A genetic algorithms approach to optimization parameter space of Geant-V prototype

Oksana Shadura
CERN, PH-SFT &
National Technical Univ. of Ukraine "Kyiv Polytechnic Institute"

Geant-V parameter space [1/2]

Component's parameters	**Concurrency properties**	**Queue management**	**Hardware properties**
GeantPropogator	Number of locks	Scheduler properties	CPU/GPU/ MIC/ARM
GeantScheduler	Cache locality/ memory management	Queue properties	
GeantTaskManager			
GeantWorkload Management			

Geant-V parameter space [2/2]

First approach what we should try to tune for Geant-V prototype:

<u>GeantPropagator</u>

•NThreads
•NEvents
•NTracks
•NperBaskets
•NminThreshold = NEvents x NTracks
•NMaxperBasket

<u>GeantScheduler</u>

•NVolumes = depends of Geometry
•NTracks
•NPriority

<u>GeantTaskManager</u>

•NThreads

<u>GeantWorkloadManager</u>

•NThreads
•NBaskets
•NIdleThreads
•MinFeeder

and etc.

Why we need optimisation?

- Geant-V is multicomponent software prototype with a complex behaviour of system as “black box”
- Geant-V is using all possible levels of parallelism, which makes complex to predict the quality (for example speed-up in Amdahl law) of simulation as a final result
- Correlated parameters (for example memory usage / RT) are not-easy to tune without multi-objective optimisation approach.

Genetic algorithm approach for optimization complex systems

- Genetic Algorithms (GAs) are population based search and optimisation methods that mimic the process of natural evolution. They fall in the category of stochastic global optimisation algorithms.

- Over the recent years GAs have been successfully applied to solve different MINLPs.

- GAs are easy to implement and are *black box optimisers* as they do not require any auxiliary information like continuity or differentiability of functions. They are robust and usually do not get trapped in a local optima.

Basic algorithm:

- Choose a population size.
- Choose the number of generations NG .
- Initialize the population.
- Repeat the following for NG generations:
    - Select a given number of pairs of individuals from the population probabilistically after assigning each structure a probability proportional to observed performance.
    - Copy the selected individual(s), then apply operators to them to produce new individual(s).
    - Select other individuals at random and replace them with the new individuals.
    - Observe and record the fittness of the new individuals.

Output the fittest individual as the answer.

Multi-objective GA Algorithms

The ultimate goal of a multi-objective optimization algorithm is to identify solutions in the Pareto optimal set. However, identifying the entire Pareto optimal set, for many multi objective problems, is practically impossible due to its size. In addition, for many problems, especially for combinatorial optimization problems, proof of solution optimality is computationally infeasible. Therefore, a practical approach to multi-objective optimization is to investigate a set of solutions (the best-known Pareto set) that represent the Pareto optimal set as much as possible.

Algorithms: Vector Evaluated Genetic Algorithms (or VEGA),Multi-objective Genetic Algorithm (MOGA), Niched Pareto Genetic Algorithm, Random Weighted Genetic Algorithm (RWGA), Nondominated Sorting Genetic Algorithm (NSGA), Strength Pareto Evolutionary Algorithm (SPEA), Pareto-Archived Evolution Strategy (PAES), Fast Non-dominated Sorting Genetic Algorithm (NSGA-II), Multi-objective Evolutionary Algorithm (MEA), Rank-Density Based Genetic Algorithm (RDGA).

Chromosome's encoding and fitness function for GA

Two main problem of GA approach are to select what&how will be coded as a representation of chromosome and find correct set of fitness functions

- Set of fitness functions will be selected from structure/ mathematical model of component interaction due to acceptable limitations of parameters in Geant-V (currently for tests I am using run.C as a basic fitness function with evaluation values of RT and memory usage)

- Representation of chromosome: could be binary encoding, permutation encoding, value encoding, tree encoding depends data. For simplicity had started with binary encoding

Optimization methods for GAs: clever & more faster [1/2]

- *Principal component analysis (PCA)*, where main purposes are the analysis of data to identify patterns and finding patterns to reduce the dimensions of the dataset with minimal loss of information.
- Could be used for tuning chromosomes classes.
- Outcome of the principal component analysis is to project a feature space (our dataset consisting of *n* x *d*-dimensional samples) onto a smaller subspace that represents our data "well". A possible application would be a pattern classification task, where we want to reduce the computational costs and the error of parameter estimation by reducing the number of dimensions of our feature space by extracting a subspace that describes our data "best".

Optimization methods for GAs: clever & more faster [2/2]

Suggestion for better performance:

- Instead of a simple weighted approach, add an intermediary intelligent system and optimise that system.
  - E.g. weights of a neural network or parameters of a fuzzy logic system.

What we can try to use?

Neuroevolution, or neuro-evolution, is a form of machine learning that uses evolutionary algorithms to train artificial neural networks. A main benefit is that neuroevolution can be applied more widely than supervised learning algorithms, which require a syllabus of correct input-output pairs.

For example: NeuroEvolution of Augmenting Topologies (NEAT) is a genetic algorithm for the generation of evolving artificial neural networks (a neuroevolution technique). It alters both the weighting parameters and structures of networks, attempting to find a balance between the fitness of evolved solutions and their diversity. It is based on applying three key techniques: tracking genes with history markers to allow crossover among topologies, applying speciation (the evolution of species) to preserve innovations, and developing topologies incrementally from simple initial structures (“complexifying").

Research plan

- Modelling of possible set of fitness functions and study of possible chromosome representation and appropriate genetic operators

- Usage of basic GA with different pairs of (input data/fitness func.) for observation of behaviour of system and correctness of pairs

- Usage of standard multi-objective GAs for more observation of more complex relationships between components

- Improvement of GA with PCA and neural weights approach

First approaches and first results (tests)

Test of simple single-objective GA for GeantV prototype (using DEAP library)
with a main goal of minimisation runtime of prototype depends of vector size of analysed particles.

- Chromosome representation - binary, size of population - 10, generations - 10
  - XXXXXXX 7 bit : first 3 are representing number of threads for prototype, last 4 are is encoded size of vector [8,1024]
  - Chromosome is generated randomly
- Fitness function is runtime (RT) of prototype
- Genetic operators:
  - Crossover in two points
  - Selection - Tournament strategy
  - Mutation - mutation with flipping bit with probability for individual 0.05
- Result:

```
..
  Min 16.1262111664
  Max 24.1407270432
  Avg 18.0737879038

-- End of (successful) evolution --
Best individual is [1, 0, 1, 1, 1, 0, 0], (16.126211166381836,)
```

1011 = 1+2+8 = 11 threads
100 = 64 bit vector