Generalized Parameter Estimation Algorithm with Sensitivity Analysis

Kevin Heath, Jacqueline Dresch, Robert Drewell

HARVEY MUDD COLLEGE

Background

Accurately modeling nonlinear biological models is often impossible to do in a reasonable time frame. The time needed to compute the correct values for each parameter can take days or weeks. Instead of checking the model at every possible parameter value, scientists often use computer programs to estimate the optimal values for each parameter. While there are numerous ways to do this, most gradient-based methods fail to find the global optimum of a nonlinear model. Consequently, such models require global optimization strategies that seeks the maximum or minimum of an objective function. The focus of my research is to design a global optimization program in C++ that can quickly and reliably estimate the correct values for any nonlinear biological model.

Introduction

There are four well-known global optimization methods:

Stochastic methods - These generate random parameter values and evaluate the result against the objective function

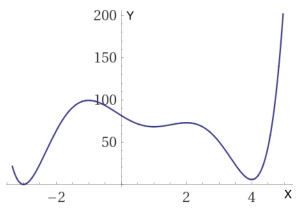
Clustering methods – Local optimization methods that start at different initial points

Evolutionary strategies – Population-based stochastic methods inspired by biological evolution

Simulated annealing – These mimic the cooling of metal to find the global optimum.

For my global optimization algorithm, I decided to combine the stochastic method, evolutionary strategy, and simulated annealing. Using OpenMP, I was able parallelize computationally intensive regions of code to drastically reduce runtime. Additionally, I applied simulated annealing to the natural selection aspect of the evolutionary algorithm to increase its chances of finding the global minimum of a nonlinear model.

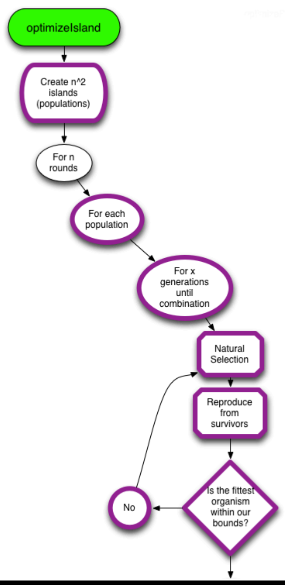
To test my stochastic evolutionary algorithm, I designed the following objective fitness function:



$$\frac{x^6}{6} - \frac{3x^5}{5} - \frac{11x^4}{4} + 9x^3 + 5x^2 - 24x + \frac{1629}{20}$$

Evolutionary Algorithm

The primary advantage of stochastic and evolutionary methods is their speed and parallelizability. While they may not be as efficient as other methods, they can be run very quickly and will eventually find the global minimum.



optimizeIsland
Create n^2 islands (populations)
For n rounds
For each population
For x generations until combination
Natural Selection
Reproduce from survivors
Is the fittest organism within our bounds?
No

Simulated Annealing

Simulated annealing works by initially jumping around and fully exploring parameter space before making fine adjustments. This works particularly well with nonlinear models, since a simulated annealing method is more likely than other methods to escape local optima and find the global optimum.

In my program, I use a Poisson distribution based on the sensitivity of each parameter. The sensitivity, or how much a parameter affects the model, is inversely related to how quickly that parameter should mutate. The mutation rate (μ) is calculated from the following Poisson distribution:

$$\frac{e^{-1+S_i}\,(1-S_i)^{f(t)}}{f(t)!}$$

It is then adjusted to the parameter space and chosen randomly from the interval (-μ, μ).

Results

The stochastic evolutionary algorithm I designed utilizes sensitivity analysis to mimic simulated annealing for its mutation rate. Tests show that my algorithm can be parallelized and converges to the global minimum more often than a normal evolutionary algorithm.

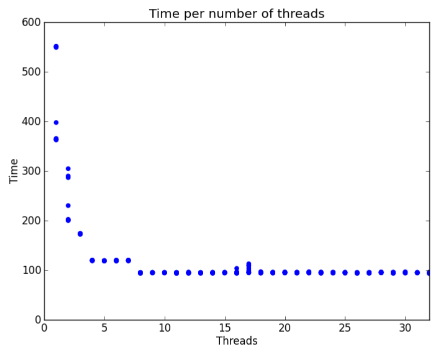


Figure 2. Time trials using multiple numbers of threads revealed that parallelization does significantly reduce run time. Using eight or more threads resulted in an almost 6-fold decrease in run time.

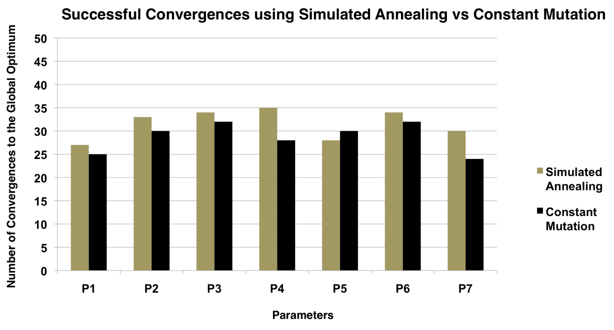


Figure 3. After running my algorithm using simulated annealing or a constant mutation rate 50 times each, the algorithm with simulated annealing clearly converges to the global minimum (-3) more often than the algorithm with a constant mutation rate.