Genetic Algorithm

056: 166

Production Systems

Shital Shah

SPRING 2004

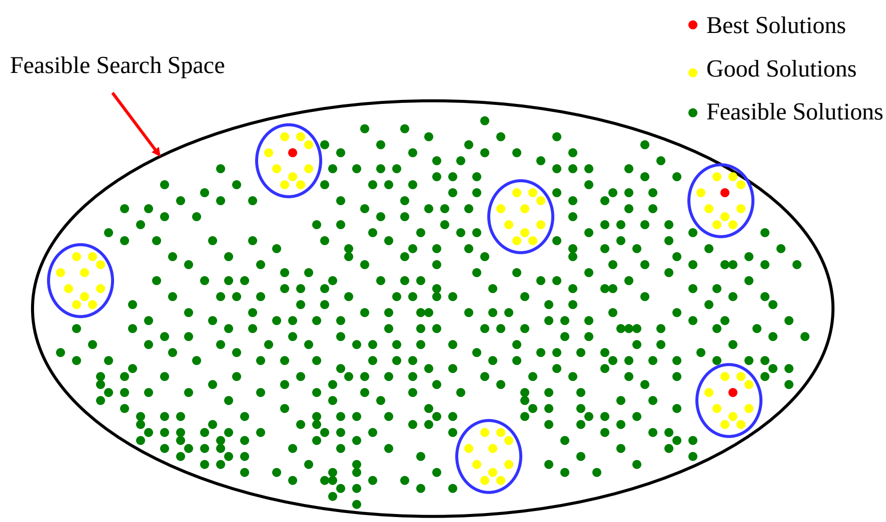
Outline

- Genetic Algorithm (GA)
  - Applications
  - Search space
- Step-by-step GA
  - Mechanism
  - Examples
- GA performance
- Other GA examples

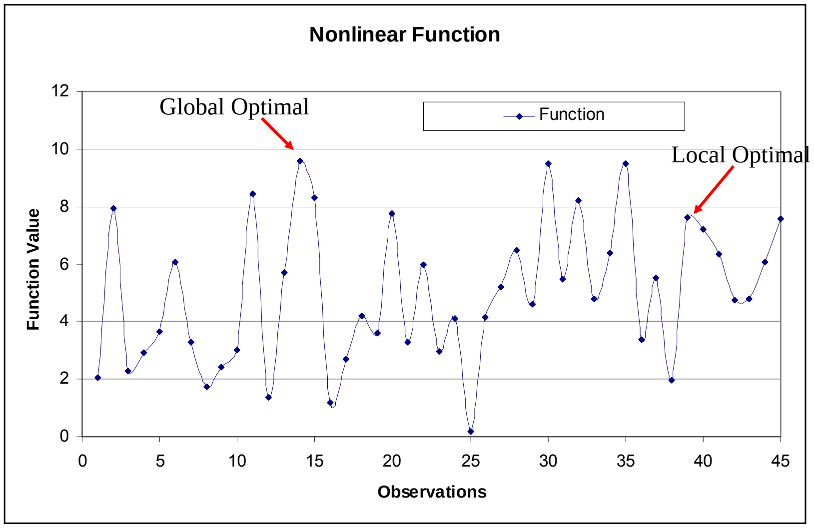
Genetic Algorithm (GA)

- GA's are search algorithms based on the mechanics of natural genetics
- GA's are a class of computer programs that use simulated evolution to solve problems
- Developed by
  - John Holland
  - David Goldberg
- Basic operations of GA
  - Selection
  - Crossover
  - Mutation

GA Search Space



Nonlinear Function Optimization



When to Apply GA

- Search space is large, complex or poorly understood
- Domain knowledge is scarce or expert knowledge is difficult to encode to narrow the search space
- No mathematical analysis is available
- Traditional search methods fail
- Optimization of NP-hard problems
  - Complex nonlinear optimization
  - Planning and scheduling problems
  - Resource allocation problems
  - Traveling salesman problem (TSP)
  - Layouts

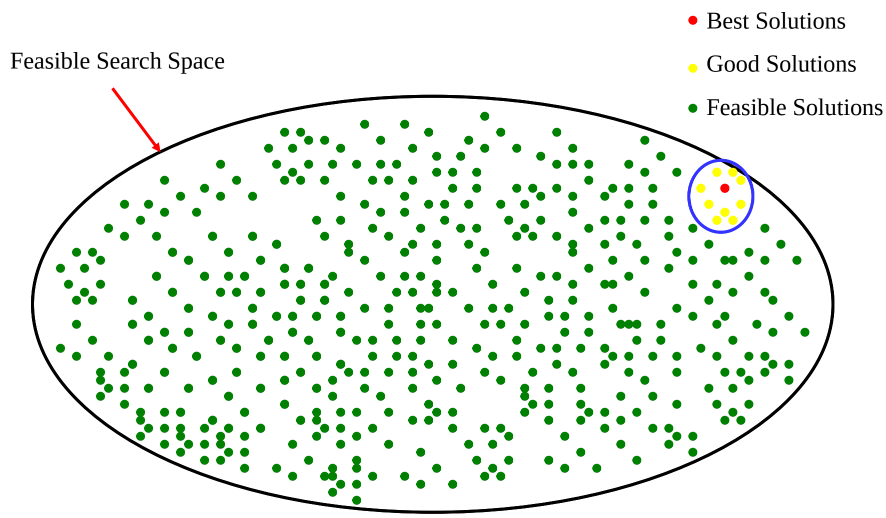
GA Applications

- All optimization problems
- Design problems (e.g. Gear designs)
- Maintenance and replacement policies
- Designing new musical tunes
- Development of control signatures
- Industry applications
  - Manufacturing
  - Airline and defense
  - Business and finance
  - Medical and pharmaceuticals
  - Power plants

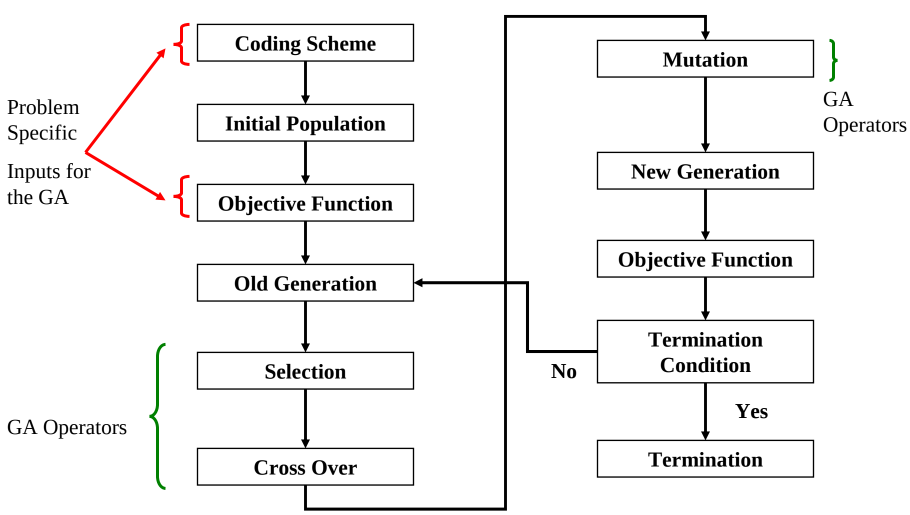
GA Example

- Maximize number of 1's in a binary search space of n = 5
- Binary search space is
  - "B1 B2 B3 B4 B5"
  - Possible solutions
    - "1 0 0 0 0", "1 0 1 0 0", "1 0 0 1 1", and so on
- The best solution is "1 1 1 1 1"

GA Example



GA Flow Chart



Coding Scheme

- Important step in GA formulation
  - Problem specific
- Contain information in some form about the solution
  - Forms Chromosomes → Potential solutions
- Search space with n bit positions
- Binary bit position → Either 0 or 1
- Coding scheme
  - “Bit1 Bit2 Bit3 Bit4 Bit5” → Chromosome
  - Possible solutions
    - “1 0 0 0 0”, “1 0 1 0 0”, “1 0 0 1 1”, and so on

Initial Population

- Initial population → Randomly created
  - Each bit is randomly chosen → {0,1}
  - User defined population size (10 – 1000)

Initial Population				
B1	B2	B3	B4	B5
0	0	0	1	0
1	1	0	1	0
0	1	0	1	0
1	0	1	0	1
1	0	0	0	0
1	1	0	0	0
0	1	0	0	0
0	1	0	0	1
1	0	0	1	1
1	1	0	0	1

GA Objective Function

- Most important step in GA formulation
  - Problem specific
  - Domain knowledge used
  - Provides a means to measure the quality of the generated solutions
  - Can optimize complex nonlinear function
- Maximize number of 1's in a binary search space of n = 5
- Objective function = ΣB_i $i = 1 \ldots 5$

GA Objective Function

Generation						Obj Function
B1	B2	B3	B4	B5	Sum (Bi)	
0	0	0	1	0	0+0+0+1+0	1
1	1	0	1	0	1+1+0+1+0	3
0	1	0	1	0	0+1+0+1+0	2
1	0	1	0	1	1+0+1+0+1	3
1	0	0	0	0	1+0+0+0+0	1
1	1	0	0	0	1+1+0+0+0	2
0	1	0	0	0	0+1+0+0+0	1
0	1	0	0	1	0+1+0+0+1	2
1	0	0	1	1	1+0+0+1+1	3
1	1	0	0	1	1+1+0+0+1	3

GA Selection

- Preference to better individuals to survive → Exploitation of knowledge
- Better objective function → Better individuals
- Various schemes
  - Roulette wheel selection
  - Tournament selection
- Provides two better chromosomes for other GA operators

GA Crossover

- Prime distinguishing factor of GA from other optimization techniques → Exploitation of knowledge
- Various Schemes
  - Single and double point crossover
  - Specialized crossover → TSP problem
- Single point crossover
  - Select two chromosomes
  - Randomly select a bit position
  - Slice the chromosomes at that position
  - Exchange the slice 2 of chromosome 1 with slice 2 of chromosome 2 and vice versa
  - Perform crossover with a probability (80%)

GA Cross Over

Selection					
B1	B2	B3	B4	B5	Obj Function
1	1	0	0	0	2
0	1	0	1	0	2

	Crossover Site					
	B1	B2	B3	B4	B5	Obj Function
Chromosome 1	1	1	0	0	0	2
Chromosome 2	0	1	0	1	0	2

Slice 1 Slice 2

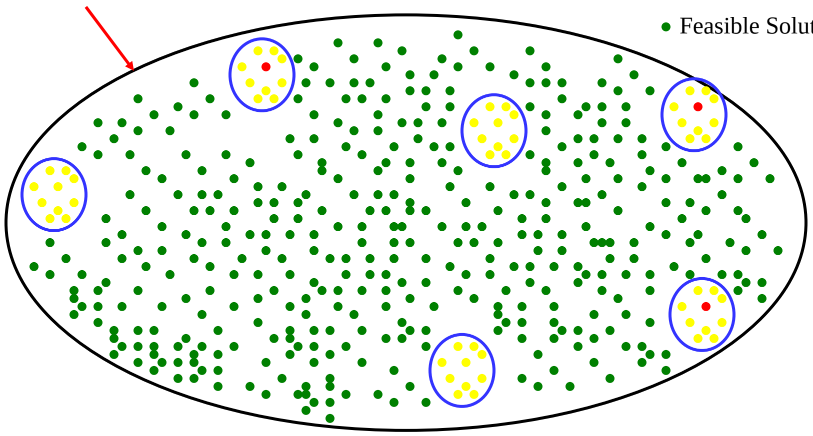
Crossover site = 3

Crossover					
B1	B2	B3	B4	B5	Obj Function
1	1	0	1	0	3
0	1	0	0	0	1

GA Search Space

Feasible Search Space

- Best Solutions
- Good Solutions
- Feasible Solutions



GA Mutation

- Maintains diversity within the population
- Inhibits premature convergence
- Provides mechanism for exploration of search space
- Special mutation operator needed for TSP
- Mutation operator
  - Select a chromosome after crossover
  - At each bit decide, with a probability, to change the bit value
  - Probability is generally low (0.001)

GA Mutation

After Crossover					
B1	B2	B3	B4	B5	Obj Function
1	1	0	1	0	3
0	1	0	0	0	1

Mutation Site					
B1	B2	B3	B4	B5	Obj Function
1	1	0	1	0	3
0	1	0	0	0	1

Bit positions chosen for mutation

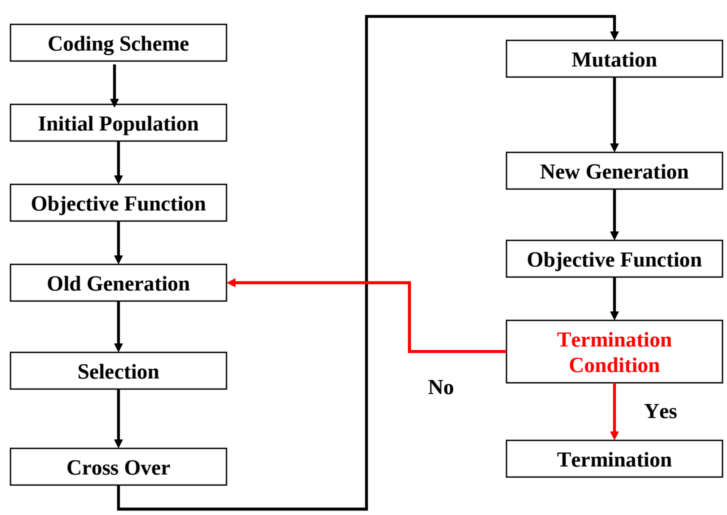
Mutation					
B1	B2	B3	B4	B5	Obj Function
1	1	1	1	0	4
0	0	0	0	0	0

GA New Population

- Repeat the selection, crossover, and mutation process till new population is formed

New Population					Objective Function
B1	B2	B3	B4	B5	
0	1	0	1	0	2
1	0	0	1	0	2
0	1	0	1	0	2
1	0	1	0	0	2
1	0	0	0	1	2
1	1	0	1	0	3
0	1	0	0	0	1
1	1	1	1	0	4
1	0	0	0	1	2
1	1	0	1	1	4

GA Flow Chart



GA Performance Measures

- On-line performance measure
  - Cumulative average of all objective function values of all chromosomes that have occurred in the evolution
- Off-line performance measure
  - Cumulative average of all objective function values of the best chromosomes that have occurred in each generation of the evolution

GA Performance

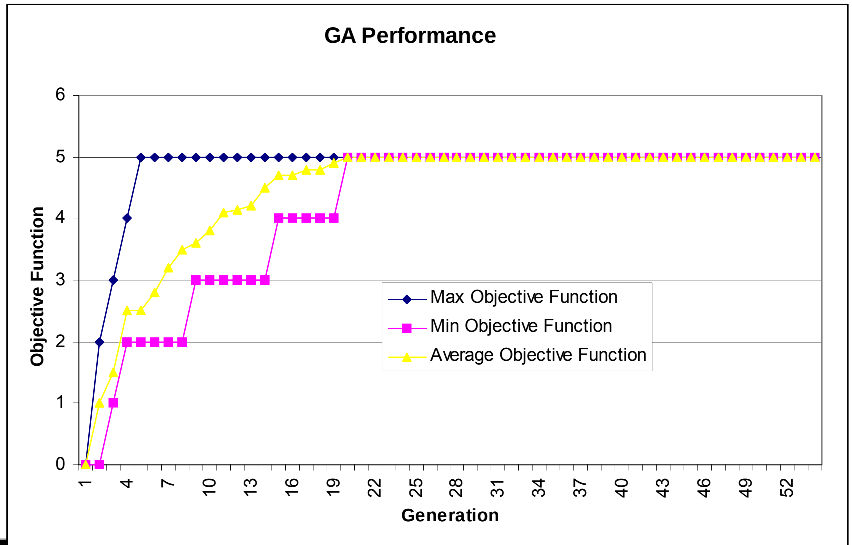
GA Generations →

Generation p					Obj
B1	B2	B3	B4	B5	Function
0	1	0	0	1	2
1	0	1	0	1	3
0	1	0	1	0	2
1	1	1	0	1	4
1	0	0	0	1	2
1	1	1	1	0	4
1	1	0	0	0	2
1	1	1	1	0	4
1	0	0	0	1	2
1	0	0	0	1	2
Min					2
Average					2.7
Max					4

Generation n					Obj
B1	B2	B3	B4	B5	Function
1	0	0	1	1	3
1	1	0	1	1	4
0	1	0	1	0	2
1	1	1	1	1	5
1	0	1	0	1	3
1	0	1	1	1	4
0	1	1	1	0	3
1	1	1	1	1	5
1	0	0	0	1	2
0	1	1	1	1	4
Min					2
Average					3.5
Max					5

Final Generation					Obj
B1	B2	B3	B4	B5	Function
1	1	1	1	1	5
1	1	1	1	1	5
0	1	1	1	0	3
1	1	1	1	1	5
1	0	1	1	1	4
1	1	1	1	1	5
1	1	1	1	0	4
1	1	1	1	1	5
1	0	1	1	1	4
1	1	1	1	1	5
Min					3
Average					4.5
Max					5

GA Performance



The University of Iowa

Intelligent Systems Laboratory

Assignment Problem

- Assignment Problem
- 5 machine – 5 jobs/parts
- Minimize the total cost of assignment

		Machines				
		M1	M2	M3	M4	M5
Jobs / Parts	J1	25	20	24	28	5
	J2	27	10	20	17	7
	J3	18	25	23	2	8
	J4	18	12	12	19	14
	J5	15	13	14	11	19
		Cij				
		C13 = 24				

The University of Iowa

Intelligent Systems Laboratory

Assignment Problem

$$Min = \sum C_{ij} * X_{ij}$$

$$\sum_i X_{ij} = 1 \qquad j = 1,2,......,n$$

$$\sum_j X_{ij} = 1 \qquad i = 1,2,......,n$$

$$X_{ij} = Binary \qquad \{0,1\}$$

GA Coding Scheme

	Machines Sequence Fixed						
	M1	M2	M3	M4	M5		
Chromosome 1	1	2	3	4	5	Jobs / Parts	Feasible
Chromosome 2	1	1	2	2	3		Infeasible
Chromosome 3	1	3	4	4	5		Infeasible
Chromosome 4	5	3	2	4	1		Feasible
Chromosome 5	2	4	5	1	3		Feasible
	Population						

GA Objective Function

- Minimize the total cost of assignment
- Take into account the infeasibility of the solutions
  - Apply special crossover and mutation operators
    - Replaces infeasible solutions with ONLY feasible solutions
  - Allow infeasibility in the population but apply a heavy penalty for each kind of infeasibility
    - Discourages infeasibility in the population

GA Objective Function

$$Min = \sum C_{ij} * X_{ij} \quad + \quad K * Infeasibility$$

$$K = 10000$$

		Machines				
		M1	M2	M3	M4	M5
Jobs / Parts	J1	25	20	24	28	5
	J2	27	10	20	17	7
	J3	18	25	23	2	8
	J4	18	12	12	19	14
	J5	15	13	14	11	19
		Cij				
		C13 = 24				

	M1	M2	M3	M4	M5	Obj Function		
Chromosome 1	1	2	3	4	5	25+10+23+19+19+1000*0	96	Feasible
Chromosome 2	1	1	2	2	3	25+20+20+17+8+1000*2	2090	Infeasible
Chromosome 3	1	3	4	4	5	25+25+12+19+19+1000*1	1100	Infeasible
Chromosome 4	5	3	2	4	1	15+25+20+19+5+1000*0	84	Feasible
Chromosome 5	2	4	5	1	3	27+12+14+28+8+1000*0	89	Feasible
Chromosome 6	2	4	5	3	1	27+12+14+2+5+1000*0	60	Feasible

Other GA Examples

- http://cs.felk.cvut.cz/~xobitko/ga/
- http://www.crhc.uiuc.edu/IGATE/ga_example/ga_example.html
- http://www4.ncsu.edu/eos/users/d/dhloughl/public/stable.htm
- http://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol4/tcw2/report.html

Questions

The University of Iowa
Intelligent Systems Laboratory