Computational Intelligence

Unit # 5

Acknowledgement

- The slides of this lecture have been taken from the lecture slides of “CSE659 – Computational Intelligence” by Dr. Sajjad Haider.

EC vs Classical Optimization

EC vs Classical Optimization

- Classical optimization algorithms have been shown to be very successful (and more efficient than EAs) **in linear, quadratic, strongly convex, unimodal** and other specialized problems
- EAs have been shown to be more efficient for **discontinuous, nondifferentiable, and multimodal problems**.

EC vs Classical Optimization

- CO uses deterministic rules to move from one point in the search space to the next point. EC, on the other hand, uses probabilistic transition rules.
- The EC search starts from a diverse set of initial points, which allows parallel search of a large rea of the search space. CO starts from one point, successively adjusting this point to move toward the optimum.

EC vs Classical Optimization

- CO uses derivative information, usually first-order or second-order, of the search space to guide the path to the optimum.
- EC, on the other hand, uses no derivative information. Only the fitness values of individuals are used to guide the search.

Different EC paradigms

The different ways in which the EA components are implemented result in different EC paradigms:

- **Genetic algorithms** (GAs), which model genetic evolution.
- **Genetic programming** (GP), which is based on genetic algorithms, but individuals are programs (represented as trees).
- **Evolutionary programming** (EP), which is derived from the simulation of adaptive behavior in evolution (i.e. *phenotypic* evolution).
- **Evolution strategies** (ESs), which are geared toward modeling the strategic parameters that control variation in evolution, i.e. the evolution of evolution.
- **Differential evolution** (DE), which is similar to genetic algorithms, differing in the reproduction mechanism used.
- **Cultural evolution** (CE), which models the evolution of culture of a population and how the culture influences the genetic and phenotypic evolution of individuals.
- **Co-evolution** (CoE), where initially “dumb” individuals evolve through cooperation, or in competition with one another, acquiring the necessary characteristics to survive.

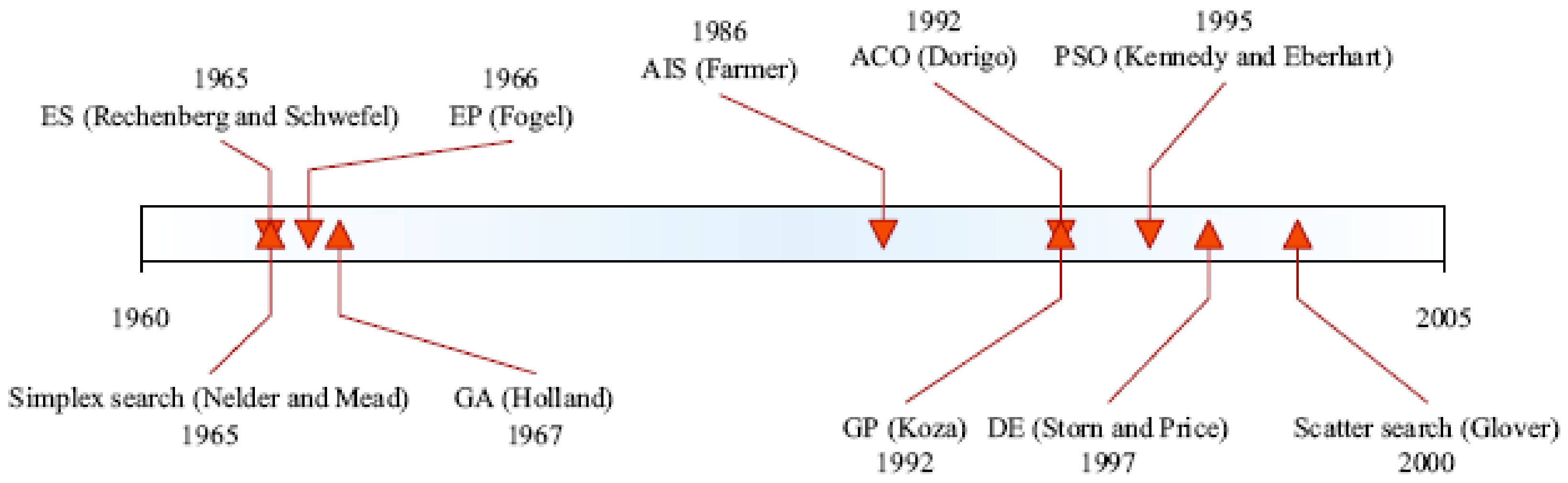
History of Evolutionary Algorithms

- Several efforts were started in parallel during 1960s
  - Evolutionary Programming (UCLA)
  - Evolution Strategies (Berlin Technical University)
  - Genetic Algorithms (University of Michigan)
- During 1990s the above communities agreed to the term “Evolutionary Computation”.

The Unifying 90s

- Till 90s, much of the R&D was done independently and in parallel without much interaction among the various groups.
- The emergence of the various EA conference in the late 1980s and early 1990s, however, changed all that as representative from various EA groups met, presented their particular viewpoints, challenged other approaches, and were challenged in return.
- The immediate result was an agreement on the term “evolutionary computation” as the name of the field and a commitment to start the field’s first journal, Evolutionary Computation.

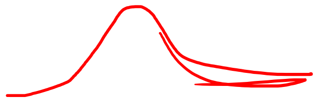
Recap: History of Evolutionary Computation



Evolutionary Programming

- At UCLA, Fogel (1966 paper) saw the potential of achieving the goals of artificial intelligence via evolutionary techniques.
- These ideas were initially explored in a context in which intelligent agents were represented as finite state machines, and an evolutionary framework called "*evolutionary programming*" was developed which was quite effective in evolving better finite state machines (agents) over time.

$x' = x + \underline{\Delta(t)}$



Canonical Evolutionary Programming

Population — m

o Mutation

- For I = 1 to m Do:
  - Use parent I to produce 1 child and add it to the offspring population
- End Do

$\overset{p}{m} - \overset{o}{m} \rightarrow \boxed{2m}$

- From the combined pool of 2m parents and children, select only the m individuals with highest fitness to survive.

Exploration
- Random initialization
- for loop (each parent)

Exploitation
- Survivor
- $\boxed{\text{Mutation}}$?

Gaussian

EP Differences

- Due to the above, EP does not make use of any recombination operator. There is no exchange of genetic material.
- Selection is based on competition. Those individuals that perform best against a group of competitors have a higher probability of being included in the next generation.
- Parents and offspring compete for survival.
- The behavior of individuals is influenced by strategy parameters, which determine the amount of variation between parents and offspring.

EP - Mutation

- As mutation is the only means of introducing variation in an EP population, it is very important that the design of a mutation operator considers the exploration–exploitation trade-off.
- The variation process should facilitate exploration in the early stages of the search to ensure that as much of the search space is covered as possible.
- After an initial exploration phase, individuals should be allowed to exploit obtained information about the search space to fine tune solutions.

EP – Mutation (Cont'd)

- In general, mutation is defined as

$$x'_i(t) = x_i(t) + \Delta x_i(t)$$

- where $x'_i(t)$ is the offspring created from parent $x_i(t)$ by adding a step size $\Delta x_i(t)$ to the parent.
- The step size is noise sampled from some probability distribution, where the deviation of the noise is determined by a strategy parameter, σ_i .
- Generally, the step size is calculated as

$$\Delta x_i(t) = \Phi(\sigma_i(t))\eta_i(t)$$

- where $\Phi : R \rightarrow R$ is a function that scales the contribution of the noise, $\eta_i(t)$.

Summary EP

Representation	Real-valued vectors
Parent Selection	Deterministic (each parent creates one offspring via mutation)
Recombination	None
Mutation	Gaussian perturbation
Survival Selection	Probabilistic (mu + mu)

η

Evolution Strategies

- At the Technical University Berlin, Rechenberg and Schwefel (1965 paper) began formulating ideas about how evolutionary processes could be used to solve difficult real-valued parameter optimization problems.
- From these early ideas emerged a family of algorithms called "*evolution strategies*" which today represent some of the most powerful evolutionary algorithms for function optimization.

Algorithm 12.1 Evolution Strategy Algorithm

```
Set the generation counter, $t = 0$;
Initialize the strategy parameters;
Create and initialize the population, $\mathcal{C}(0)$, of $\mu$ individuals;
for each individual, $\chi_i(t) \in \mathcal{C}(t)$ do
    Evaluate the fitness, $f(\mathbf{x}_i(t))$;
end
while stopping condition(s) not true do
    for $i = 1, \ldots, \lambda$ do
        Choose $\rho \geq 2$ parents at random;
        Create offspring through application of crossover operator on parent
        genotypes and strategy parameters;
        Mutate offspring strategy parameters and genotype;
        Evaluate the fitness of the offspring;
    end
    Select the new population, $\mathcal{C}(t+1)$;
    $t = t + 1$;
end
```

ES

$\overset{100}{\mu} + \overset{20}{\lambda}$

$(\mu+\lambda) \longrightarrow \mu$

$(\mu,\lambda) \longrightarrow \mu$

ES - Recombination

- The basic recombination scheme in evolution strategies involves two parents that create one child.
- To obtain λ offspring recombination is performed λ times.
- There are two recombination variants distinguished by the manner of recombining parent alleles. $(1.5, 3.5, 8) \times (2, 1, 6) \rightarrow (2, 3.5, 8)$
  - Using discrete recombination one of the parent alleles is randomly chosen with equal chance for either parents.
  - In intermediate recombination the values of the parent alleles are averaged.

ES – Step-Size Adjustment

- Theoretical studies motivated an on-line adjustment of step sizes by the famous 1/5 success rule of Rechenberg.
- This rule states that the ratio of successful mutations to all mutations should be 1/5.
- If the ratio is greater then 1/5 the step size should be increased to make a wider search of the space, and if the ratio is less than 1/5 then it should be decreased to concentrate the search more around the current solution.

ES - Survival Selection

- After creating λ offspring and calculating their fitness, the best μ of them are chosen deterministically, either from the offspring only, called $\underbrace{(\mu, \lambda)}$ selection, or from the union of the parents and offspring, called $(\mu + \lambda)$ selection.

generational

ES - Survival Selection (Cont'd)

- (μ, λ) is typically preferred over (μ + λ) for the following reasons
  - The $\underline{(\mu, \lambda)}$ discards all parents and is there in principle able to leave (small) local optima, so it is advantageous in the case of multimodal topologies
  - If the fitness function is not fixed, but changes in time, the (μ + λ) selection preserves outdated solutions, so it is not able to follow the moving optimum well.
  - (μ + λ) selection hinders the self-adaptation mechanism because mis-adapted parameter may survive for a relatively large number of generations

Summary of ES

Representation ✓	Real-valued vectors
Parent Selection ✓	Uniform random
Recombination	Discrete or intermediary
Mutation	Gaussian perturbation
Survival Selection	(mu, lambda) or (mu + lambda)

Genetic Algorithms

- At the University of Michigan, Holland (1962 paper) saw evolutionary processes as a key element in the design and implementation of robust adaptive systems, capable of dealing with an uncertain and changing environment.

Universal Genetic Code

- Holland emphasized the importance of a universal string-like “genetic” representation to be used internally by a GA to represent the genome of an individual.
- He initially suggested a binary string representation in which each bit internally is viewed as a gene, and the mapping to the external phenotype left unspecified and problem specific.

Genetic Algorithms

- In the previous schemes, individuals die off only when replaced by younger individuals with higher fitness.
- This results in a significant loss of diversity in the population and can increase the likelihood of becoming trapped on a false peak.
- One way to handle this problem is to allow new individuals to replace existing individuals with higher fitness.
- A more direct method is to use generational model in which parent survive for exactly one generation and are completely replaced by their offspring.
- **This is the form that standard GA take and also the form that (μ , λ) – ES model take.**

Genetic Algorithms (Cont'd)

- GA also implement the biological notion of fitness in a somewhat different fashion than we have seen so far (fitness proportional scheme).
- The EP and ES systems all used objective fitness to determine which offspring survive to adulthood.
- However, once in the parent pool, there is no additional bias with respect to which individuals get to reproduce – all parents have an equal chance.

Genetic Algorithms (Cont’d)

- Another important difference between GAs and the other models is the way in which offspring are produced.
- The basic idea is that offspring inherit gene values from more than one parent.
- This mixing of parental gene values along with an occasional mutation provides the potential for a much more aggressive exploration of the space.

Genetic Algorithms (Cont'd)

- Crossover provides an additional source of variation involving larger initial steps, improving the initial rate of convergence.
- Since crossover does not introduce new gene values, its influence diminishes as the population becomes more homogenous, and the behavior of GA with crossover becomes nearly identical to a GA with no crossover.

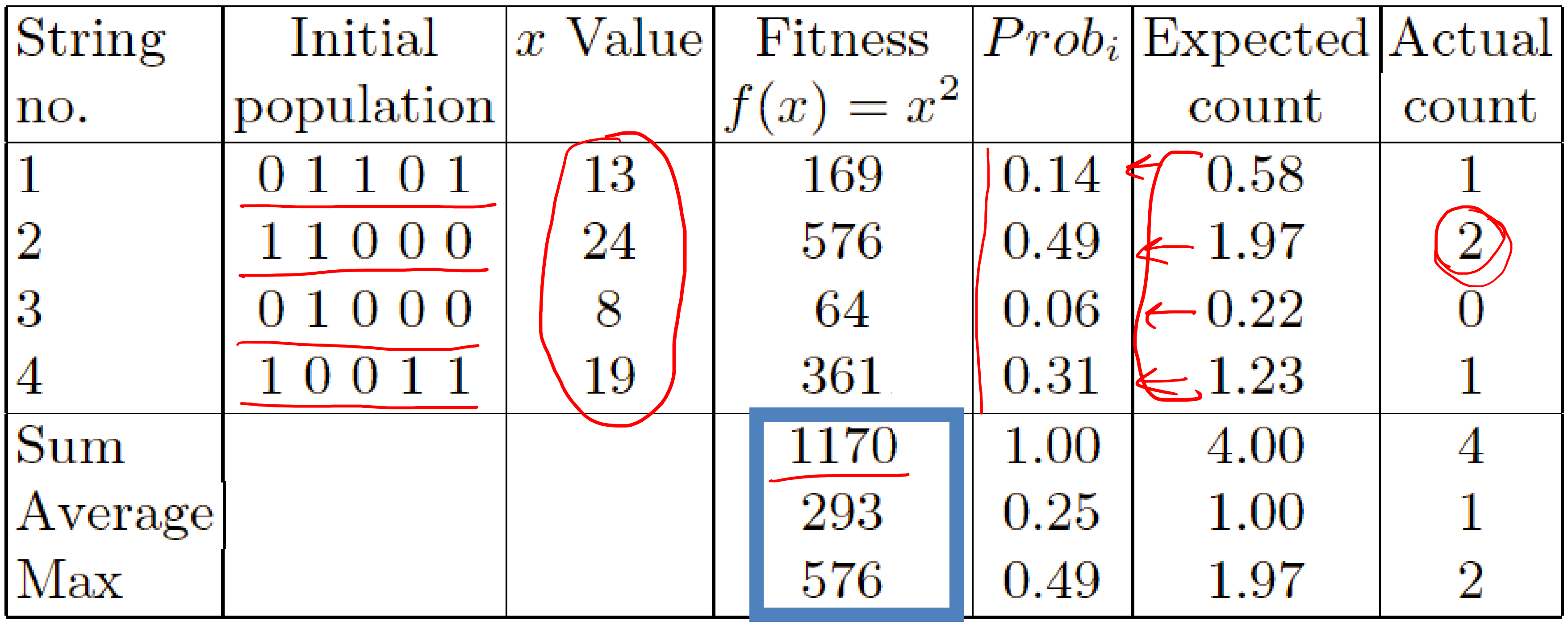
Example (Goldberg)

- Simple problem: max x^2 over $\{0,1,\ldots,31\}$

x

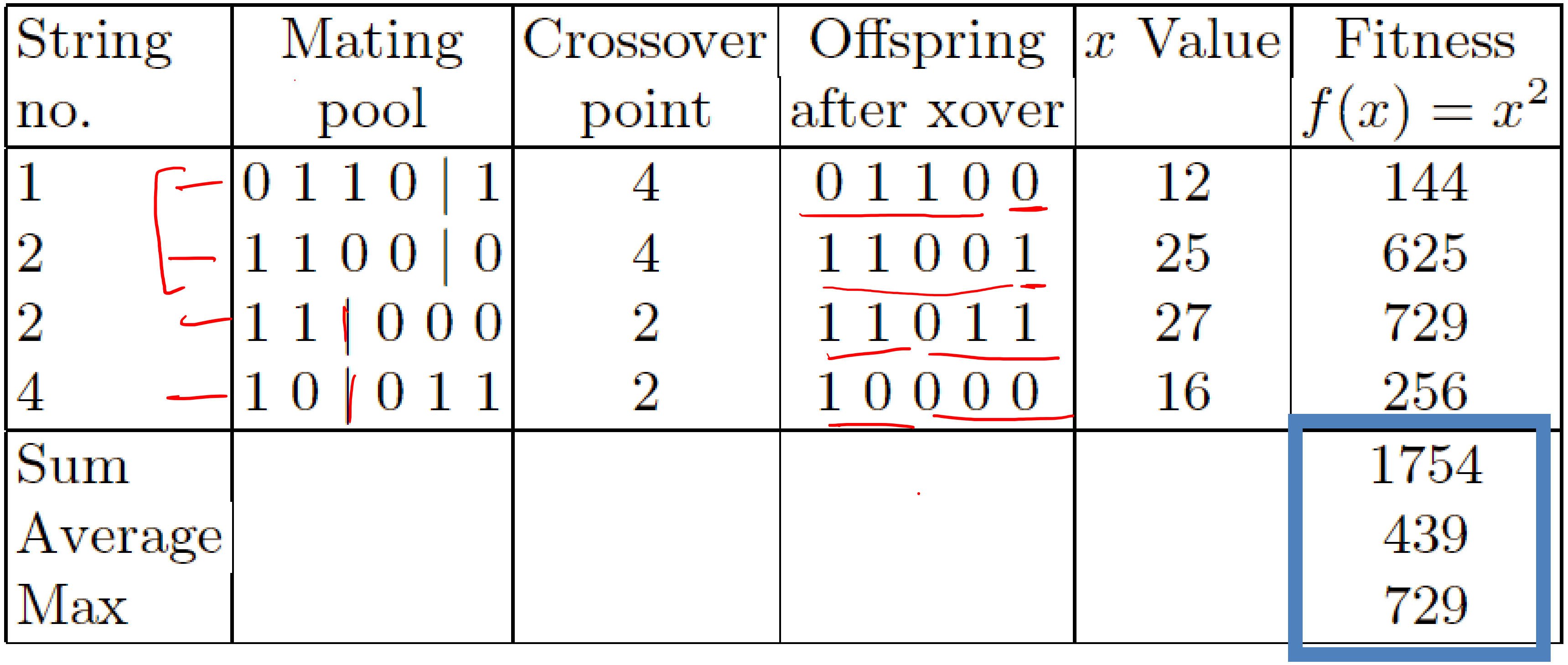
- GA approach:
  - Representation: binary code, e.g. 01101 $\leftrightarrow$ 13
  - Population size: 4
  - 1-point crossover, bitwise mutation
  - Roulette wheel selection
  - Random initialization
- We show one generational cycle done by hand

x^2 Example: Selection



String no.	Initial population	x Value	Fitness $f(x) = x^2$	$Prob_i$	Expected count	Actual count
1	0 1 1 0 1	13	169	0.14	0.58	1
2	1 1 0 0 0	24	576	0.49	1.97	2
3	0 1 0 0 0	8	64	0.06	0.22	0
4	1 0 0 1 1	19	361	0.31	1.23	1
Sum			1170	1.00	4.00	4
Average			293	0.25	1.00	1
Max			576	0.49	1.97	2

x^2 Example: Crossover



String no.	Mating pool	Crossover point	Offspring after xover	x Value	Fitness $f(x) = x^2$
1	0 1 1 0 | 1	4	0 1 1 0 0	12	144
2	1 1 0 0 | 0	4	1 1 0 0 1	25	625
2	1 1 | 0 0 0	2	1 1 0 1 1	27	729
4	1 0 | 0 1 1	2	1 0 0 0 0	16	256
Sum					1754
Average					439
Max					729

x^2 Example: Mutation

$ES(\mu, \underline{\lambda})$

String no.	Offspring after xover	Offspring after mutation	x Value	Fitness $f(x) = x^2$
1	0 1 1 0 0	$\boxed{1}$ 1 1 0 0	26	676
2	1 1 0 0 1	1 1 0 0 1	25	625
2	1 1 0 1 1	1 1 0 1 1	27	729
4	1 0 0 0 0	1 0 $\boxed{1}$ 0 0	18	324
Sum				2354
Average				588.5
Max				729

Summary of GA

Representation	Binary strings (Genotype)
Parent Selection	Fitness Proportional
Recombination	N-Crossover
Mutation	Bit-flip
Survival Selection	Generational