Genetic Algorithms: Setting Parmeters and Incorporating Constraints

OUTLINE OF TOPICS:

1. Setting GA parameters
   - general guidelines for binary coded GA (some can be extended to real valued GA)
   - estimating computational requirements
2. Constraint Handling (two methods)
   - Penalty function only method
   - Method by Deb (2000) plus tournament selection

Note next lecture will cover theory topics

GA population paramters

- GA performance dependent on algorithm parameters
  - **population size**, *N*:
    - too small, not enough diversity
    - too large, too inefficient
    - N=10n is reasonable first guess (n is # decision variables)

allows you space to swap around variables.

GA Generation Number

– **maximum number of generations**, G_{max}:

– for expensive problems where computational time is limited, G_{max} is inversely related to population size

Max # objective fn evaluations=G_{max} *population size

(where # objective fn evaluations is user determined).

Total Computational Budget is the maximum amount of CPU time you can take to solve the problem.

time on CPU
⇒ # of cycles available

Total computational budget=
=Max # objective fn evaluations* CPU/eval

Where CPU/eval=CPU time per objective function evaluation.

- remember that the best solution can only be sequentially improved upon a maximum of G_{max} times

Summary

$$Pop = \frac{\#evals}{GMAX} \qquad \#evals = \frac{Total\ Comp}{CPU/Eval}$$

$$\Rightarrow Pop = \frac{Total\ Computational\ Budget}{GMAX * CPU/Eval}$$

$\Rightarrow$ If Pop is large, GMAX is small and vice versa

Trade-off Population Size and Number of Generations

- From the point of view of searching we would like both Pop and Gmax to be large.
- Hence there is a trade-off between Populations size (Pop) and the number of generations (Gmax) since Pop*Gmax= constant.
- If the ratio of Total Computational budget to CPU/eval is small, then you will be restricted and need to consider both.

Crossover Guidelines for GA

- **Crossover**
  - single-point crossover very common
  - multi-point crossover is alternative for long strings ⇒ increase diversity
  - when single point crossover is used, a reasonable default value for probability of crossover is 0.9-1.0
  - as the number of crossover points increases, the probability of crossover should decrease

Mutation Guidelines for GA

- **Mutation**
  - typically defined for each bit
  - common value for probability of mutation is 1/[population size] ⇒ more diversity → less need for mutations
  - mutation is a secondary operator to crossover → set mutation and crossover probabilities so that crossover is main driver of population evolution

What does a probability of mutation = 0.5 imply?

⇒ 50% chance to flip a bit!
→ could flip a lot of bits!

Methods for GA Selection

Selection Methods include:

- Biased Roulette Wheel selection
- Tournament Selection

Tournament selection has been shown to more efficient and less prone to premature convergence

Approach for Problem Analysis and Population Sizing for GAs

- Given a new problem, how long should you expect to run a GA to find good solutions?
- Finding good solutions with a GA requires adequate time for population evolution and subsequent convergence
- Paper by Reed et al. (2000) goes over general approach for determining GA requirements and feasibility
- Reed et al. (2000). Designing a competent simple genetic algorithm for search and optimization. *Water Resources Research*, v36 (12), 3757-3761

Reed et al. (2000): "Designing a Competent Simple Genetic Algorithm for Search and Optimization"

- "et al." includes:
  - Dr. David Goldberg (author of Genetic Algorithm book)
  - Dr. Barbara Minsker (past PhD student of Shoemaker)
- Paper derives approach based on GA schema theorems and various empirical studies
- This paper is based on the analysis of a simple binary GA that uses binary tournament selection and uniform crossover
- Similar to the GA coded for HW 3
- Results are not generally extendable to real value GAs

Picking probabilities of crossover and mutation

- Reed et al. (2000) suggest:

$$P_c = (s-1)/s \quad (4)$$

- $$P_m = 1/N \quad (5)$$
- Where P_c=prob of crossover and
- P_m=prob of mutation
- N= population size
- s = number of individuals competing in tournament selection.

Limitations Associated with Reed et al. analysis

- Reed et al also suggest that
  - Gmax=1.4 N (Gmax is number of generations)

    population size
- Their analysis is based on asymptotic analysis and the schema theorem.
- However, asymptotic analysis assumes you can make an arbitrarily large number of evaluations.
- It is not necessarily true that one will run the code as long as necessary to get an identical population so you should be aware of the limitations of the analysis.
- In addition in the Theory session of the course, we will discuss the Schema Theorem for GA, and the fact that it is not accepted as a mathematical rigorous proof.

Practical Guideline

- It is hard to decide how to set GA parameters.
- Usually this is done by doing some trial GA runs for a small number of generations and evaluations, with each run having different values for N, Gmax and Pc and Pm. to pick which algorithm parameter values tend to work for a fixed problem. (called “tuning parameters”)
- However, it helpful to get an idea of even which parameter values to try out.
- It is very reasonable to use the ideas in the Reed et al. paper to help you pick the range of values to use in the parameter tuning.

Practical Guideline for GA Parameter Tuning

- Assume the user has decided on MaxEval, which is the maximum number of evaluations you will make.
- The following is <u>one</u> way to select tuning ranges.
- Then (#generations)*(# in population) =

  =Gmax* N= MaxEval
- This then results in
- 1.4 N^2 =Max Evals or

$$N = \sqrt{(MaxEval / 1.4)} \qquad G\max = MaxEval / N$$

- You could then evaluate a range of values including the computed N and Gmax above