

## Iterative Solving in GAMS

- Once you have the two models set up, you need to solve them repeatedly
- So put the solve statements in a loop:  
 Loop (i,  
     y1.fx=y1.l      y2.fx=y2.l ...  
     Solve UBP using NLP minimizing z;  
     x1.fx=x1.l ...  
     Solve LBP using MIP minimizing zp;  
     test for convergence  
     );

CPE 778, Lect. 28

97

---

---

---

---

---

---

---

## Stochastic Algorithms

- A totally different class of algorithms than the deterministic ones we've been using:
  - No relaxations
  - No upper bounds (maximization)
- Use random numbers to generate search directions
- Accept a new point with a given probability
- No guarantees of convergence

CPE 778, Lect. 29

98

---

---

---

---

---

---

---

## Stochastic Algorithms - Advantages

- Only use function evaluations
  - No requirements of continuity, differentiability, linearity, convexity
  - Do not even need a closed-form expression for constraints or objective function!
- Simple to implement
- Highly parallelizable
- Generate numerous feasible, near-optimal solutions quickly

CPE 778, Lect. 29

99

---

---

---

---

---

---

---

## Stochastic Algorithms - Examples

### • Simulated Annealing

- Based on simulations of materials (Metropolis, 1953)
- Accept a move to an inferior solution with a decreasing probability

### • Tabu Search

- Use a memory list to avoid repeating old solutions, escape local optima
- Apply local intensification to converge on optimum when a sufficiently good solution is found

CPE 778, Lect. 29

100

## Stochastic Algorithms - Examples

### • Genetic Algorithms

- Evolve solutions from a random population, based on Darwinian principles

### • Ant Colony Optimization

- Ants derive shortest path from nest to food with almost no communication
- Good for graph/network problems in which data structures can be designed to emulate natural situation

CPE 778, Lect. 29

101

## Example: Monte Carlo

- Monte Carlo methods include any algorithm which uses random numbers to derive a solution

### • Why might it be preferable to use random numbers in a constrained search?

- No need to compute gradients or other derivative-based information
- For NLP and MINLP problems, we only have local information even if we analyze each constraint, and this can lead to early termination in a local solution

CPE 778, Lect. 29

102

## Simulated Annealing

- Simulated annealing is a stochastic algorithm which is primarily used to solve large NLP and MINLP problems. The goal is to improve on the basic steepest descent idea of just moving in the direction of greatest decrease of our objective function
- To set up the algorithm, you need to define a move, that is, a way to change the variables of the system to get from one feasible solution to the next

CPE 778, Lect. 29

103

---

---

---

---

---

---

---

---

## Moves

- Let's define moves for a set of optimization problems:
  - Separation sequencing (assume we want each of a set of many components 99% pure)
    - Start with an order of separation (ABCD  $\rightarrow$  AB/CD  $\rightarrow$  A/B/C/D) and then switch one component in one separation
  - Molecular simulation
    - Move each atom a small, fixed spatial distance
  - Molecular design
    - Change one functional group, or divert one bond and move hydrogens

CPE 778, Lect. 29

104

---

---

---

---

---

---

---

---

## Beating Steepest Descent

- A naive algorithm to find a local minimum would just make a random move (but only a feasible one), and if that move decreases the objective function, accept it. This will usually find the nearest local minimum, or saddle point
- If we would like to look for the global optimum, we need to do something smarter
- In the annealing of metals, a mixture of ores is heated very hot and then cooled very slowly, in the hope that it will reach its minimum Gibbs Free Energy conformation. Let's apply this to an NLP problem.

CPE 778, Lect. 29

105

---

---

---

---

---

---

---

---

## The Metropolis Algorithm

- Based on that metals analogy, Metropolis (1953) proposed that moves which make the objective function lower be always accepted, but those which make the objective function worse be accepted with a probability of

$$P_{\text{accept}} = \exp\left(\frac{f(i) - f(j)}{k_B T}\right)$$

- The  $f$ 's are the objective function values at the current ( $i$ ) and proposed new ( $j$ ) states.  $k_B$  is Boltzmann's constant, but what is  $T$ ?

CPE 778, Lect. 35

106

## Temperature Deprogramming

- The  $T$  is called the control parameter, or commonly just the temperature. It allows the algorithm to slowly decrease that probability, with the hope that the algorithm has settled in the well which includes the global minimum, and then will not be able to escape that well and will reach that global minimum
- There are many algorithms to choose how to decrease  $T$ , and the best one to choose depends on the problem. These functions are called temperature deprogramming schedules.

CPE 778, Lect. 29

107

## Basic Algorithm

- Start with an initial solution  $\bar{x}$ , and an initial high temperature  $T$
- Repeat until temperature reaches some lower limit:
  - Repeat  $L$  times:
    - Make a random move to get the  $(i+1)$ th point
    - If that move decreases the objective function, accept it; otherwise, accept it with a probability  $\exp\left(\frac{f(x_i) - f(x_{i+1})}{T}\right)$
  - Reduce temperature according to the deprogramming schedule, for example,

$$T_{\text{new}} = 0.9T_{\text{old}}$$

CPE 778, Lect. 29

108

## Genetic Algorithms

- **Basic idea: simulate evolution to create improved solutions**
  1. Start with an initial population of solutions (initial guesses)
  2. Alter them in some way (stochastically)
  3. Copy each of them with a given probability based on their "fitness" (objective value)
  4. Repeat steps 2-3, storing the best solutions
- **Challenges**
  - Parameters: population size, mutation/crossover probabilities, reproduction/death limits and probabilities

CPE 778, Lect. 30

109

## Genetic Algorithms

- **Applications:**
  - MINLPs with many binary variables
  - Multi-objective NLPs, MINLPs
  - Black-box NLPs
  - Multi-stage scheduling with adaptive objectives and/or constraints
  - "Fuzzy" objectives or constraints

CPE 778, Lect. 30

110

## Genetic Algorithms - Example

- **Start with a binary program:**

$$\max \quad z = y_1 + 3y_2 + y_3$$

$$f(\bar{y}) \leq 0$$

$\bar{y}$  binary,  $f$  nonlinear, nonconvex

- **Now set up an initial population:**

$$\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \quad \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

- Evaluate the "fitness" of these solutions. If infeasible, fitness=0. If feasible, fitness=z

CPE 778, Lect. 30

111

### Genetic Algorithms: Crossover & Mutation

- Now alter the “genes” in some way:

- Mutation: flip a binary, randomly  $\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \Rightarrow \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}$

– Crossover: swap the top halves of two solutions (choose parents, break point randomly)

$$\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \Rightarrow \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} \& \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}$$

CPE 778, Lect. 30

112

---

---

---

---

---

---

---

---

### Genetic Algorithms: Recombination

- Crossover and mutation are called recombination operators
- Many other recombination operations are possible
- The key to a successful GA is to use recombination operators which:
  - Move quickly across the search space
  - Produce new but still feasible offspring
  - Generate steady improvement in the average fitness of the population

CPE 778, Lect. 30

113

---

---

---

---

---

---

---

---