CSE/ECE 848 Introduction to Evolutionary Computation

Module 4, Lecture 18, Part 1a Combinatorial Optimization

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

Combinatorial Optimization

- Defined as any optimization conducted over a FINITE set of objects (so NOT over the reals, for example)
- Common combinatorial optimization problems:
  - Finding the optimum order of "visiting" N locations – the permutation problem, with N! solutions (TSP)
  - Finding the optimal subset of a set of objects (for example, knapsack problems)
  - Job shop/Flow shop scheduling problems
  - Delivery problems (which trucks get which objects to deliver where)
  - Warehouse and similar siting problems
  - Optimal layout/routing of IC chips
  - MANY other types of "logistics" problems

Today, essentially ALL optimization problems are *actually* combinatorial---

- Even when we're working on optimizing real-valued parameters for some physical system, WE use double-precision representations of those numbers, so we actually have only a finite number of possible solutions—but a LARGE finite number ☺
- We sometimes very fruitfully reduce that to a much smaller number of distinctly different solutions, so we can dramatically reduce the search space, and that works very well for many engineering problems, for example
- BUT, in such cases, we still know when variables are generally differentiable almost everywhere, etc., so we take advantage of continuum properties, even in a discrete domain

"Purer" Combinatorial Optimization

- Today we consider problems where each variable may have no underlying ordering of its alleles ("wood, steel, aluminum, rubber" or "truck, rail, plane", etc.)
- We'll start with a SEQUENCING problem (what order to perform certain operations)
- Evolutionary algorithms are the state of the art for solving many of these types of problems (scheduling classrooms at a university, for example)
- Many EC algorithms are buried in commercial software so the user never sees them, but they are there…

Traveling Salesman Problem (TSP)— The CLASSICAL Sequencing Problem

- Given n cities on a map, and the distances between all pairs of cities, calculate the shortest route from city A that visits each city exactly once and returns to city A.
- This is an NP-hard problem, so we'd better not be claiming to solve arbitrarily large instances in polynomial time! But we can generate excellent solutions to many large instances… (within some bounded fraction of the optimum… called *approximation*)
- However, while GA approaches have been applied successfully, the current TSP leader is a branch-and-cut approach (or has been, at least…)
- We are using TSP only as an example of how to evolve permutations as solutions to problems!

Direct Representation of a Permutation/Sequencing Problem

- Chromosome has EXACTLY ONE copy of each int in [0,N-1]
- Must find optimal ordering of those ints
- 1-pt, 2-pt, uniform crossover ALL useless:

```
    (3, 1, 7, 4, | 8, 5, 6, 2)
X   (4, 2, 7, 5, | 3, 8, 1, 6)
    (3, 1, 7, 4, | 3, 8, 1, 6)   ← has 2 3's, 2 1's, no 5, no 2, not a
                                   valid "tour" of 8 cities
```

- What about single-point mutation?
  (That always fails!)

Direct Representation of a Permutation/ Optimal Ordering Problem

- So must change either the *representation* or the *operators*!
- Let's first explore changing the operators
- Example mutation operators that preserve valid permutations:
  - *swap* 2 loci
  - *scramble* K adjacent loci
  - *shuffle* K arbitrary loci
  - etc.
- Any of these can work—what is best depends on the problem, of course!

Crossover Operators for Permutation Problems (Using Direct Representations)

What properties do we want:

- 1) Want each child to combine building blocks from both parents in a way that preserves high-order *schemata* in as meaningful a way as possible, and
- 2) Want all solutions generated to be feasible solutions
- BUT what’s a building block in this rep? Depends on the problem, crossover operator

Example Operators for Permutation-Based Representations, Using TSP Example:

PMX -- Partially Matched Crossover:

- 2 crossover points picked at random, intervening section specifies "cities" to interchange between parents:
- A = 9 8 4 | 5 6 7 | 1 3 2 10
- B = 8 7 1 | 2 9 10 | 3 5 4 6

- A' = 6 8 4 | 2 9 10 | 1 3 5 7
- B' = 8 10 1 | 5 6 7 | 3 2 4 9

Swap values of the selected loci between the two chromosomes

- (i.e., swap 5 with 2, 6 with 3, and 7 with 10 in both children.)
- Thus, some ordering information from each parent is preserved, and no infeasible solutions are generated because of "swaps."