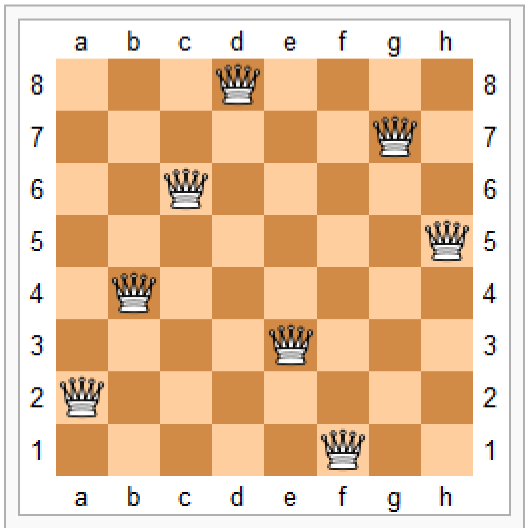
Project – Part A
Introduction to Artificial Intelligence
Winter 2013

Value of Part A: 12% (out of 20%)
Due date: Sunday, March 17, 11:59 PM
Team members: No more than three students in each group
Language: Matlab, C, C++, or Java
Marking values, Problem 1: 6%, Problem 2: 6%

Project Part A - Objectives

1) Understanding of Genetic Algorithm (GA) and Differential Evolution (DE)
2) Implementing GA, DE, and five benchmark functions
3) Reporting the comparative results properly and analyzing the results
4) Modeling and Solving a real-world problem using GA

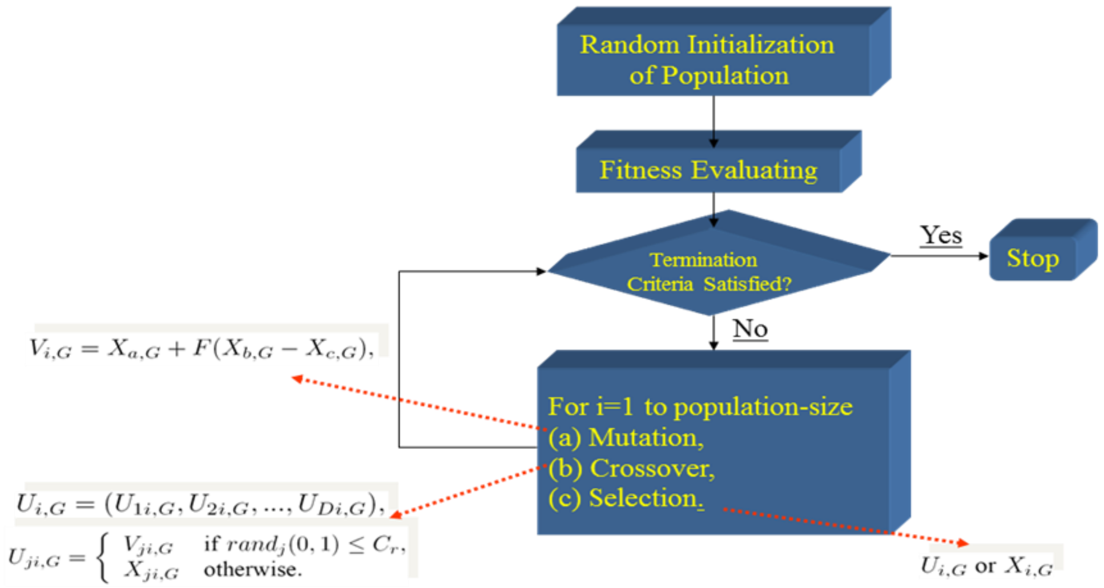
Problem 1) "The **eight queens puzzle** is the problem of placing eight chess queens on an 8×8 chessboard so that no two queens attack each other. Thus, a solution requires that no two queens share the same row, column, or diagonal". One sample solution is presented as follow: [Wikipedia].



One solution to the eight queens puzzle [Wikipedia]

In order to solve this problem with a Genetic Algorithm (GA), please design (model): **a) Structure of the corresponding chromosome, b) Fitness function, c) Crossover operator, and d) Mutation operator, then, solve the problem using the Basic Genetic Algorithm covered in your textbook and the lectures.** You need to report at least 10 different solutions obtained by your implemented GA.

Problem 2) Differential Evolution (DE): The Figure 1 presents the flowchart of DE algorithm. A numerical example for one iteration of the DE is provided in Figure 2. Algorithm 1 provides the pseudocode of the DE.



Random Initialization of Population
Fitness Evaluating
Termination Criteria Satisfied?
Yes
Stop
No
For i=1 to population-size
(a) Mutation,
(b) Crossover,
(c) Selection.
$V_{i,G} = X_{a,G} + F(X_{b,G} - X_{c,G}),$
$U_{i,G} = (U_{1i,G}, U_{2i,G}, ..., U_{Di,G}),$
$U_{ji,G} = \begin{cases} V_{ji,G} & \text{if } rand_j(0,1) \leq C_r, \\ X_{ji,G} & \text{otherwise.} \end{cases}$
$U_{i,G}$ or $X_{i,G}$

Figure 1: Flowchart of the DE.

Differential Evolution (DE)

Differential Evolution (DE) is a population-based and directed search method. Like other evolutionary algorithms, it starts with an initial population vector, which is randomly generated when no preliminary knowledge about the solution space is available.

Let us assume that $X_{i,G}(i = 1, 2, ..., N_p)$ are solution vectors in generation G (N_p :population size). Successive populations are generated by adding the weighted difference of two randomly selected vectors to a third randomly selected vector.

For classical DE (indicated by $DE/rand/1/bin$), the mutation, crossover, and selection operators are straightforwardly defined as follows:

Mutation - For each vector $X_{i,G}$ in generation G a mutant vector $V_{i,G}$ is defined by

$$V_{i,G} = X_{a,G} + F(X_{b,G} - X_{c,G}),$$

where $i = \{1, 2, ..., N_p\}$ and a, b, and c are mutually different random integer indices selected from $\{1, 2, ..., N_p\}$. Further, i, a, b, and c are different so that $N_p \geq 4$ is required. $F \in [0, 2]$ is a real constant which determines the amplification of the added differential variation of $(X_{b,G} - X_{c,G})$. Larger values for F result in higher diversity in the generated population and lower values cause faster convergence.

Crossover - DE utilizes the crossover operation to generate new solutions by shuffling competing vectors and also to increase the diversity of the population. For the classical version of the DE ($DE/rand/1/bin$), the binary crossover (shown by 'bin' in the notation) is utilized. It defines the following trial vector:

$$U_{i,G} = (U_{1i,G}, U_{2i,G}, ..., U_{Di,G}),$$

where $j = 1, 2, ..., D$ (D : problem dimension) and

$$U_{ji,G} = \begin{cases} V_{ji,G} & \text{if } rand_j(0,1) \leq C_r \vee j = k, \\ X_{ji,G} & \text{otherwise.} \end{cases}$$

$C_r \in (0, 1)$ is the predefined crossover rate constant, and $rand_j(0, 1)$ is the j^{th} evaluation of a uniform random number generator. $k \in \{1, 2, ..., D\}$ is a random parameter index, chosen once for each i to make sure that at least one parameter is always selected from the mutated vector, $V_{ji,G}$. Most popular values for C_r are in the range of $(0.4, 1)$.

Selection - The approach that must decide which vector ($U_{i,G}$ or $X_{i,G}$) should be a member of the next (new) generation, $G + 1$. For a maximization problem, the vector with the higher fitness value is chosen. There are other variants based on different mutation and crossover strategies .

Numerical Example of Differential Evolution (DE)

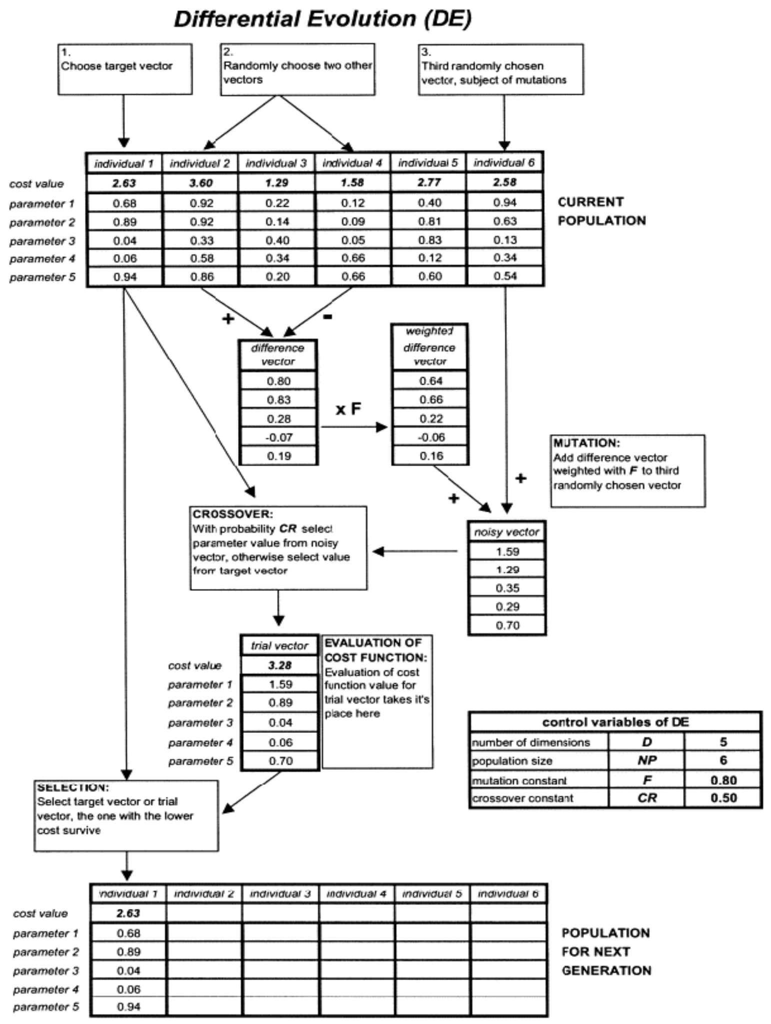
min f(x)=$X_1+X_2+X_3+X_4+X_5$

$$V_{i,G+1} = X_{r_1,G} + F(X_{r_2,G} - X_{r_3,G})$$

$$\vec{U}_{G,i} = \{U_{G,i,1}, U_{G,i,2}, ..., U_{G,i,D}\}$$

$$U_{G,i,j} = \begin{cases} V_{G,i,j}, & rand_j(0,1) \le CR \text{ or } j = j_{rand}, \\ X_{G,i,j}, & otherwise. \end{cases}$$

$$i = 1, 2, ..., NP \quad j = 1, 2, ..., D$$



1. Choose target vector

2. Randomly choose two other vectors

3. Third randomly chosen vector, subject of mutations

CURRENT POPULATION

	individual 1	individual 2	individual 3	individual 4	individual 5	individual 6
cost value	**2.63**	**3.60**	**1.29**	**1.58**	**2.77**	**2.58**
parameter 1	0.68	0.92	0.22	0.12	0.40	0.94
parameter 2	0.89	0.92	0.14	0.09	0.81	0.63
parameter 3	0.04	0.33	0.40	0.05	0.83	0.13
parameter 4	0.06	0.58	0.34	0.66	0.12	0.34
parameter 5	0.94	0.86	0.20	0.66	0.60	0.54

difference vector
0.80
0.83
0.28
-0.07
0.19

weighted difference vector
0.64
0.66
0.22
-0.06
0.16

MUTATION: Add difference vector weighted with ***F*** to third randomly chosen vector

noisy vector
1.59
1.29
0.35
0.29
0.70

CROSSOVER: With probability ***CR*** select parameter value from noisy vector, otherwise select value from target vector

	trial vector
cost value	**3.28**
parameter 1	1.59
parameter 2	0.89
parameter 3	0.04
parameter 4	0.06
parameter 5	0.70

EVALUATION OF COST FUNCTION: Evaluation of cost function value for trial vector takes it's place here

control variables of DE		
number of dimensions	***D***	**5**
population size	***NP***	**6**
mutation constant	***F***	**0.80**
crossover constant	***CR***	**0.50**

SELECTION: Select target vector or trial vector, the one with the lower cost survive

POPULATION FOR NEXT GENERATION

	individual 1	individual 2	individual 3	individual 4	individual 5	individual 6
cost value	**2.63**					
parameter 1	0.68					
parameter 2	0.89					
parameter 3	0.04					
parameter 4	0.06					
parameter 5	0.94					

Figure 2: A numerical example of DE.

Algorithm 1 Differential Evolution (DE). P_0: Initial population, N_p: Population size, V: Noise vector, U: Trial vector, D: Problem dimension,
NFC: Number of function calls, MAX_{NFC}: Maximum number of function calls, F: Mutation constant, $rand(0,1)$: Uniformly generated random number, C_r: Crossover rate, $f(\cdot)$: Objective function, P': Population of the next generation.

```
1:  Generate uniformly distributed random population $P_0$
2:  while ($\text{NFC} < \text{MAX}_{\text{NFC}}$ ) do
3:    //Generate-and-Test-Loop
4:    for $i = 0$ to $N_p$ do
5:      Select three parents $X_a$, $X_b$, and $X_c$ randomly from current population where
        $i \neq a \neq b \neq c$
        //Mutation
6:      $V_i \leftarrow X_a + F \times (X_c - X_b)$
        //Crossover
7:      for $j = 0$ to $D$ do
8:        if $rand(0,1) < C_r$ then
9:          $U_{i,j} \leftarrow V_{i,j}$
10:       else
11:         $U_{i,j} \leftarrow X_{i,j}$
12:       end if
13:     end for
        //Selection
14:     Evaluate $U_i$
15:     if $(f(U_i) \leq f(X_i))$ then
16:       $X'_i \leftarrow U_i$
17:     else
18:       $X'_i \leftarrow X_i$
19:     end if
20:   end for
21:   $X \leftarrow X'$
22: end while
```

Benchmark Functions for testing DE: The followings are five benchmark functions which you need to minimize them using DE. For more information visit: http://www.geatbx.com/docu/fcnindex-01.html#P89_3085.

- 1^{st} *De Jong*

$$f_1(X) = \sum_{i=1}^{n} {x_i}^2,$$
$$\text{with} - 5.12 \le x_i \le 5.12,$$
$$min(f_1) = f_1(0, ..., 0) = 0.$$

Unimodal, scalable, convex, and easy function.

- *Axis Parallel Hyper-Ellipsoid*

$$f_2(X) = \sum_{i=1}^{n} i{x_i}^2,$$
$$\text{with} - 5.12 \le x_i \le 5.12,$$
$$min(f_2) = f_2(0, ..., 0) = 0.$$

Unimodal, scalable, convex, and easy function.

- *Schwefel's Problem 1.2*

$$f_3(X) = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_j \right)^2,$$
$$\text{with} - 65 \le x_i \le 65,$$
$$min(f_3) = f_3(0, ..., 0) = 0.$$

Unimodal and scalable function.

- *Rosenbrock's Valley*

$$f_4(X) = \sum_{i=1}^{n-1} \left[100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2 \right],$$
$$\text{with} - 2 \le x_i \le 2,$$
$$min(f_4) = f_4(1, ..., 1) = 0.$$

This function is known as the Banana function. It is a non-convex unimodal classic optimization problem. Locating the minimum is very challenging for many optimizers, because the optimum is inside a long, narrow, parabolic shaped flat valley.

- *Rastrigin's Function*

$$f_5(X) = 10n + \sum_{i=1}^{n} (x_i^2 - 10\cos(2\pi x_i)),$$
$$\text{with} - 5.12 \le x_i \le 5.12,$$
$$min(f_5) = f_5(0, ..., 0) = 0.$$

Parameter settings for DE

Where n indicates the problem's dimensionality.

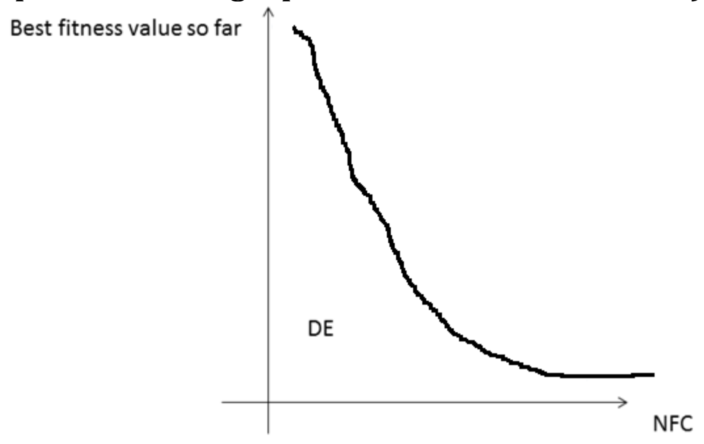
- Problems' dimensionality: $n=30$
- Population size: $N_p= 2*n$
- Mutation constant: $F=0.5$
- Crossover rate: $C_r = 0.9$
- Maximum number of function calls: $MAX_{NFC}= 1000*n$
- Number of runs per algorithm per function: 50

Results which should be reported

a) Solving the mentioned five benchmark functions and reporting results in the following table:

Benchmark Function	Average best fitness value and Standard Deviation (over 50 runs)
f_1	
f_2	
f_3	
f_4	
f_5	

b) The following performance graph (best fitness value so far Vs. number of function calls) for DE for all functions of previous experiments. (Hint: you need to have 5 performance graphs, one for each function).



What do you need to submit as a zipped folder?

1) Your implementations' source codes and execution file for each (if any) in two separate folders called "GA and 8-Queen problem", "DE and Benchmark Functions" and a readme file to say how to run your program.

2) Your report as a PDF file, including results (the table and graphs) and corresponding result analysis and conclusion.