Artificial Intelligence and Evolutionary Computing

Lecture 8

IT/T/414A

Winter Semester 2014

Instructor: Kartick Ch. MONDAL

Evolutionary Programming

Difference between various names:

- What is a Genetic Algorithm?
  - A string of 1’s and 0’s to encode different solutions in the form of vectors of values or parameters.
- What is Genetic Programming?
  - Evolving LISP programs using the GA principle.
- What is Evolutionary Programming?
  - So… at the end, nearly any type of computing tool could be “evolved” in some way using the GA principles
  - The most generic term came out.

EP vs. GA

- GA requires the solution attempt to be encoded in a string of values
  - genome
- EP uses whatever representation fits the problem
- Mutation in EP is a normally distributed perturbation
- Has infrequent large changes, frequent small changes
- Mutation rate decays as run time elapses
- GA mutation tends to be fixed size changes that create entirely new values

EP vs. ES

- EP and ES are very similar
- Major differences are selection and recombination
- Selection in EP is stochastic
  - tournament based
  - randomly selected participants
- Selection in ES is deterministic
  - bad individuals are purged
  - good individuals breed
  - no randomness involved
- No recombination is used in EP
- Multi-individual ES uses recombination

Evolutionary Programming

- Invented early 1960s in the USA
- Created by Lawrence Fogel
- Regarded artificial intelligence as the ability to predict a symbol based on previous symbols
- Evolved a population of finite state automata to perform this prediction

Evolutionary Programming

- Evolution consisted of adding, modifying or deleting state transitions
- Task was to predict characters from streams of characters
- FSA with the least number of errors were allowed to reproduce

Evolutionary Programming

- Resembles ES, developed independently
- Early versions of EP applied to the evolution of transition table of finite state machines
- One population of solutions, reproduction is by mutation only
- Like ES operates on the decision variable of the problem directly (ie **Genotype = Phenotype**)
- Tournament selection of parents
  - better fitness more likely a parent
  - children generated until population doubled in size
  - everyone evaluated and the half of population with lowest fitness deleted.

Evolutionary Programming

- There is no fixed structure for representation.
- There is only mutation operation, and cross-over is not used in this method.
- Each child is determined by its parent in a way of mutation.
- So, we can conclude that there are three steps:
  - Initialize population and calculate fitness values for initial population
  - Mutate the parents and generate new population
  - Calculate fitness values of new generation and continue from the second step.

Evolutionary Programming

- Mutation is at a very critical point, because it is the only method which leads to the variation.
- Main application areas:
  - Cellular design problems.
  - Constraint optimization
  - ......
- Not a widely used evolutionary algorithm, because the variation between individuals is very small and the convergence speed is not enough.

Evolutionary Programming

1. Create a population of solutions
2. Evaluate each solution in the population
3. Select individuals to reproduce
- tournament selection
4. Mutate the reproduction population
5. Repeat 2 - 4 until stopping condition is reached

Evolutionary Programming

- Reproduction in EP is via mutation
- Mutation may be normally distributed
- No prescribed method of representation
  - Use whatever works for the problem
    - FSA, ANN etc
- Crossover / recombination is not used
- Not a widely used evolutionary algorithm, because the variation between individuals is very small and the convergence speed is not enough.

EP technical summary tableau

Representation	Real-valued vectors
Recombination	None
Mutation	Gaussian perturbation
Parent selection	Deterministic
Survivor selection	Probabilistic ($\mu+\mu$)
Specialty	Self-adaptation of mutation step sizes (in meta-EP)

Evolutionary Programming

- An Example Evolutionary Computation

```
Procedure EC{
  t = 0;
  Initialize P(t);
  Evaluate P(t);
  While (Not Done)
  {
    Parents(t) = Select_Parents(P(t));
    Offspring(t) = Procreate(Parents(t));
    Evaluate(Offspring(t));
    P(t+1)= Select_Survivors(P(t),Offspring(t));
    t = t + 1;
  }
```

Evolutionary Programming

- In EC, the replacement method could also be called Select_Survivors.
- Therefore, during this course, we will discuss two forms of selection:
  - Parent Selection (Select_Parents(Pop(t))), and
  - Survivor Selection (Select_Survivors(Pop(t), Offspring(t)).
- In EC, there are basically two types of survivor selection methods:
  - $(\mu+\lambda)$ and
  - (μ,λ)

Evolutionary Programming

- In $(\mu+\lambda)$ selection,
  - The population of μ individuals (where $\mu = |P(t)|$, the population size) are used to create λ offspring (where $\lambda = |Offspring(t)|$).
  - The best μ of $P(t) \cup Offspring(t)$ are selected to survive.

- How might a $(\mu+1)$-EC operate?

Evolutionary Programming

- In (μ,λ) selection,
  - The population of μ individuals (where $\mu = |P(t)|$, the population size) are used to create λ offspring (where $\lambda = |Offspring(t)|$).
  - The best μ of Offspring(t) are selected to survive.

- What constraints must be placed on λ by a (μ, λ)-EC?

Evolutionary Programming: Early EP

- Early evolutionary programs evolved populations of finite state machines in an effort to solve prediction problems [see http://www.natural-selection.com/Library/1998/Revisiting_EP.pdf].
- Early EP used 5 mutation operators:
  - Add a State,
  - Delete a State,
  - Mutate the Start State,
  - Mutate a Link, and
  - Mutate an output symbol.

Evolutionary Programming: Early EP

- The fitness of an individual machine was mean absolute error in predicting an output symbol given an input symbol.
- Each of the μ machines were allowed to create one offspring by applying 1of the 5 mutations presented earlier.
- The population of μ offspring machines were exposed to an environment, in the form of a string of s symbols.

Evolutionary Programming: Early EP

- The best μ of the 2 μ machines were selected to survive.
  - For $\mu > 9$, q-tournament selection was applied to develop a subjective fitness. In this form of selection:
    - Each individual competes with q (typically less than 10) randomly selected machines,
    - The subjective fitness for a machine in the number of individuals (of the q randomly selected individuals) that it is better than.
- The process of procreation, evaluation, and selection was repeated for a total of 5 generations.

Evolutionary Programming: Early EP

- After every 5 generations the best machine was allowed to predict the (s+1) symbol. After the prediction, the true (s+1) was added to the environment. This process was repeated for M-1 cycles (with the final length of the environment being (s+M)).

Evolutionary Programming: Standard EP

- David B. Fogel adapted EP for solving parameter optimization problems in the late 1980’s (Bäck 1996).
- Standard EP, like its predecessor, attempts to solve minimization problems.
- Let’s take a look at it!

Evolutionary Programming: Standard EP

```
Procedure standardEP{
     t = 0;
     Initialize P(t); /* of μ individuals */
     Evaluate P(t);
     while (t <= (4000-μ)/μ){
       for (i=0; i<μ; i++){
         Create_Offspring(<x_i,y_i>,<x_μ+i,y_μ+i>):
            x_μ+i = x_i + sqrt(fit_i)N_x(0,1);
            y_μ+i = y_i + sqrt(fit_i)N_y(0,1);
         fit_μ+i = Evaluate(<x_μ+i,y_μ+i>);
       }
       Compute Subjective Fitness if μ ≥ 10;
       P(t+1) = Best μ of the 2μ individuals;
       t = t + 1;
     }
}
```

Evolutionary Programming: Continuous Standard EP

```
Procedure Continuous_standardEP{
    t = 0;
    Initialize P(t); /* of μ individuals */
    Evaluate P(t);
    while (t <= (4000-μ)){
      Create_Offspring(<x_i,y_i>,<x_μ,y_μ>):
          x_μ = x_i + sqrt(fit_i)N_x(0,1);
          y_μ = y_i + sqrt(fit_i)N_y(0,1);
      fit_μ = Evaluate(<x_μ,y_μ>);
      P(t+1) = Best μ of the μ+1 individuals;
      t = t + 1;
    }
}
```

Evolutionary Programming: Meta-EP

Procedure metaEP{

t = 0;

Initialize P(t); /* of μ individuals */

Evaluate P(t);

while (t <= $(4000-\mu)/\mu$){

for (i=0; i<μ; i++){

Create_Offspring($\langle x_i, y_i, \sigma_{i,x}, \sigma_{i,y}\rangle$, $\langle x_{\mu+i}, y_{\mu+i}, \sigma_{\mu+i,x}, \sigma_{\mu+i,y}\rangle$):

$x_{\mu+i} = x_i + \sigma_{i,x} N_x(0,1);$

$\sigma_{\mu+i,x} = \sigma_{i,x} + \eta\ \sigma_{i,x} N_x(0,1);$

$y_{\mu+i} = y_i + \sigma_{i,y} N_y(0,1);$

$\sigma_{\mu+i,y} = \sigma_{i,y} + \eta\ \sigma_{i,y} N_y(0,1);$

$fit_{\mu+i}$ = Evaluate($\langle x_{\mu+i}, y_{\mu+i}\rangle$);

}

Compute Subjective Fitness if $\mu \geq 10$;

P(t+1) = Best μ of the 2μ individuals;

t = t + 1;

}

}

Evolutionary Programming: Continuous Meta-EP

```
Procedure continuous_metaEP{
      t = 0;
      Initialize P(t); /* of μ individuals */
      Evaluate P(t);
      while (t <= (4000-μ)){
        Create_Offspring(<x_i,y_i,σ_i,x,σ_i,y>, <x_μ,y_μ,σ_μ,x,σ_μ,y>):

                x_μ   = x_i + σ_i,x N_x(0,1);
                σ_μ,x = σ_i,x + η σ_i,x N_x(0,1);

                y_μ = y_i + σ_i,y N_y(0,1);
                σ_μ,y = σ_i,y + η σ_i,y N_y(0,1);

                fit_μ = Evaluate(<x_μ,y_μ>);

            P(t+1) = Best μ of the μ+1 individuals;
            t = t + 1;
      }
}
```

Summary

- ES, EP and GP are all different kinds of evolutionary computation
- Each developed separately, but have common themes
- The boundaries between each are not clear-cut!
- Each have their own niches

	ES	EP	GA	GP
Representation	Real-valued	Real-valued	Binary-Valued	Lisp S-expressions
Self-Adaptation	Standard deviations and covariances	Variance	None	None
Fitness	Objective function values	Scaled objective function value	Scaled objective function value	Scaled objective function value
Mutation	Main operator	Only operator	Background operator	Background operator
Recombination	Different variants, important for self-adaptation	None	Main Operator	Main Operator
Selection	Deterministic extinctive	Probabilistic, extinctive	Probabilistic, preservative	Probabilistic, preservative

References

- Informed Search Algorithms slayts of the course CmpE 540.
- http://en.wikipedia.org/wiki/Evolutionary_algorithm
- http://www.cs.sandia.gov/opt/survey/ea.html
- http://www.faqs.org/faqs/ai-faq/genetic/part2/section-3.html
- http://en.wikipedia.org/wiki/Genetic_programming
- http://alphard.ethz.ch/gerber/approx/default.html
- http://en.wikipedia.org/wiki/Evolutionary_programming
- http://en.wikipedia.org/wiki/Genetic_algorithm
- http://homepage.sunrise.ch/homepage/pglaus/gentore.htm
- http://www.ads.tuwien.ac.at/raidl/tspga/TSPGA.html
- http://en.wikipedia.org/wiki/Evolution_strategy

General Architecture of Evolutionary Algorithms