On Differential Evolution

Alexander G. Ororbia II

Biologically-Inspired Intelligent Systems

CSCI-633

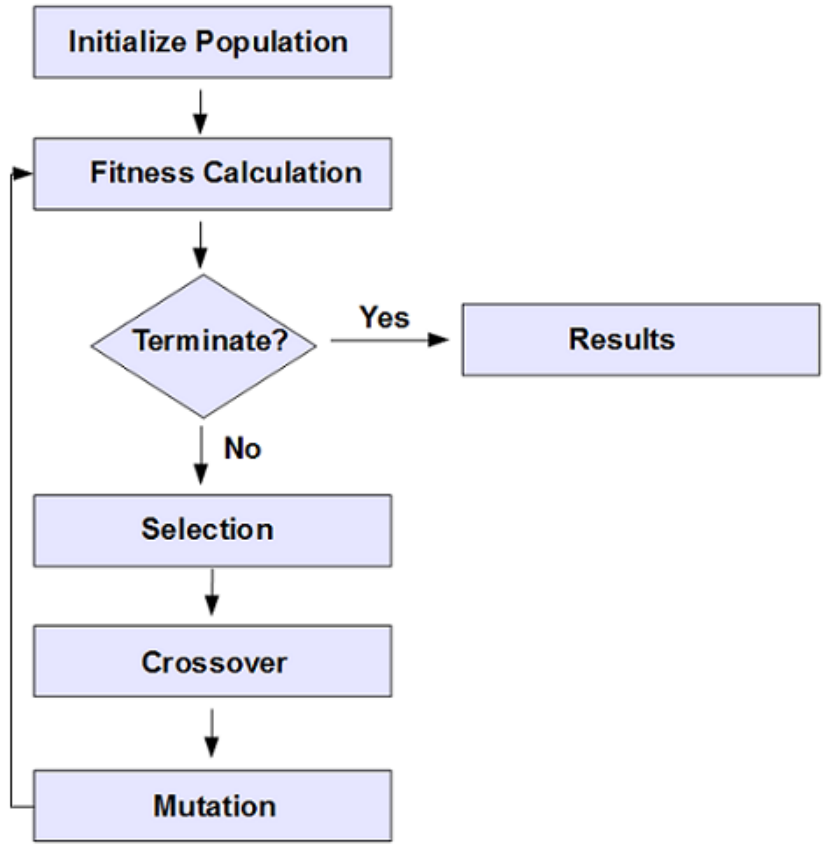
2/15/2024

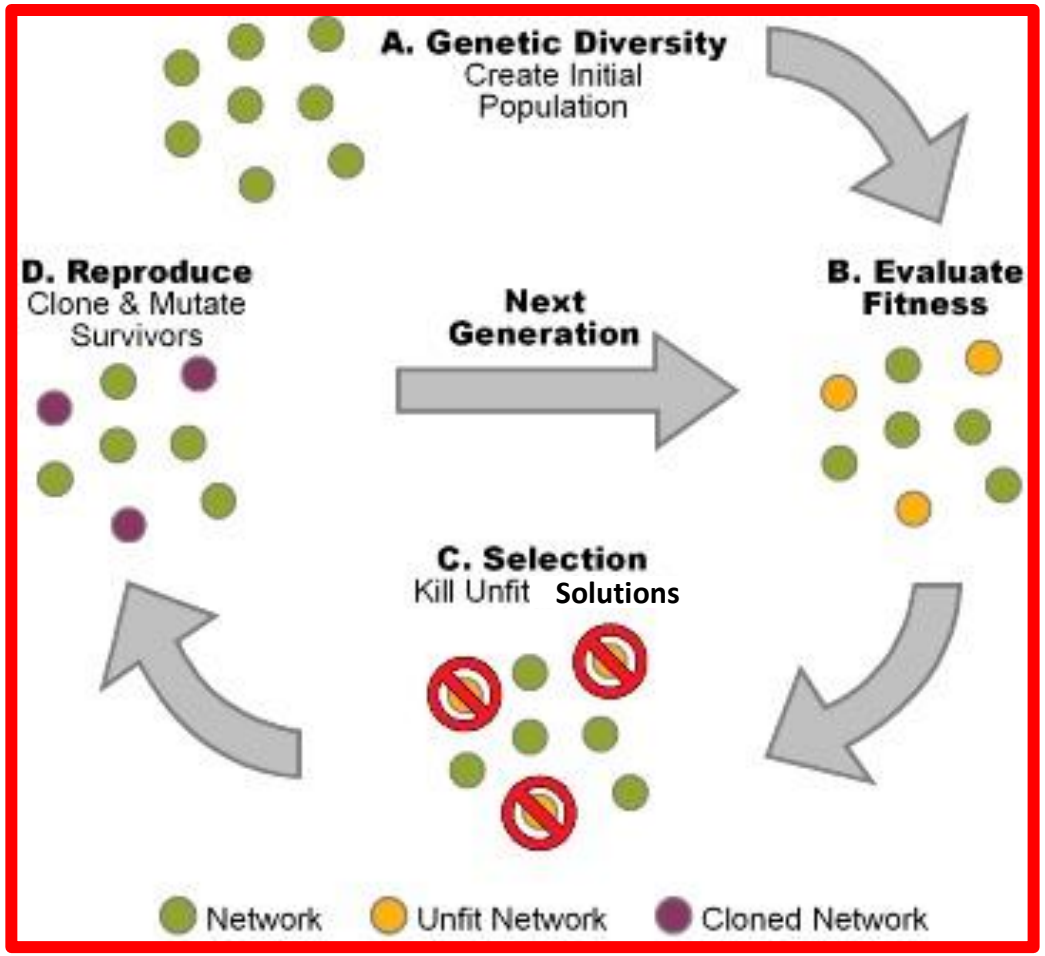
Evolutionary Computation through GAs

Genetic Algorithm

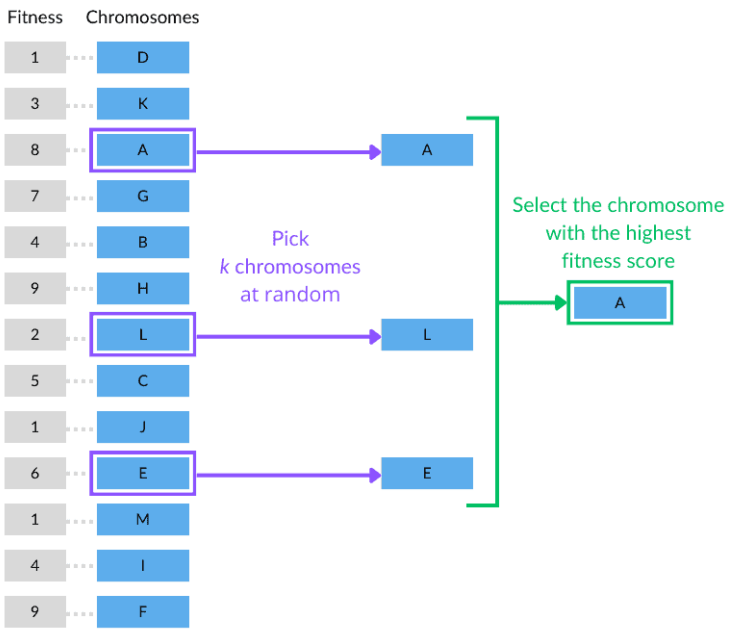
Objective function $f(x)$, $x = (x_1, ..., x_d)^T$
Encode the solutions into chromosomes (strings)
Define fitness F (eg, $F \propto f(x)$ for maximization)
Generate the initial population
Initialize the probabilities of crossover (p_c) and mutation (p_m)
    while (t <Max number of generations)
        Generate new solution by crossover and mutation
        Crossover with a crossover probability p_c
        Mutate with a mutation probability p_m
        Accept the new solutions if their fitness increase
        Select the current best for the next generation (elitism)
        Update $t = t + 1$
    end while
Decode the results and visualization

Genetic Algorithm (Process Flow)





Tournament Selection




```
/* Select the fittest individual (or chromosome) from a randomly selected list of individuals T */
algorithm Tournament(T, k):
    // INPUT
    //   T = a list of individuals randomly selected from a population.
    //   k = the tournament size. In other words, the number of elements in T.
    // OUTPUT
    //   the fittest individual.
    Best <- T[1];
    for i from 2 to k do
        Next <- T[i];
        if Fitness(Next) > Fitness(Best) then
            Best <- Next;
    return Best;
```

```
/* Assume we wish to select n individuals from the population P */
algorithm TournamentSelection(P, k, n):
    // INPUT
    //   P = the population.
    //   k = the tournament size, such that 1 ? k ? the number of individuals in P.
    //   n = the total number of individuals we wish to select.
    // OUTPUT
    //   the pool of individuals selected in the tournaments.
    T <- an empty sequence;
    B <- an empty sequence;
    for i from 1 to n do
        Pick k individuals from P at random, with or without replacement, and add them to T;
        B[i] <- Tournament(T, k);
        T <- [ ]
    return B;
```

```
Procedure Memetic Algorithm Based on an EA
Initialization: t = 0;  // Initialization of the generation counter
                Randomly generate an initial population P(t);
                Compute the fitness f(p) ∀p ∈ P(t);
while Stopping conditions are not satisfied do
    Selection: Accordingly to f(p) choose a subset of P(t) and store it in M(t);
    Offspring: Recombine and mutate individuals p ∈ M(t) and store them in M'(t);
    Learning: Improve p' by local search or heuristic ∀p' ∈ M'(t);
    Evaluation: Compute the fitness f(p') ∀p' ∈ M'(t);
    if Lamarckian learning then
       Update chromosome of p' according to improvement ∀p' ∈ M'(t);
    fi
    New generation: Generate P(t+1) by selecting some individuals from P(t) and M'(t);
    t = t + 1;  // Increment the generation counter
end while
Return best individual p ∈ P(t − 1) as result;
```

https://en.wikipedia.org/wiki/Memetic_algorithm#

Genetic Algorithms: Race Car Evolution

- https://www.youtube.com/watch?v=Aut32pR5PQA

Metaheuristic: Differential Evolution (DE)

- Vector-based (population-based) algorithm, Storn & Price (1996/1997)
  - Individuals ‘evolve’ by recombination w/ other individuals & differentials between other individuals
- Devised for continuous search spaces, derivative-free
- No encoding/decoding required – real numbers are solution strings/chromosomes
- *DE/rand/1/bin*

DE General Mechanics

- Builds on the idea of genetic algorithms
- Three primary steps:
  - Mutation, crossover, selection

- Name convention: DE/x/y/z – x is mutation scheme, e.g., random (Rand) or best (Best), y is number of difference vectors, z is crossover scheme, e.g., binomial (Bin) or exponential (Exp) or either/agnostic (*)
  - **We will start with**: DE/Rand/1/*

Questions?