Applications of Distributed computing in Drug development

Mark Sale M.D.

Global Director, Research Modeling and Simulation.

Taxonomy of distributed computing:

- Parallel processing - more than one processor in a machine
  - Oak Ridge virtual parallel machine project. - 1970's
- Clusters - Beowulf cluster - 1994.
  - Dedicated nodes - better load balancing
  - Isolated network - don't need to worry (much) about network traffic
- Grid computing

Grid computing:

- “Grid is a type of parallel and distributed system that enables the sharing, selection, and aggregation of geographically distributed "autonomous" resources dynamically at runtime depending on their availability, capability, performance, cost, and users' quality-of-service requirements.” [1]

1 www.gridcomputing.com

Example:

- SETI@home (Search for Extraterrestrial Intelligence)
  - Data from radio telescope was collected, by segment of the sky
  - Sent, over internet, to individuals computers- often home computers, for analysis - looking for patterns suggesting ETI.

What does that mean?

- They aren't your computers (and so you don't have to pay for them)
- They aren't your computers (and so load-management is limited)
- They run over internet protocols (usually as web services) - I/O is limited
  - Computers typically "check in" every so often to see if there is anything to do (q 5 minutes in GSK implementation)

But...

- You can basically have as many computers as you want (GSK has 1000 so far, could have 50,000) - for free.
- You also need to break the problem up into fairly large pieces (for example, a NONMEM run).

Other applications:

Application	Status	Priority	Timelines
BLAST	On Hold	High	On Hold - Investigating Use
TurboSEQUEST	On Hold	Medium	On Hold - Awaiting Binaries
Protein Docking	In-Progress	Medium	28th Feb 2002
Confort	In-Progress	Medium	1st March 2003
*LigandFit	Completed	High	31st Dec 2002
*Diverse Solutions	Completed	Medium	1st Nov 2002
*Gold	Completed	High	20th Nov 2002
SCAMPI	Released	High	16th Sept 2002
Nonmem	Released	High	30th Sept 2002

All except NONMEM are vendor provided

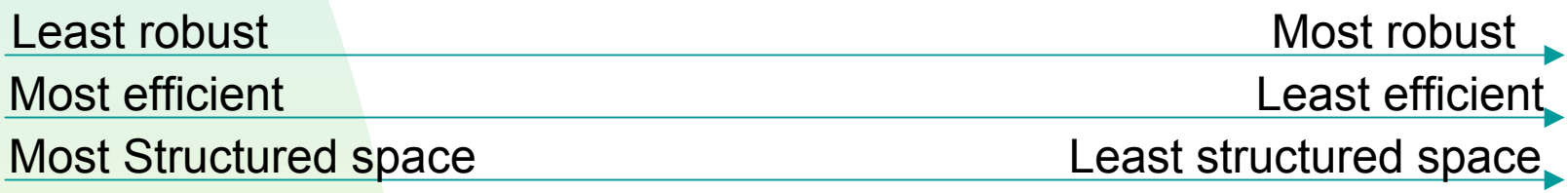
Grid computing, Pop PK models and Genetic algorithm

- Proposed new method* for model selection.
  - Current forward addition algorithm is less than perfect - there is no reason to believe it will ever give you the best model.
  - Other disciplines refer to this as binary tree search (or trial and error, or hunt and peck)
  - Roughly corresponds to a hill climbing algorithm.

*Patent pending

Model selection is a discrete (finite) space search problem

- Spectrum of discrete space search algorithms, from most robust/least efficient* to least robust, most efficient



Downhill/binary tree/forward addition/trial and error - monotonic

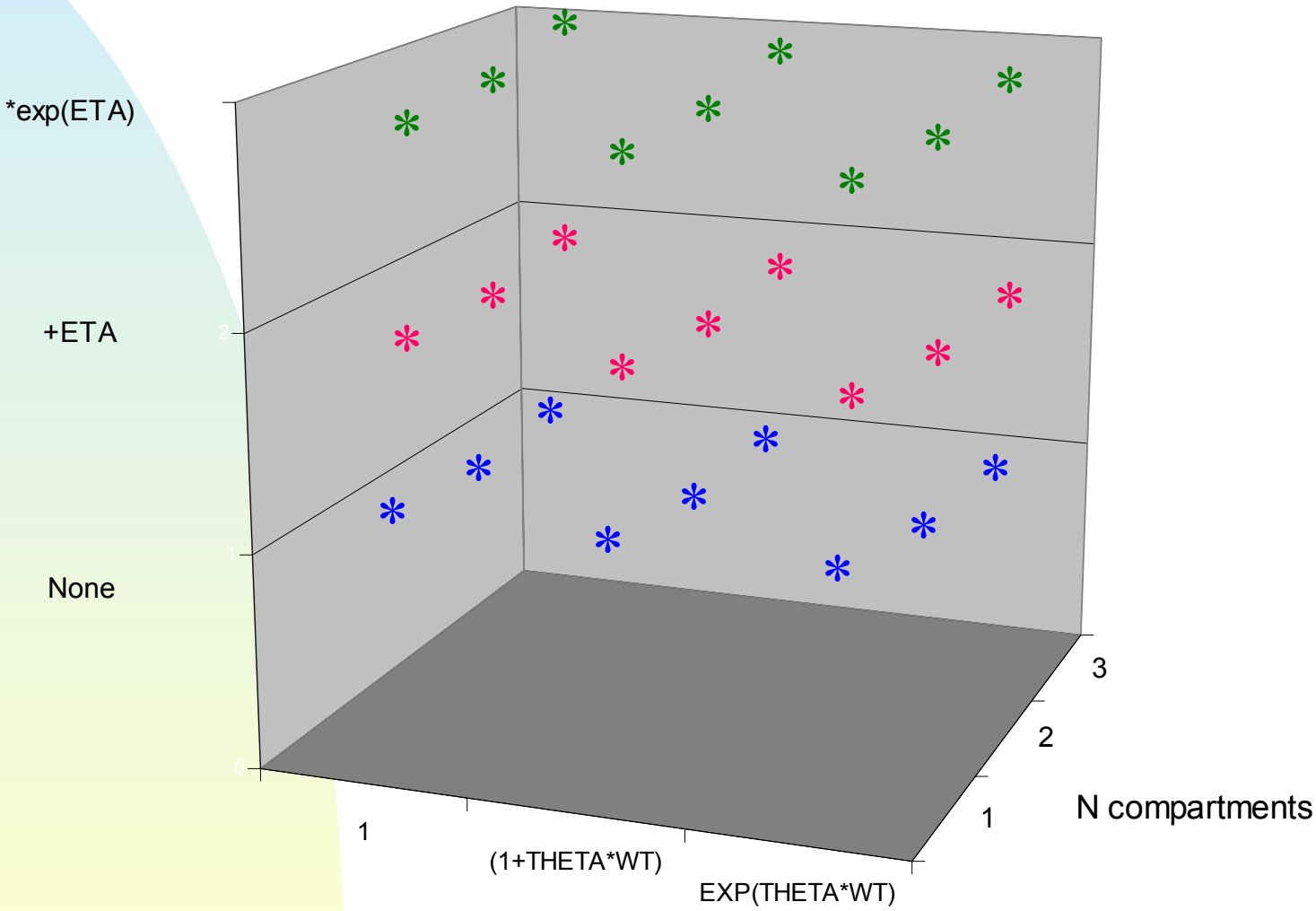
Integer programming based methods (ordered categorical)

Random search methods (GA, SA)

Exhaustive search (no structure)

*efficient defined as number of models until you reach the end, robust is defined as the probability of finding the best, or near best model using the algorithm

Candidate model space:



How to search space

- Full grid search

$$\prod_{1}^{Number\ of\ dimensions} (number\ of\ values)$$

- If we have 5 parameters (Ka,V,CL,K23,K32) and 4 covariates (Age, wt, gender, race), and 3 candidate models for each (none, additive, log), we get 3^20 candidate models in the space (3,486,784,401) (*omega options*number of compartments etc)

Genetic Algorithm

- Attempts to replicate the “optimization” process of evolution/survival of the fittest.
- Each set of candidate features is coded into a base 2 number. If there are 2 options in the feature set, the values are [0 1] if four

  [(0 0),(0 1),(1 0),(1 1)]
- These “genes” are strung together into a genome.

The genome - 3 genes, 2 models (individuals)

1|2 V=f(WT) V = f(age)...

cmt?

0 0, 1 0, 0 ;1 cmt, V=f(1+Θ•wt), V!f(age)

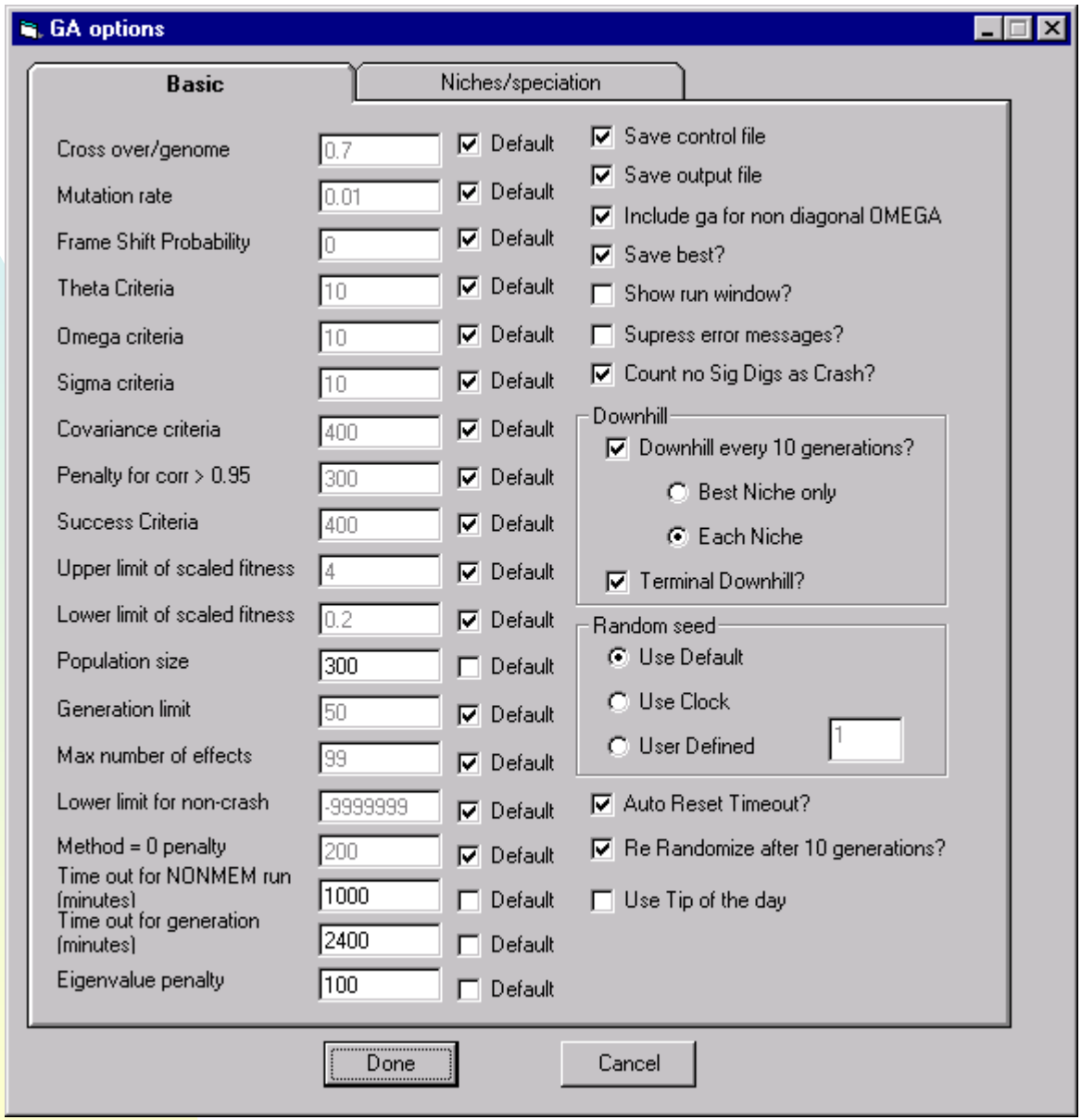
1 1, 0 0, 1 ;2 cmt, V=f(e^wt), V= f(1+Θ• age)

- Generate a random “population” of these individuals/models, assess the “fitness” of each.
- Select from the population, with replacement, proportional to fitness.
- Pair off those selected, cross over and mutate.
- New generation.

Fitness?

- What do we want the answer to be?
  - Minimize successfully
  - Covariance step
  - Low objective function
  - Parsimonious - few theta, etas, epsilons
  - ABS(Correlations) < 0.95
  - Ratio of largest/smallest Eigenvalues < 1000
  - FOCE?

Options for GA search



Does it work?

We compared simple GA to exhaustive search for a 13 dimensional problem (solution space size = 12,288). Simple GA found the same answer

We compared simple GA to exhaustive search for a 16 dimensional problem (solution space size = 98,304). Simple GA found a solution that was one bit different from the best solution.

GA is good at finding generally good regions

- Not very good at finding THE best solution within a small region. To find that one bit it must mutate the best model at that specific bit.

Hill Climbing

- Periodically, the algorithm will take the current best solution(s), and do a local hill climbing search - changing each bit (one at a time) and evaluating the solution. This is repeated until no further improvement is seen.

Result:

- Now found “best” solution in search space

Add re randomization:

- Same robustness and efficiency as exhaustive search - at least when sample size is < 1% of the search space.

Search parameters:

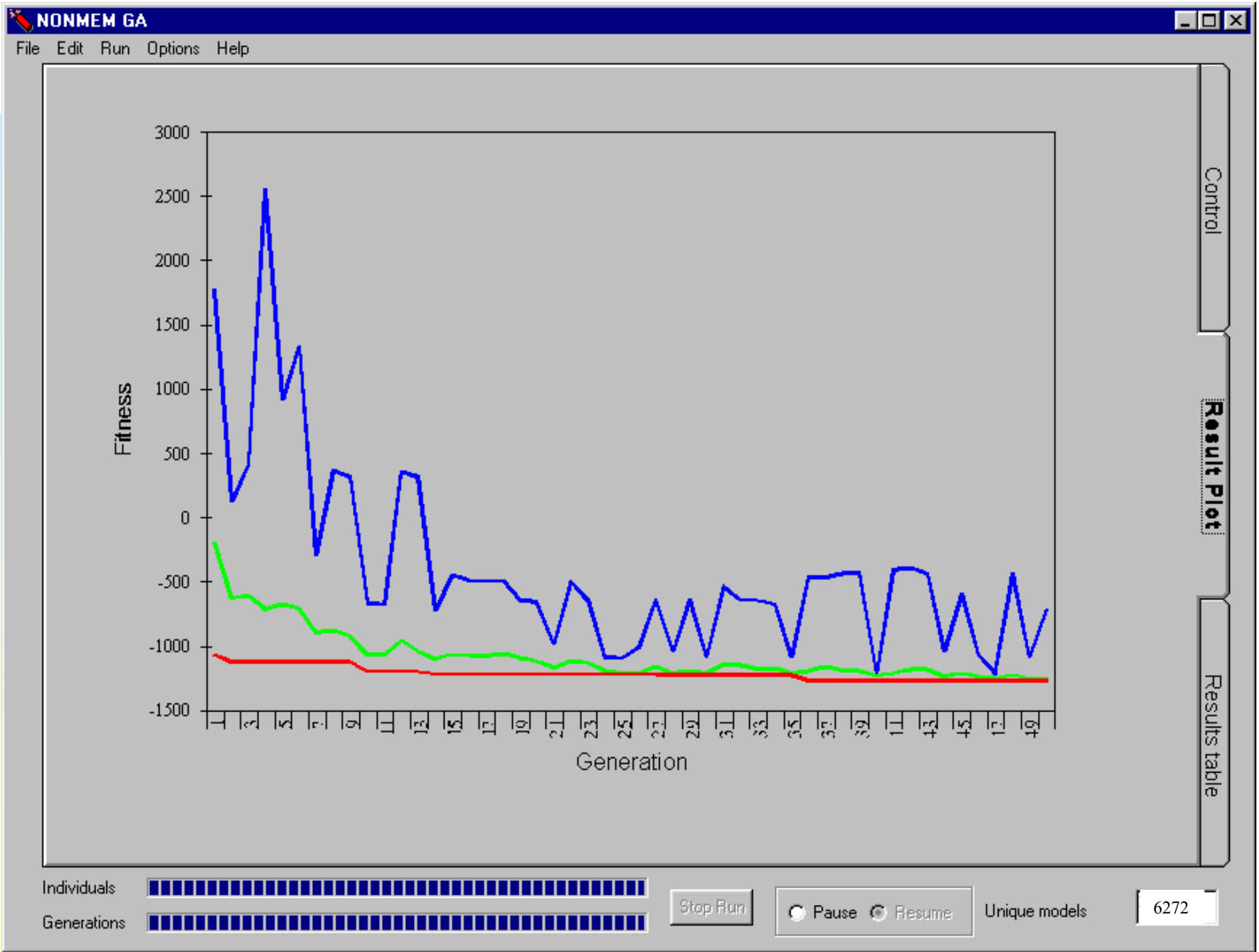
- Population size - number of “individuals” (model) used in the selection process. We’ve tried between 50 and 500. Increase in speed is minimal with population size > 300.
- Number of generations - typically used about 50
- Number of niches 4-8 for population size of 300.

Number of unique models in typical analysis:

- 300 models per generation
- 50 generations
- 50% rerun models
- 5 Hill climbing runs (once per 10 generations, downhill in each niche) - typically ~500 per run
- Total of > 8000 models.
- Not as efficient as traditional methods. However, with 300 computers running them, computational burden is reasonable.

Distributed solution

- Currently 1000 computers available
- Control files are written on local computer, sent, with all required NONMEM *.obj and *.lib files and all required compiler files (g77 compiler) to “agent” computer.
- Server is queried q 1 minute to see if it is done.
- Zipped results sent back to local computer.


NONMEM GA
File Edit Run Options Help
3000
2500
2000
1500
1000
500
0
-500
-1000
-1500
Fitness
Generation
Control
Result Plot
Results table
Individuals
Generations
Stop Run
Pause
Resume
Unique models
6272

Output:

```
Min fitness for generation 5 = -868.463
No change in min fitness for 1 generations
Starting generation 6 (n=139) 05/30/2003 12:57:54 PM
Job name = us0081421.ga69201540
Data set name = us0081421.data18191436
Number of processors used for generation 5 = 121
Total CPU time for generation  5 = 9.11 hours
Mean number of processor for generation  5 = 45.55
Maximum NONMEM execution time without timing out =
4.20 minutes, model #119
NONMEM timeout reset to 8.66 minutes
Run dist for generation 6 successful at 05/30/2003
1:10:03 PM
Run distributed time = 00:12
get result time = 00:02
```

Issues:

- Currently, computer must be on the intranet continuously.
- Difficult to recover from network failure.
- Inefficiency as current version waits until all models in generation are finished until new models are created.

- All to be solved in next version - web application

Conclusion:

- (Modified) Genetic algorithm seem to be a promising approach to model selection in NONMEM
- Distributed computing results in a 30 to 150 fold increase in speed for Genetic algorithm search, making it a practical application
- Other applications include MCMC methods, where distributed computing may be useful not only in model selection, but convergence criteria and model selection criteria.