EIE557 Computational Intelligence and Its Applications

Laboratory 1: Optimization Using Genetic Algorithms

Objectives:
1. To gain experience in solving optimization problems using GA's
2. To be familiar with a GA optimization tool implemented in MATLAB

1. Introduction to the Genetic Algorithm Optimization Toolbox (GAOT)

GAOT implements GA's in the Matlab environment using both binary and real-value representations. The implementation is very flexible in the genetic operators, selection functions, as well as the evaluation functions that can be used.

The pseudo code of the simple genetic algorithm (SGA) implemented is given below:

(1) Supply (randomly generate) a population P_0 of N individuals and respective function values.
(2) $i \leftarrow 1$
(3) $P_i^{'} \leftarrow slection\_function\left(P_{i-1}\right)$
(4) $P_i \leftarrow reproduction\_function\left(P_i^{'}\right)$
(5) $evaluate\left(P_i\right)$
(6) $i \leftarrow i+1$
(7) Repeat Steps 3 to 6 until termination
(8) Print out best solution found

1.1 Solution Representation

Each individual or chromosome is made up of a sequence of genes from a certain Alphabet. An alphabet could consist of binary digits (0 and 1), floating point numbers, integers, symbols (i.e. A, B, C, D), matrices, etc.

1.2 Selection Function

The selection of individuals to produce successive generations plays an extremely important role in a genetic algorithm. A probabilistic selection is performed based upon the individual's fitness such that the better individuals have an increased chance of being selected. An individual in the population can be selected more than once with all individuals in the population having a chance of being selected to reproduce into the next generation. The three selections functions implemented are summarized in Table 1-1.

Table 1-1: Implemented selection functions

Name	The keyword	Parameter(s)
Roulette wheel	'roulette'	None
Normalized geometric selection	'normGeomSelect'	Probability of selecting the best
Tournament	'tournSelect'	No. of individuals in each tournament

1.3 Genetic Operators

Genetic operators provide the basic search mechanism of the GA. The operators are used to create new solutions based on existing solutions in the population. There are two basic types of operators: crossover and mutation. Crossover takes two individuals and produces two individuals while mutation alters one individual to produce a single new solution. Table 1-2 summarizes the different crossover and mutation operators available for binary and real-valued chromosome representation.

Table 1-2: Implemented crossover and mutation operators

Genetic operators	Binary chromosomes	Real-valued chromosomes
Crossover	• Simple (one-point) crossover 'simpleXover'	• Simple (one-point) crossover 'simpleXover' • Arithmetic crossover 'arithXover' • Heuristic crossover 'heuristicXover'
Mutation	• Binary mutation 'binaryMutation'	• Uniform mutation 'unifMutation' • Non-uniform mutation 'nonUnifMutation' • Multi-non-uniform mutation 'multiNonUnifMutation' • Boundary mutation 'boundaryMutation'

1.4 Initialization, Termination, and Evaluation Functions

The initialization, termination, and evaluation functions implemented are summarized in Table 1-3.

Table 1-3: Initialization, termination and evaluation functions

	Setting
Initialization	Random generation
Termination	Maximum number of generations
Evaluation	Problem dependent, defined for each problem

2. Exercises

2.1 Evolving Investment Strategies Using the GA

2.1.1 Introduction

A best rule will be found by using the simple genetic algorithm to make monthly decision on whether to make investment to the Hangseng Index (HSI) stocks. The decision rules will make use of the HK macroeconomic data and data from other markets in the previous month to decide the investment for the current month. Two strategies are implemented:

Investment Strategy 1:

A decision is made on whether to be in and out of the HSI stocks. We assume that the return for a month when we do not invest is zero by ignoring the bank interest. The fitness is calculated by accumulating the return of the 'investment' months.

Investment Strategy 2:

A decision is made on whether to long and short the HSI stocks. The fitness is calculated by a summation of the return of those 'long' months plus the summation of the return of those 'short' months. Note that an HSI gain/lose in a month will be a positive/negative return to a 'long" month and a negative/positive return for a 'short' month.

Table 2-1: Data sets to be used in constructing decision rules for investment strategies

Variable	Description	Data File	Remarks
1	Dow Jones Index	dj.mat	Monthly average of daily close prices
2	S&P 500 Index	s&p.mat	Monthly average of daily close prices
3	Nasdaq Index	nasdq.mat	Monthly average of daily close prices
4	Spot Gold Price	gold.mat	Monthly average of daily close prices
5	Japanese Yen/US dollar	jpy.mat	Monthly average of daily close prices
6	10-Year Treasury Bond Yield	10y-yeild.mat	Monthly average of daily close prices
7	HK Property Trading Volume	property-trade-volume.mat	Monthly data
8	HK Consumer Price Index	cpi.mat	Monthly data
9	HK M3 (money	money.mat	Monthly data

	supply)		
10	HK Saving	saving.mat	Monthly data
11	HK Number of Visitors	visitors.mat	Monthly data
12	HK Retail Index	retail-index.mat	Monthly data
13	HK Trading (Export)	trade-export.mat	Monthly data
14	HK Unemployment Rate	unemployment-rate.mat	Monthly data
15	Monthly Change of Variable 1	dj_c.mat	
16	Monthly Change of Variable 2	s&p_c.mat	
17	Monthly Change of Variable 3	nasdq_c.mat	
18	Monthly Change of Variable 4	gold_c.mat	
19	Monthly Change of Variable 5	jpy_c.mat	
20	Monthly Change of Variable 6	10y-yeild_c.mat	

Table 2-1 shows Hong Kong macroeconomic data sets and the data from the other markets that may be used for building up decision rules. A combination of three variables will be used to construct the decision rules using the GA. Example decision rules look like:

Investment Strategy 1:

IF Variable *i* is LESS THAN *X*, AND Variable *j* is GREATER THAN OR EQUAL TO *Y*, AND Variable *k* is LESS THAN *Z*, THEN invest in common stocks ELSE keep the cash.

Investment Strategy 2:

IF Variable *i* is LESS THAN *X*, AND Variable *j* is GREATER THAN OR EQUAL TO *Y*, AND Variable *k* is LESS THAN *Z*, THEN long common stocks ELSE short the common stocks.

In the implementation, 32 cutoff values are used for each variable. Eight logical groupings and eight combinations of “LESS THAN” and “GREATER THAN OR EQUAL TO” for three variables are used. Therefore, a solution chromosome will be a 21-bit binary string.

2.1.2 Problems

(1) Use a combination of three variables to run the GA to find the best rule. Some possible combinations are (a) Variables 1, 4, 9; (b) Variables 2, 4, and 9; (c) Variables 3, 4, and 9; (d) Variables 7, 8, and 9, and (e) Variables 9, 15, 18. A suggested parameter set for the GA is given in Table 2-2. Record the best rule that produces the best return for each of two investment strategies. Show the return vs. generation graph.

Table 2-2: Suggested parameters for running the GA

Population Size	No. of Generations	Selection Function	Probability of the simple crossover	Probability of binary mutation
10	15	Roulette wheel select	0.6	0.05

(2) Use the same three-variable combination as in (1) and change the GA parameters to run the GA. Use Table 2-3 to report the simulation results. Explain your findings. Use graphs and other data if necessary.

(3) Use different combinations of three variables to go through the GA simulation in order to find the best combination of three variables. You may fix the GA parameter set. Report and discuss your findings. Use Table 2-4 to report the experimental results.

(4) Explain the simulation results in the context of investment performance and discuss whether the strategies could work in real situations.

Note: Students who find the top performed rule(s) of using different combinations of three variables will get bonus marks for the lab exercise.

Procedure 1:

Run "**gainvest.m**" to conduct simulation to find the rule for a combination of three variables you choose.

In "gaininvest.m', you may make the following changes:

1) **Select combinations**

Line 5-8:

```
% specify the condition data set
var1='retail-index.mat';
var2='dj.mat';
var3='jpy.mat';
```

2) **Select investment strategy**

Line 10-11:
% select strategy
strategy=1;%% '1' for strategy 1; '2' for strategy 2

3) **GA parameters setting**

Line 15-16
% initialization
initPop=initializega(10,[0 2^21],'gainvesteval',[],[1 0]);

- 'initializega.m' is a function to generate the initial population.
- 10: the population size
- [0 2^21]: the bounds for the variable (here the variable is a 21-bit binary string.)
- 'gainvesteval': the fitness (the return or lose) of each initial individual is computed in gainvesteval.m.
- []: no parameter setting is required for the fitness calculation.
- [1 0]:
  The first parameter: the smallest difference between two chromosomes (1 for this problem).
  The second parameter: 0 for binary-coded chromosomes (use 0 for this problem) and 1 for real-valued coded chromosomes.

Line 18-20:
% Now let's run the ga
[x,endPop,bPop,traceInfo] = ga([0 2^21],'gainvesteval',[],initPop,[1 0 0],'maxGenTerm',15,'normGeomSelect',[0.08],['simpleXover'],[0.6],'binaryMutation',[0.05]);

- 'ga.m' is a function to run the GA
- [0 2^21]: the lower and upper bounds of the variable
- 'gainvesteval': the fitness (the return or lose) is computed in gainvesteval.m.
- initPop: use the individuals in initPop for the initial population.
- [1 0 0]:
  The first parameter: the smallest difference between two chromosomes (1 for this problem)
  The second parameter: **0 if you want to apply genetic operators probabilistically (use 0 for binary-coded chromosomes) and 1 if you supply a deterministic number of operator applications (use 1 for real-valued chromosomes).**
  The third parameter: 1 to output progress and 0 for the 'quiet' mode.
- 'maxGenTerm': Use the maximum number of generations to terminate the training
- 15: the maximum number of generations. You may change this parameter.

- 'normGeomSelect': Select Normalized geometric Selection scheme. You may select ‘roulette’ (with option []) and ‘tournSelect’ (with option [‘the number of tournament size’].
- [0.08]: the probability for selecting the best chromosome in 'normGeomSelect', you may change this parameter
- ‘simpleXover’: only simple crossover is available for binary-coded chromosomes.
- [0.6]: crossover probability, you may change this parameter
- ‘binaryMutation’: only binary (single-point) mutation is available for binary-coded chromosomes
- [0.05]: mutation probability, you may change this parameter.

Note: You may type “help function name” in the command window to show the parameters for that function. For example, ‘help ga’.

4) **Output variables** of gainvest.m :

best_rule -- the best rule obtained;
final_gain_train – gain in the training period;
correct_rate_train – correct rate of predictions in the training period;
final_gain_test -- gain in the test period;
correct_rate_test – correct rate of predictions in the test period.

5) **The fitness function** of this problem is defined in ‘gainvesteval.m’.

6) **Data sets** are in the folder “data”, including “condition_data” and “gain_data”.

Table 2-3: Experimental results for the combination of three variables you have chosen **(the gain of the buy-and-hold strategy is 26% for the training period of 01/2000 – 12/2003 and -6% for the test period of 01/2004 – 06/2005).**

Simulation run	GA parameters used including no. of generation, population size, selection function, probabilities for crossover and mutation	The best rule obtained	Gain (lose) in the training period (01/2000 – 12/2003)		Gain (lose) in the test period (01/2004 – 06/2005)	
			Strategy 1	Strategy 2	Strategy 1	Strategy 2
1						
2						
3						
4						
5						

Table 2-4: Experimental results for different three-variable combinations

Simulation run and variables used	GA parameters used including no. of generation, population size, selection function, probabilities for crossover and mutation	The best rule obtained	Gain (lose) on the training period (01/2000 – 12/2003)		Gain (lose) on the test period (01/2004 – 06/2005)	
			Strategy 1	Strategy 2	Strategy 1	Strategy 2
1 Variables:						
2 Variables:						
3 Variables:						
4 Variables:						
5 Variables:						

Warning: This is a lab exercise only to accompany your subject study on computational intelligence and its applications. It does not suggest that the decision rules found always work to your favor. Historical data may not produce good strategies for investment in the future. Using the decision rules obtained in this lab exercise to guide your investment is at your own risk. The subject lecturer shall not bear any liability for any loss caused by using these rules in your investment.

2.2 Optimization of Single-Variable Functions

2.2.1 Introduction

The function to be maximized is $f(x) = x\sin(10\pi \cdot x) + 2.0$ ($x \in [-1,2]$).

Use a binary string to represent a solution. Assume that a precision of 10^{-5} should be achieved (19-bit strings should be used).

2.2.2 Problems

(1) Decide a fitness function to be used.

(2) Run the GA using different GA parameter setting to find the best solutions. Use Table 2-5 to report the simulation results.

(3) Is a real-value encoded chromosome suitable for this problem?

Table 2-5: Experimental results for the optimization of the single-variable function

Simulation run	GA parameters used including no. of generation, population size, selection function, probabilities for crossover and mutation	The best variable value	The function value
1			
2			
3			
4			
5			

Procedure 2:

Run "**gasingle_var.m**" to conduct the GA simulation.

initPop=initializega(10,[-1 2],'gasingle_vareval',[],[10^(-5) 0]);

- 10: the population size
- [-1 2]: the bounds for the variable
- 'gasingle_vareval': the fitness of each initial individual is computed in gasingle_vareval.m.
- []: no parameter setting is required for the fitness calculation.
- [10^5 0]:
  The first parameter: the precision
  The second parameter: 0 for binary-coded chromosomes and 1 for real-valued coded chromosomes.

```
[x,endPop,bPop,traceInfo] = ga([-1 2],'gasingle_vareval',[],initPop,[10^(-5) 0
0],'maxGenTerm',15,...
 'normGeomSelect',[0.08],['simpleXover'],[0.6],'binaryMutation',[.05]);
```

- initPop: use the solutions in initPop for the initial population.
- [10^(-5) 0 0]:
  The first parameter: the precision
  The second parameter: **0 if you want to apply genetic operators probabilistically (use 0 for binary-coded chromosomes) and 1 if you supply a deterministic number of operator applications (use 1 for real-valued chromosomes).**
  The third parameter: 1 to output progress and 0 for the 'quiet' mode.

1) **Output variables** of gasingle_var.m:

   x -- the best variable value
   f -- the maximum function value

2) The evaluate (fitness) function of this problem is defined in 'gasingle_vareval.m'.

2.3 Multi-Variable Functions

2.3.1 Introduction

The function to be maximized is: $f(x_1, x_2,..., x_{10}) = (x_1 x_2 x_3 x_4 x_5)/((x_6 x_7 x_8 x_9 x_{10}))$ where $x_i \in [1,10], i = 1,2,...,10$.

2.3.1 Problems

(1) Use 10-gene real-value coded chromosomes and parameters shown in Table 2-5 to conduct the simulation. Set the number of generation to 200 and the population size to 30. Run the GA with different selection, crossover and mutation functions

and different parameters. Use Table 2-6 to report the simulation results. Discuss your findings.

Table 2-5: Suggested parameters used for real-value coded multi-variable function optimization

Name	Parameters
Uniform mutation	5 (the number of mutations of the type to be carried out in each generation with randomly selected parents)
Boundary mutation	5 (as the above)
Simple crossover	5 (the number of crossover operations of the type to be carried out in each generation with randomly selected parents)
Arithmetic crossover	5 (as the above)
Normalized geometric selection	0.08 (the probability for the best individual)
Tournament selection	5 (the tournament size)

Table 2-6: Experimental results for the optimization of the multi-variable function

Simulation run	GA operators (selection, crossover, and mutation functions) and parameters used	Best variable values	The function value
1			
2			
3			
4			
5			
6			
7			

8			
9			
10			

Procedure 3:

Run "**gamultiple_var.m**" to conduct the GA simulation.

- initPop=initializega(30,[1 10;1 10;1 10;1 10;1 10;1 10;1 10;1 10;1 10;1 10],'gamultiple_vareval',[],[10^(-10) 1]);

- 30: the population size
- [1 10;1 10;1 10;1 10;1 10;1 10;1 10;1 10;1 10;1 10]: the bounds for 10 variables (here the variables are real-value coded)
- "gamultiple_vareval": the fitness of each initial individual is computed in gamultiple_vareval.m.
- []: no parameter setting is required for the fitness calculation.
- [10^(-10) 1]):
  The first parameter: the precision
  The second parameter: 0 for binary-coded chromosomes and 1 for real-valued coded chromosomes.

[x,endPop,bPop,traceInfo] = ga([1 10;1 10;1 10;1 10;1 10;1 10;1 10;1 10;1 10;1 10],'gamultiple_vareval',[],initPop,[10^(-10) 1 0],'maxGenTerm',100,...
'normGeomSelect',[0.08],['simpleXover arithXover'],[5 0; 5 0],'boundaryMutation unifMutation',[5 0; 5 0]);

- initPop: use the individuals in initPop for the initial population.
- [10^(-10) 1 0]:
  The first parameter: the precision
  The second parameter: **0 if you want to apply genetic operators probabilistically (use 0 for binary-coded chromosomes) and 1 if you supply a deterministic number of operator applications (use 1 for real-valued chromosomes).**
  The third parameter: 1 to output progress and 0 for the 'quiet' mode.
- 'normGeomSelect': Select Normalized geometric Selection scheme. You may select 'roulette' (with option []) and 'tournSelect' (with option ['the number of tournament size'].

- [0.08]: the probability for the best chromosome for ‘normGeomSelect’, you may change this parameter. For ‘tournSelect’, you need to specify the tournament size.
- ['simpleXover arithXover'],[5 0; 5 0]:
  Use ‘simpleXover’ and ‘arithXover’ 5 times each in each generation. You may use one crossover operator only.
- 'boundaryMutation unifMutation',[5 0; 5 0]: Use 'boundaryMutation’ ‘unifMutation' each 5 times in each generation. You may use one mutation operator only.

1) **Output variables** of gamultiple_var.m:

   sol -- the best variable values
   f -- the function value

2) The evaluate (fitness) function of this problem is defined in ‘gamultiple_vareval.m’.

Appendix:

Basic Commands and Simple Matlab Programming

A. Commonly Used Commands

Basic operations:

1) Programming in the **Edit window**. Save the programs as a **"*.m"** file in your working directory. Say "D:\ga\test.m".
2) Change the **current directory** into your working directory "D:\ga\" in **command window**.
3) Type the file name in the **command window** to **run the program**. Say "test".

Useful commands:

1) **help** – "help" + "function name" show the usage of the function;
2) **dir** – show the files in the current directory.
3) **who** – display the variables in the memory.
4) **clear all** – clear all data in the memory.
5) **save** – save the variables into a "*.mat " file.
6) **load** - load('path/filename') load the data named 'filename, in the 'path' directory.
7) **plot** – plot(X,Y) plots vector Y versus vector X.

B. Save Graphs

Use the menu in the figure window to save a graph.