Introduction to Evolutionary Computation

This material is adapted from a presentation constructed by Gerry Dozier at Auburn University
Introduction to Evolutionary Computation

- Evolutionary Computation is the field of study devoted to the design, development, and analysis of problem solvers based on natural selection (simulated evolution).
- Evolution has proven to be a powerful search process.
- Evolutionary Computation has been successfully applied to a wide range of problems including:
  - Aircraft Design,
  - Routing in Communications Networks,
  - Tracking Windshear,
  - Game Playing (Checkers [Fogel])
Introduction to Evolutionary Computation
(Applications cont.)

• Robotics,
• Air Traffic Control,
• Design,
• Scheduling,
• Machine Learning,
• Pattern Recognition,
• Job Shop Scheduling,
• VLSI Circuit Layout,
• Evolvable Hardware
Introduction to Evolutionary Computation
(cont.)

- An Example Evolutionary Computation

Procedure EC{
    t = 0;
    Initialize Pop(t);
    Evaluate Pop(t);
    While (Not Done)
    {
        Parents(t) = Select_Parents(Pop(t));
        Offspring(t) = Procreate(Parents(t));
        Evaluate(Offspring(t));
        Pop(t+1) = Replace(Pop(t),Offspring(t));
        t = t + 1;
    }
}
Introduction to Evolutionary Computation (cont.)

- In an Evolutionary Computation, a population of candidate solutions (CSs) is randomly generated.
- Each of the CSs is evaluated and assigned a fitness based on a user specified evaluation function. The evaluation function is used to determine the ‘goodness’ of a CS.
- A number of individuals are then selected to be parents based on their fitness. The Select_Parents method must be one that balances the urge for selecting the best performing CSs with the need for population diversity.
The selected parents are then allowed to create a set of offspring which are evaluated and assigned a fitness using the same evaluation function defined by the user.

Finally, a decision must be made as to which individuals of the current population and the offspring population should be allowed to survive.
Introduction to Evolutionary Computation (cont.)

• Once a decision is made the survivors comprise the next generation (Pop(t+1)).
• This process of selecting parents based on their fitness, allowing them to create offspring, and replacing weaker members of the population is repeated for a user specified number of cycles.
• Stopping conditions for evolutionary search could be:
  – The discovery of an optimal or near optimal solution
  – Convergence on a single solution or set of similar solutions,
  – When the EC detects the problem has no feasible solution,
  – After a user-specified threshold has been reached, or
  – After a maximum number of cycles.
Introduction to Evolutionary Computation:
A Simple Example

- Let’s walk through a simple example!
- Let’s say you were asked to solve the following problem:
  - Maximize:
    - \( f_6(x,y) = 0.5 + \frac{(\sin(\sqrt{x^2+y^2}))^2 - 0.5)}{(1.0 + 0.001(x^2+y^2))^2} \)
  - Where \( x \) and \( y \) are taken from \([-100.0, 100.0]\]
  - You must find a solution that is greater than 0.99754, and
  - you can only evaluate a total of 4000 candidate solutions (CSs)
- This seems like a difficult problem. It would be nice if we could see what it looks like! This may help us determine a good algorithm for solving it.
Introduction to Evolutionary Computation: A Simple Example

A 3D view of $f_6(x,y)$:

$$0.5 + (\sin(\sqrt{x^2+y^2}))^2 - 0.5)/(1.0+0.001*(x^2+y^2))$$
Introduction to Evolutionary Computation: A Simple Example

- If we just look at only one dimension $f_6(x, 1.0)$
Introduction to Evolutionary Computation: A Simple Example

- Let’s develop a simple EC for solving this problem
- An individual (chromosome or CS)
  - \( <x_i, y_i> \)
  - \( \text{fit}_i = f_6(x_i, y_i) \)
Procedure simpleEC{
    t = 0;
    Initialize Pop(t); /* of P individuals */
    Evaluate Pop(t);
    while (t <= 4000-P){
        Select_Parent(<x_mom,y_mom>); /* Randomly */
        Select_Parent(<x_dad,y_dad>); /* Randomly */
        Create_Offspring(<x_kid,y_kid>):
            x_kid = rnd(x_mom, x_dad) + N_x(0,\sigma);
            y_kid = rnd(y_mom, y_dad) + N_y(0,\sigma);
        fit_kid = Evaluate(<x_kid,y_kid>);
        Pop(t+1) = Replace(worst,kid);{Pop(t)-{worst}}\{kid}
        t = t + 1;
    }
}
Introduction to Evolutionary Computation: 
Reading List

