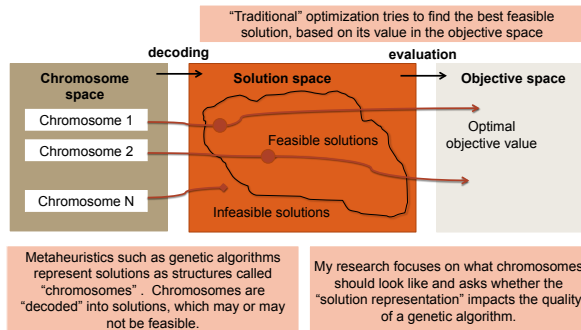


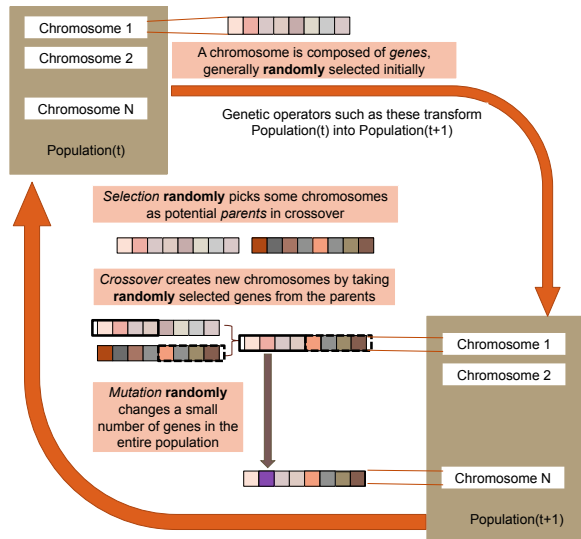
**Mary E. Kurz**  
**Department of Industrial Engineering**  
**Clemson University**

With thanks to at least a few but probably not all of the people who have helped in these adventures!  
 Matt Saltzman, Steve Stevenson, James Von Oehson, Eddie Duffy of Clemson; Doug Clayton of CycleComputing

Genetic algorithms are computationally intensive, high-throughput, inherently random metaheuristics used for optimization



Genetic algorithms are motivated by an analogy to "real" genetics



This genetic algorithm uses up to 600 million random numbers per run but less than 30 minutes running time. A study may consist of over 270,000 runs, with up to 150 trillion random numbers!

# Adventures in Random Numbers with Condor

## Abstract

Condor enables extremely large computational studies. When these studies involve random numbers and the desire for statistical analysis, as in research utilizing genetic algorithms, researchers can *easily* compromise their results through inappropriate use of random numbers.

## Where rudimentary knowledge of random numbers leads

The default C/C++ random number generator rand() returns an integer between 0 and 32767.

After about 32768 random number calls, the random numbers are repeating themselves – the stream cannot be independent! And, even if each of the 32768 numbers are used as starting points, we could only get 32768 independent streams of any length!

## Where more advanced knowledge of random numbers leads

The Mersenne Twister is a random number generator with an extremely large period:  $2^{19337}-1$ . This means we could run more than  $10^{5800}$  of these genetic algorithms in a row without using up all the numbers.

But, in Condor, we run many instances simultaneously. In a traditional computing environment, we can generate random numbers as if in a one lane road: run  $i+1$  picks up where run  $i$  left off. In a grid-enabled environment, such as that provided by Condor, we do not know when the "previous" run stopped generating random numbers because it is running at the same time, though on a different computer.

## Two potential but infeasible solutions

Assume at most MAX\_LEN random numbers from the Mersenne Twister are required for each experimental run. Provide each experiment with a unique identifier  $id$ .

1. For experiment  $id$ , generate but discard  $MAX\_LEN*(id-1)$  random numbers before beginning the real work of experiment  $id$ . In this research, this would take over 4000 hours for the last file with  $id$  270,000!!!
2. Generate  $MAX\_LEN*270,000$  random numbers and save those values in 270,000 separate files with 600,000 numbers each, identified by its  $id$ . When experiment  $id$  is run, use its random numbers. In this research, this would require over 750 TB of storage!!!

## The real solution?

Generate  $MAX\_LEN*270,000$  random numbers. After each set of MAX\_LEN numbers, save the state of the Mersenne Twister in a file identified by  $id$ . When experiment  $id$  is run, use the state from the appropriate file to reinitialize the Mersenne Twister. The state generation is in process on Clemson's server clemix with MAX\_LEN set to 1 billion. The state generation is expected to take about 22 days for 360,000  $id$  files. These state initialization files will be made available at [www.clemson.edu/~mkurz](http://www.clemson.edu/~mkurz) when complete.

## References

- Matsumoto, Makoto and Takuji Nishimura, "Mersenne Twister: A 623-Dimensionally Equidistributed Uniform Pseudo-Random Number Generator", *ACM Transactions on Modeling and Computer Simulation*, Vol. 8, No. 1, January 1998, Pages 3–30.
- Matsumoto, Makoto. "Mersenne Twister Home Page", <http://www.math.sci.hiroshima-u.ac.jp/~m-mat/MT/emt.html>.
- Saltzman, Matthew. "Pseudorandom Numbers and Condor", [ccit.clemson.edu/support/research/user\\_info/documentation/condor/condor\\_prng.php](http://ccit.clemson.edu/support/research/user_info/documentation/condor/condor_prng.php).