CSE/ECE 848
Introduction to Evolutionary Computation

Module 4, Lecture 18, Part 1b
Combinatorial Optimization

Erik D. Goodman, Executive Director
BEACON Center for the Study of Evolution in Action
Professor, ECE, ME, and CSE

Example Operators for Permutation-Based Representations:

Order Crossover:

- A = 9 8 4 | 5 6 7 | 1 3 2 10 (segment A and B)
- B = 8 7 1 | 2 3 10 | 9 5 4 6
- ==> B* = 8 H 1 | 2 3 10 | 9 H 4 H (in B, repl. 5 6 7 with H's)
- ==> B** = 2 3 10 | H H H | 9 4 8 1 (move middle segment from B to left, gather H's, append rest, with wrap-around)
- ==> B' = 2 3 10 | 5 6 7 | 9 4 8 1
- Similarly, A' = 5 6 7 | 2 3 10 | 1 9 8 4
- Order crossover preserves more information about RELATIVE ORDER than does PMX, but less about ABSOLUTE POSITION of each "city" (for TSP example).

Example Operators for Permutation-Based Representations:

Cycle Crossover:

- Cycle crossover forces the city in each position to come from that same position on one of the two parents:
- C = 9 8 2 1 7 4 5 10 6 3
- D = 1 2 3 4 5 6 7 8 9 10
- 9 - - - - - - - - -
- ==> 9 - - 1 - - - - - -
- ==> 9 - - 1 - 4 - - 6 - , which completes 1st cycle; then (depending on whose cycle crossover you choose), (i) start from first unassigned position in D and perform another cycle, or (ii) just fill in the rest of the numbers from chromosome D:
- (i) yields ==> 9 2 - 1 - 4 - 8 6 10
- ==> 9 2 3 1 - 4 - 8 6 10
- ==> C' = 9 2 3 1 7 4 5 8 6 10 D' is done similarly.
- (ii) yields ==> C' = 9 2 3 1 5 4 7 8 6 10. D' is done similarly.

Example Operators for Permutation-Based Representations:

Uniform Order-Based Crossover:

- (< Lawrence Davis, Handbook of Genetic Algorithms)
- Analogous to *uniform* crossover for ordinary parameter-based chromosomes. Effectively acts as if many one- or two-point crossovers were performed at once on a pair of chromosomes, combining parents' genes on a locus-by-locus basis, so is quite disruptive of longer schemata. (I don't like it much, but it works quite well for some problems.)
- A = 1 2 3 4 5 6 7 8
- B = 8 6 4 2 7 5 3 1
- Binary Template: 0 1 1 0 1 1 0 0 (random)
- ==> - 2 3 - 5 6 - -
- (then, reordering rest of A's nodes to the order THEY appear in B)
- ==> A' = 8 2 3 4 5 6 7 1
- (and similarly for B', ==> 8 4 5 2 6 7 3 1

Alternative Approach to Permutation Problems: Change the *Representation*, Not the *Operators*

Random Key GA (Bean, 1994) Is commonly used:

- Each locus represents one of the items to be sequenced (a city to visit, for example)
- Each locus is initialized as a random real in (0,1)
- The phenotype is determined by sorting the real numbers, remembering that their locus numbers represent the items to be sequenced: for example

  (1, 2, 3, 4, 5, 6)

  (.423, .623, .154, .334, .988, .744) translates to:

  3→4→1→2→6→5

Random Key GA, cont.

- Crossover can now be 1-pt, 2-pt, uniform, etc., WITHOUT fear of generating ANY illegal tours: that is, sorting will always produce a legal sequence
- Bean used a biased uniform crossover: 70% probability that each allele comes from first parent, 30% from second (this tends to make an offspring be more similar to one parent, which they found empirically to be effective)
- Original RKGA generated 1% new random individuals instead of locus-wise mutation, to maintain diversity
- Elitist selection kept fittest 20% unchanged, crossed others to make 79% of new population

Random Key GA, cont.

- Of course, like any other GA that keeps introducing new individuals, RKGA *converges asymptotically with probability 1,* but so do enumeration and random search in a combinatorial domain
- What ACTUALLY matters is the typical RATE of convergence
- RKGA does well APPROXIMATING optimal solutions, rather than in ”fine tuning” of solutions, so is useful for many real-world problems where fast, close-to-optimal solutions are valuable
- But how does it scale? Fast sort for each individual in a problem with K ”cities” is O(K logK), so for population size N, requires O(NK log K) per generation just for sorting, which can be a significant cost. (Many other techniques sort at most N individuals per generation, so O(N logN).

Other Types of Combinatorial Problems (OTHER than Sequencing)

- Knapsack-like subset selection problems
  - For N items, if N not huge, fixed length binary representation often used:
    - Locus i is 1 if item i is INCLUDED
    - Locus i is 0 otherwise
    - Or if multiple knapsacks, integer index of a knapsack for each item
    - Items may be differently weighted/sized/valued, with various types of capacity constraints and cost functions

Other Types of Combinatorial Problems (OTHER than Sequencing)(cont.)

- Knapsack-like subset selection problems
  - Otherwise, sometimes use variable length chromosome, where index of items to include is listed
    - Must then ”repair” any duplications that operators introduce—just need rules for what to do, and they may be problem-specific
- In general, *find* a representation and operators that make sense for YOUR problem, given what you know about handling of building blocks (schemata)