Computational Intelligence

Unit # 9

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.

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.
- Crossover and mutation operate as before at the gene level except at a much finer level of granularity.
- In the case of a binary representation, mutation simplifies to a "bit flipping" operator.

Example (Goldberg)

- Simple problem: max x^2 over {0,1,…,31}
- GA approach:
  - Representation: binary code, e.g. 01101 ↔ 13
  - Population size: 4
  - 1-point crossover, bitwise mutation
  - Roulette wheel selection
  - Random initialization
- We show one generational cycle done by hand

7

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

8

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

9

x^2 Example: Mutation

String no.	Offspring after xover	Offspring after mutation	x Value	Fitness $f(x) = x^2$
1	0 1 1 0 0	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 1 0 0	18	324
Sum				2354
Average				588.5
Max				729

10

Summary of GA

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