result in x_1 being better than x_o. The mutation variance is automatically adjusted to increase the rate of convergence. The nominal value of c is 0.817.

Initialize the non-negative mutation variance σ^2
$x_0 \leftarrow$ randomly generated individual
While not(termination criterion)
    Generate a random vector r with $r_i \sim N(0, \sigma^2)$ for $i \in [1, n]$
    $x_1 \leftarrow x_0 + r$
    If x_1 is better than x_0 then
        $x_0 \leftarrow x_1$
    End if
    $\phi \leftarrow$ proportion of successful mutations during the past G generations
    If $\phi < 1/5$
        $\sigma \leftarrow c^2\sigma$
    else if $\phi > 1/5$
        $\sigma \leftarrow \sigma/c^2$
    End if
Next generation

The sphere function is given as

$$f(x) = \sum_{i=1}^{n} x_i^2$$
$$x^* = 0$$
$$f(x^*) = 0$$

where $x_i \in [-5.12, +5.12]$. This is called function 1 in Ken De Jong's thesis [De Jong, 1975], and it is Problem 1.1 and Problem 2.17 in [Schwefel, 1995], see below. Figure C.1 below shows a plot of $f(x)$ in two dimensions. This is a very simple optimization problem, and almost any reasonable algorithm should be able to find its minimum accurately, but it provides a good preliminary test for optimization algorithms. It also provides a good benchmark for comparison between algorithms, because many optimization problems are approximately quadratic near their minimum.

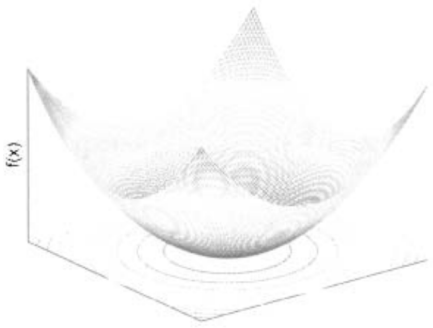

Figure C.1 The two-dimensional sphere function.

1. De Jong, K. (1975). An Analysis of the Behaviour of a Class of Genetic Adaptive Systems. PhD thesis, University of Michigan.
2. Schwefel, H.-P. (1995). Evolution and Optimum Seeking. John Wiley & Sons. Expanded version of [Schwefel, 1981].

Deliverables

1. Use the adaptive (1 + 1)-ES shown Figure 3 to minimize the 10-dimensional sphere function (given below) on a domain of [—5.12,+5.12].
2. Initialize the standard deviation of the mutation of each dimension to $0.1/(2\sqrt{3})$.
3. Simulate for 500 generations, and record the cost at each generation.
4. Run 50 simulations like this, and average the 50 cost values at each generation.
5. Plot the average cost values as a function of generation number.
6. Do this for $c = 0.6, 0.8, \&1.0$. Which value of c gives the best performance?
7. Now that you have implemented and tested your ES code, use the DEAP platform to do the same and compare results. (`https://deap.readthedocs.io/en/master/index.html`). Examples of using DEAP to implement ES solutions can be found at: `https://deap.readthedocs.io/en/master/examples/index.html`. We will use this platform quite a bit moving forward.

Problem 3

Ant Foraging Behavior and TSP:

1. NetLogo is a high-level multi-agent modelling environment, very suitable for fast creation of agent-based models. NetLogo has a large library of sample models. The model "ANTS" demonstrates a colony of ants forages for food. Though each ant follows a set of simple rules, the colony as a whole acts in a sophisticated way.

   The first part of this question is to experiments on the NetLogo's ANTS model. The model provides control sliders to change the model parameters.

   Run experiments with population (30, 50, 100), diffusion rate (40, 80), evaporation rate (10, 20) and different placements of the food sources. Examining the ant colony's food foraging and transporting behavior (finish time), report your observations.

2. The target problem in this question is the 29-city TSP (known as Bays29 in literature). The distance between cities is symmetrical, and Euclidean distance measure is used. Each city must be visited once and only once. The objective is to minimize the sum of the Euclidean distances of a complete TSP tour. Ignore curvature of the earth. The 29 city coordinate matrix is listed below.

City no	X-coord.	Y-coord.	City no	X-coord.	Y-coord.	City no	X-coord.	Y-coord.
1	1150.0	1760.0	11	840.0	550.0	21	830.0	1770.0
2	630.0	1660.0	12	1170.0	2300.0	22	490.0	500.0
3	40.0	2090.0	13	970.0	1340.0	23	1840.0	1240.0
4	750.0	1100.0	14	510.0	700.0	24	1260.0	1500.0
5	750.0	2030.0	15	750.0	900.0	25	1280.0	790.0
6	1030.0	2070.0	16	1280.0	1200.0	26	490.0	2130.0
7	1650.0	650.0	17	230.0	590.0	27	1460.0	1420.0
8	1490.0	1630.0	18	460.0	860.0	28	1260.0	1910.0
9	790.0	2260.0	19	1040.0	950.0	29	360.0	1980.0
10	710.0	1310.0	20	590.0	1390.0			

Deliverables

For this problem do the following:

1. Code a simple ACO algorithm to solve the problem. To do this you need to select a transition rule, select online and offline pheromone update rules, set a population size and select a stopping criterion.
2. Run your ACO algorithm.
3. Perform the following changes on your ACO code (one by one) and compare the results.

   a. Change the values of pheromone persistence constant 3 times: record the performance of your ACO

   b. Change the values of state transition control parameter 3 times: record the performance of your ACO

   c. Change the population size twice: record the performance of your ACO

   d. Turn off online pheromone update: record the performance of your ACO
4. Compare the results in a,b,c, and d and summarize your observations/comments.

NetLogo ver-4.1.3, Wilensky, U. (1999). NetLogo. http://ccl.northwestern.edu/netlogo/. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL.