Contents

- One-Max Genetic Algorithm Examples
- Initialize Population
- Evolution Loop
- Evaluate Population
- Selection
- Recombination
- Crossover
- Mutation
- Elitism
- Early Termination
- Plot Result

One-Max Genetic Algorithm Examples

```
clear
```

Set Parameters

```
popSize = 10;
nGenes  = 8;
rateMut = 1/nGenes;
rateXov = 0.9;
maxGens = 2;
```

Initialize Population

Initialize population randomly between zero and one

```
pop = randi([0 1],popSize,nGenes);
%figure(1);imagesc(pop);xlabel('Genes');ylabel('Individuals');title('Children')
```

Evolution Loop

Loop through evaluation, selection, and recombination for 'maxGens' generations or unless a termination criteria is reached

```
for gen = 1:maxGens
```

Evaluate Population

As we are merely counting the number of ones in each individuals genome we can use the 'sum' function as a fitness measure. We record the best fitness and median fitness to record performance, and keep the index of the best individual. This individual is inserted into the population unchanged at the end of recombination (Elitism).

```
fitness = sum(pop,2);
[bestFitness(gen), bestIndividual] = max(fitness);
medianFitness(gen) = median(fitness);
```

Selection

Parents are selected via tournament selection:

1. Two individuals are chosen for each breeding opportunity (2 spots per needed child)
2. The individuals in each matchup with the highest fitness is chosen as a parent for recombination

```
matchupA = randi(popSize,2,popSize);
winnerAbool  = fitness(matchupA(1,:)) > fitness(matchupA(2,:));
parentAindx  = [matchupA(1,winnerAbool) matchupA(2,~winnerAbool)];

matchupB = randi(popSize,2,popSize);
winnerBbool  = fitness(matchupB(1,:)) > fitness(matchupB(2,:));
parentBindx  = [matchupA(1,winnerBbool) matchupA(2,~winnerBbool)];
```

Recombination

Create new population by combining and altering chosen parents

Crossover

Combine parents via single point crossover

```
   % Determine if parents will crossover or not
   doXover = (rand(1,popSize) < rateXov);

   % Determine Crossover Points
   crossPt = randi([1 nGenes-1],1,popSize) .* doXover;

   % Create Children Individuals
   indIndx = 1:popSize;
   newPop(indIndx,:) = [pop(parentAindx(indIndx), 1:end-crossPt(indIndx) )...
                        pop(parentBindx(indIndx), 1+end-crossPt(indIndx):end)];
```

Mutation

Possible bit flip for every gene

```
   mutation = rand(popSize,nGenes) < rateMut;
   newPop = xor(newPop, mutation);
```

Elitism

```
newPop(1,:) = pop(bestIndividual,:);
pop = newPop;
fitness = sum(newPop,2);

% View Parents and Children
% subplot(1,3,1);imagesc(pop(parentAindx,:));xlabel('Genes');ylabel('Individuals');title('ParentsA')
% subplot(1,3,2);imagesc(pop(parentBindx,:));xlabel('Genes');ylabel('Individuals');title('ParentsB')
% subplot(1,3,3);imagesc(pop);xlabel('Genes');ylabel('Individuals');title('Children')
% pause(0.1)
```

Early Termination

```
if bestFitness(gen) == nGenes; break; end
```

```
end
```

Plot Result

figure(2);clf; plot(bestFitness);hold on; plot(medianFitness); xlabel('Generations');ylabel('Fitness'); legend('Max Fitness', 'Median Fitness','Location','SouthEast')

Published with MATLAB® R2015b