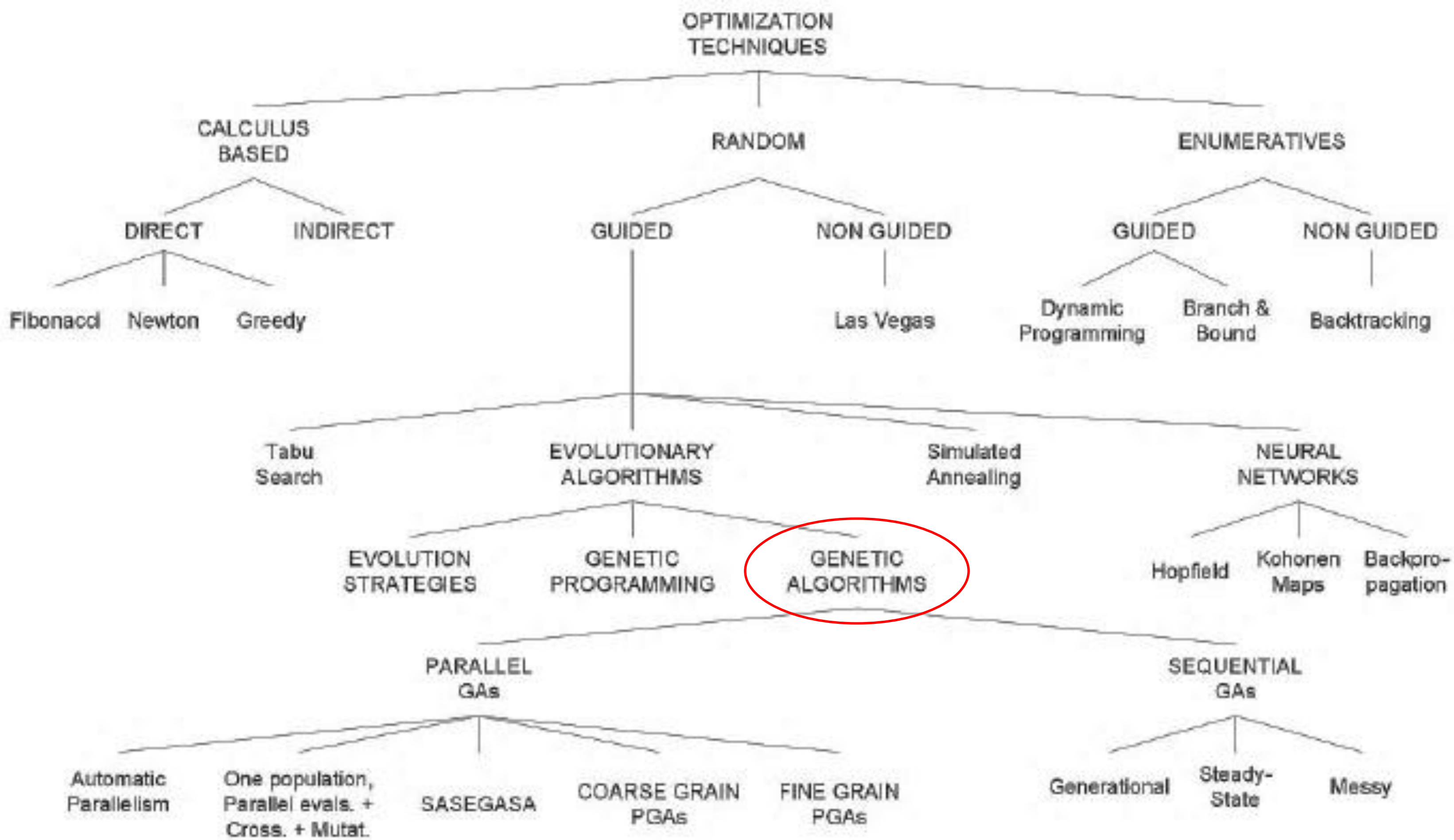
CMSC 691
High Performance Distributed Systems

Case study: Genetic Algorithms

Dr. Alberto Cano
Assistant Professor
Department of Computer Science
acano@vcu.edu

Taxonomy of optimization techniques



Genetic algorithms

- Originally developed by J. Holland, 1970s
- Nature-inspired methods based in heuristics and survival of the fittest via reproduction, crossover, mutation, and selection
- Genetic algorithms maintain a **population** of **individuals** (candidate solutions) ranked according to a **fitness** function, and evolve them by iteratively applying a set of stochastic **genetic operators**
- Provide approximate non-deterministic solutions
- Individual representation: binary, integer, numeric arrays
- Similar metaheuristics: ant colony optimization, artificial bee colony, penguins search optimization, particle swarm optimization, tabu search, genetic programming, etc

Simple genetic algorithm

Create an initial population of random individuals

Evaluate the fitness of all individuals

while termination condition not met do

select fitter individuals for reproduction

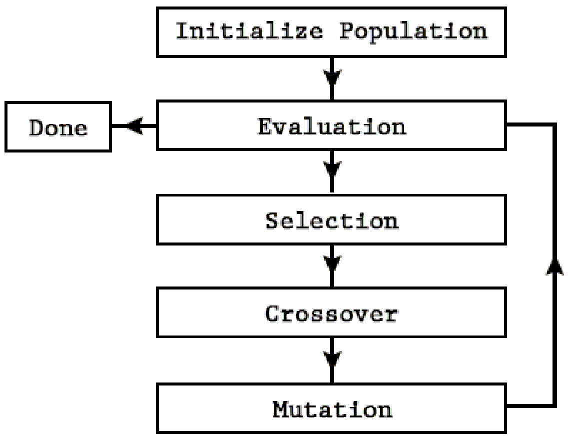
recombine between individuals (crossover)

mutate individuals

evaluate the fitness of the offspring individuals

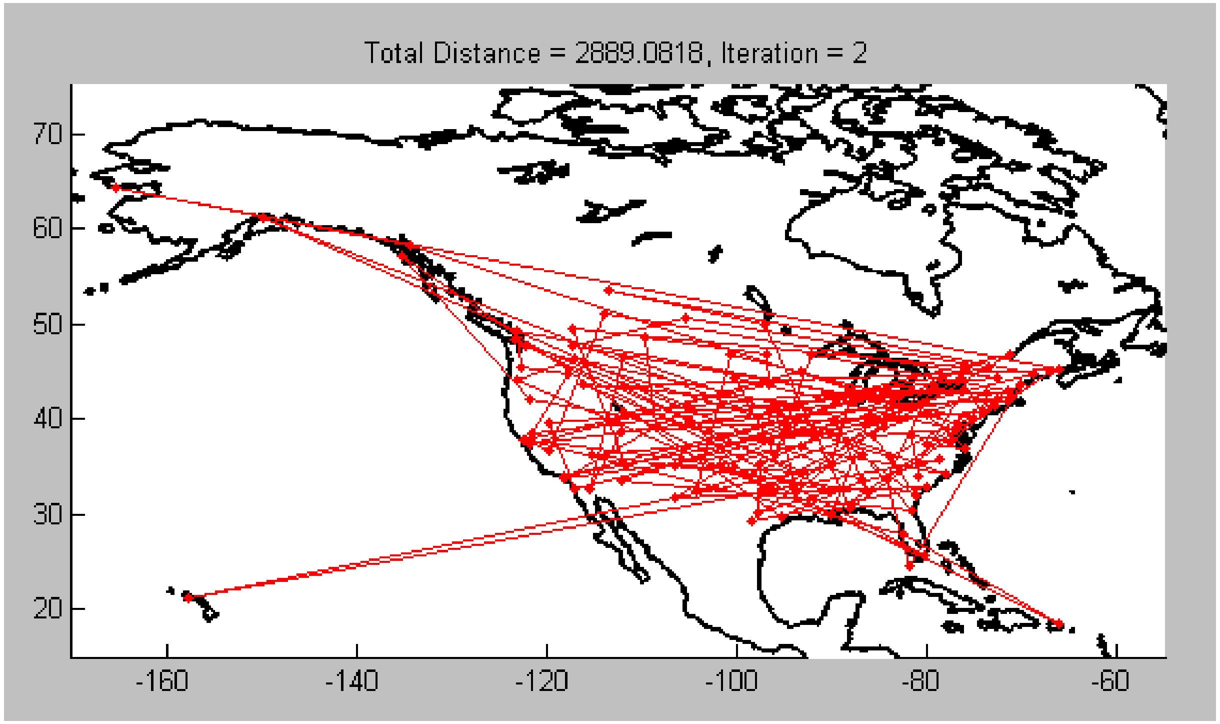
generate a new population keeping the best individuals

end while



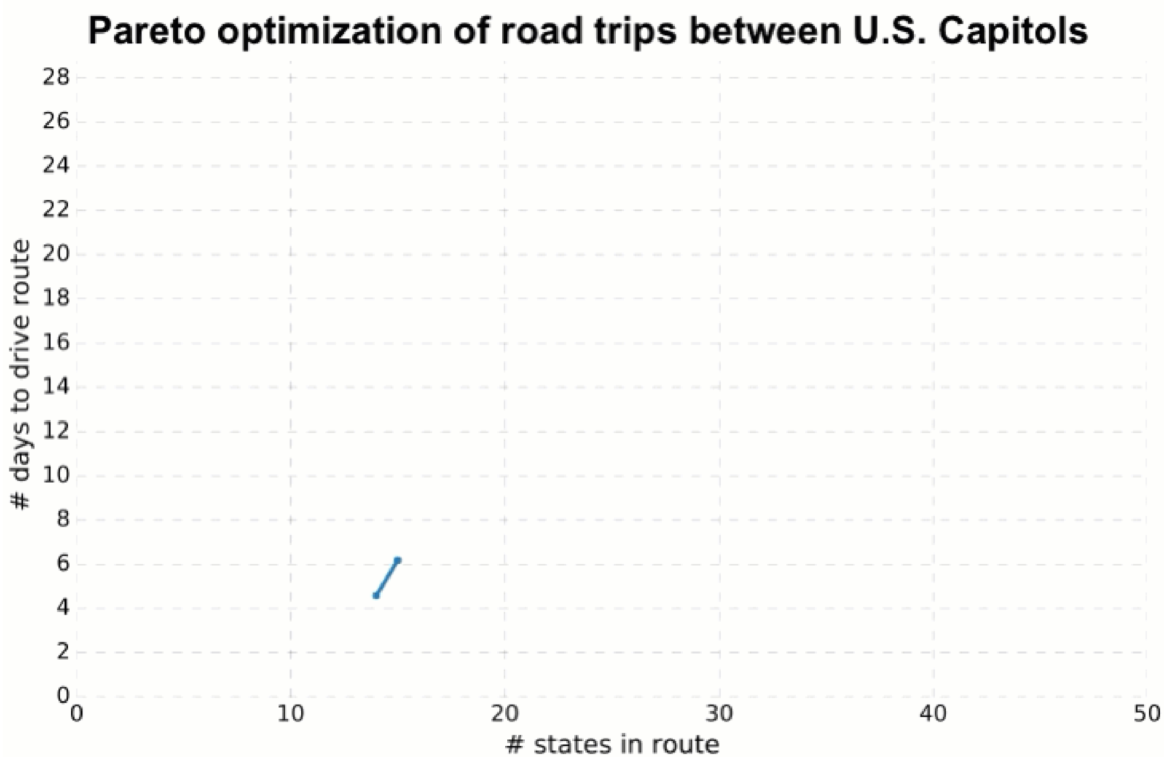
Example: Optimizing the Traveling Salesman Problem (NP-hard)

- Individual represents a permutation of the cities to visit
- Fitness function: minimize the total distance (single objective)
- We may implement restrictions to the problem very easily



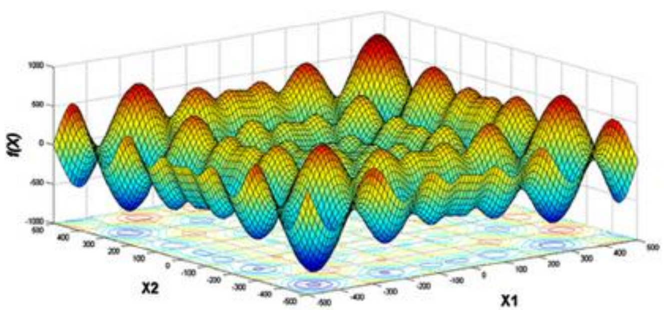
Example: Multi-objective optimization

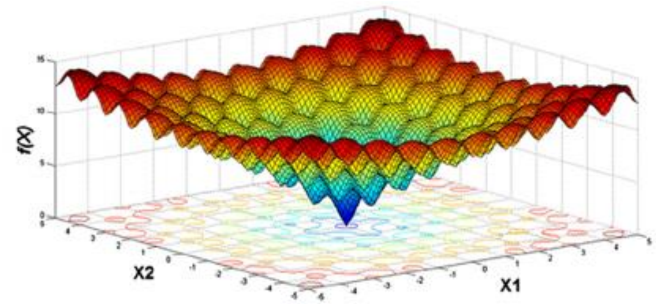
- Optimize multiple fitness functions simultaneously
- Conflicting objectives, trade-off solutions
- Iteratively improving the Pareto front (non-dominated solutions)

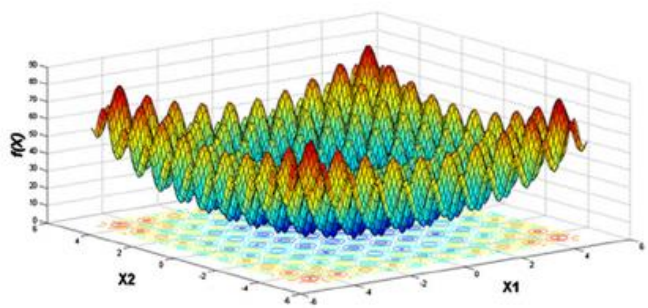


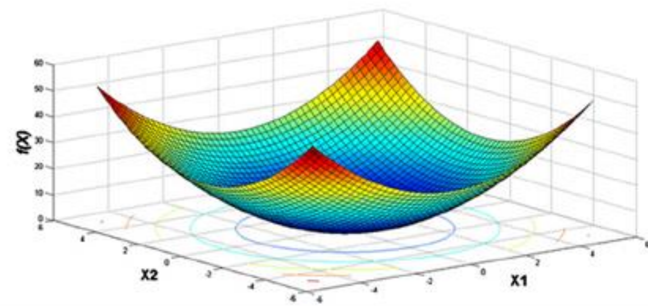
Case study: GA for Large Scale Global Optimization

- IEEE CEC Competition on Large Scale Global Optimization
- Objective: find the minimum of an unknown function in 1,000D
- Individual representation using numeric arrays size 1,000D









$$f(\bar{x}) = \sum_{i=1}^{D} x_i^2$$

Case study: GA for Large Scale Global Optimization

- Initialization of the population with P individuals using random continuous values within a range, e.g. [-100,100]
- Evaluation of the fitness of the initial solutions

Population	Gene 1	Gene 2	Gene 3	Gene 4	…	Gene D	Fitness
Individual 1	-72.62	87.66	-45.40	-19.83	37.31	94.96	9875
Individual 2	42.72	55.68	-9.98	-7.59	24.34	-3.41	5711
…	-48.31	-57.97	72.99	10.26	-51.24	-36.80	7653
Individual P	-85.35	-12.52	88.43	19.54	44.16	53.11	8952

Case study: GA for Large Scale Global Optimization

- Crossover (one point or multiple points)

Parent 1	-72.62	87.66	-45.40	-19.83	37.31	94.96
Parent 2	42.72	55.68	-9.98	-7.59	24.34	-3.41

Offspring 1	-72.62	87.66	-45.40	-7.59	24.34	-3.41
Offspring 2	42.72	55.68	-9.98	-19.83	37.31	94.96

- Applicable to any individual representation
- Recombine sections of the genotype
- Crossover probability should be large to converge fast

Case study: GA for Large Scale Global Optimization

- BLX-α for numeric representation

$$\beta_i^s = (min_i - I \cdot a) + \alpha \cdot |(max_i + I \cdot a) - (min_i - I \cdot a)|$$

Parent 1	-72.62	87.66	-45.40	-19.83	37.31	94.96
Parent 2	42.72	55.68	-9.98	-7.59	24.34	-3.41

Offspring	-34.26	65.12	-7.68	-17.42	36.48	23.65

- Provides faster convergence (exploitation of the search space)
- Generates new values, also out of the parents range

Case study: GA for Large Scale Global Optimization

- Mutation
- Generates new genetic material, providing diversity (exploration)
- Mutation probability should be small to not destroy evolution
- In early iterations the new values should have large stdev
- In final iterations the new values should have small stdev

Parent	-72.62	87.66	-45.40	-19.83	37.31	94.96
Offspring	-72.62	87.66	10.42	-19.83	37.31	94.96

Case study: GA for Large Scale Global Optimization

- Selection of the individuals to recombine via crossover should maximize their genetic diversity distance
- The best individuals among the parent population and the offspring are select to become the population for the next generation, keeping the population size constant
- Generational algorithms create a offspring population, steady-state algorithms create one new solution at a time (delete the worst)
- Let's see some working code!

Case study: GA for Large Scale Global Optimization

- Selection, Crossover, Mutation, Evaluation are iterative steps
- Parallel & distributed computing: speedup and/or throughput
- Population-level parallelization
  - Parallel evaluation of each individual
  - Parallel crossover of individuals
  - Parallel mutation of individuals
- Gene-level parallelization
  - Parallel crossover of genes within individuals
  - Parallel mutation of genes within individuals

Case study: GA for Large Scale Global Optimization

- Distributed computing works here?
- Option 1: Distribute computing for remote evaluation of solutions means overhead due to network transfer > performance improvement in this LSGO problem. FAIL.
- Option 2: Distribute individuals into multiple computers means a huge overhead every time running genetic operators. FAIL.
- Option 3: Distribute populations into multiple computers, run local algorithms and then migrate individuals. YEP!. This is known as multi-population genetic algorithms. I'll show you some code now.
- Option 4: Distribute genes into multiple computers. YEP!
(super secret, it works! trust me, we'll see later)

CMSC 691
High Performance Distributed Systems

Case study: Genetic Algorithms

Dr. Alberto Cano
Assistant Professor
Department of Computer Science
acano@vcu.edu