CSE/ECE 848
Introduction to Evolutionary Computation

Module 2, Lecture 6, Part 2d
More Principles of Evolutionary Computation – EC versus Other Methods

Erik Goodman
Professor, ECE, ME, and CSE
Michigan State University

EC Differences from Other Search Methods

- Maintain a set or "population" of solutions at each stage
- Typically use a *recombination* operator (operating on two or more solutions to generate an offspring)
- All we have learned up to time t is represented by time *t*'s POPULATION of solutions
- BIG advantage over single-point methods in ease of parallelization!

Contrast of EC with Other Search Methods

- "indirect" -- setting derivatives to 0
- "direct" -- hill climber
- enumerative
- random
- simulated annealing
- Tabu
- Response Surface Modeling (RSM) -- fits approx. surface to set of points, searches on that cheap-to-evaluate surface, avoids full evaluations until finds local optima on response surface; repeats

BEWARE of Claims about ANY Algorithm's Asymptotic Behavior – "Eventually" is a LONG Time

- LOTS of methods can guarantee to find the best solution, probability 1, eventually…
  - Enumeration
  - Random search (even better without resampling)
  - SA (properly configured)
  - Any GA that avoids "absorbing states" in a Markov chain
  - Any EA that continues to add new random individuals
- The POINT: you can't afford to wait that long, if the problem is anything interesting, so this claim doesn't do much for you!!!

When Might a GA Be Useful?

- Highly multimodal functions
- Discrete or discontinuous functions
- High-dimensionality functions, including many combinatorial ones
- Non-additive dependencies among parameters -- "epistasis" makes it hard for other approaches
- Often used for approximating solutions to NP-hard combinatorial problems
- DON'T USE if a hill-climber, etc., will work well

The Limits to Search

- No search method is best for all problems – per the No Free Lunch Theorem
- BUT fortunately, most real-world problems have some simple relationships among their variables that a suitable algorithm can discover and exploit! (See Kolmogorov complexity)
- Needle-in-a-haystack is just *hard*, in fact
- Don't let anyone tell you a GA (or THEIR favorite method) is best for all problems!!!
- Efficient search must be able to EXPLOIT correlations/relationships in the search space, or it is no better than random search or enumeration
- Must balance with EXPLORATION, so don't just find nearest local optimum

Examples of Successful Real-World GA/EC Application

- Antenna design
- Drug design
- Chemical classification
- Electronic circuits (Koza)
- Factory floor scheduling (Volvo, Deere, others)
- Turbine engine design (GE)
- Crashworthy car design (GM/Red Cedar)
- Protein folding prediction
- Network design
- Control systems design
- Production parameter choice
- Satellite constellation orbital design
- Stock/commodity analysis/trading
- VLSI partitioning/ placement/routing
- Cell phone factory tuning
- Data Mining
- Deep Learning Network Architecture