Genetic Algorithms Tutorial

Tamer Elsayed, Ph.D. Candidate
Advisor: Professor Michael Ortiz

Mechanical Engineering Department
Division of Engineering and Applied Science
California Institute of Technology

Overview

- Introduction
- Terminology
- Basic operators:
  - selection
  - crossover
  - mutation
- Why do Genetic Algorithms (GA's) work?
- Recommended parameters
- Comparison with other methods
  - Gradient methods
  - Simulated annealing
- Applications
- Conclusion

Introduction

- Development of Holland, computer scientist and psychologist at UM in 1975

- Historical focus and emphasis on function optimization applications, a trap!

- Genetic Algorithms as generalized adaptive systems

- Non deterministic methods:
  - can handle non-continuous functions
  - can handle non-analytically defined functions
  - can find a set of good solutions

Introduction

- To break the limit of deterministic methods (non-convex)
  - search space involve both continuous and discrete domains
  - objective function lacks regularity (expensive gradient)
  - objective function admits a huge number of local optima
  - parallel implementation

- In counterpart
  - computationally expensive
  - no guarantee to find global optimum

Introduction

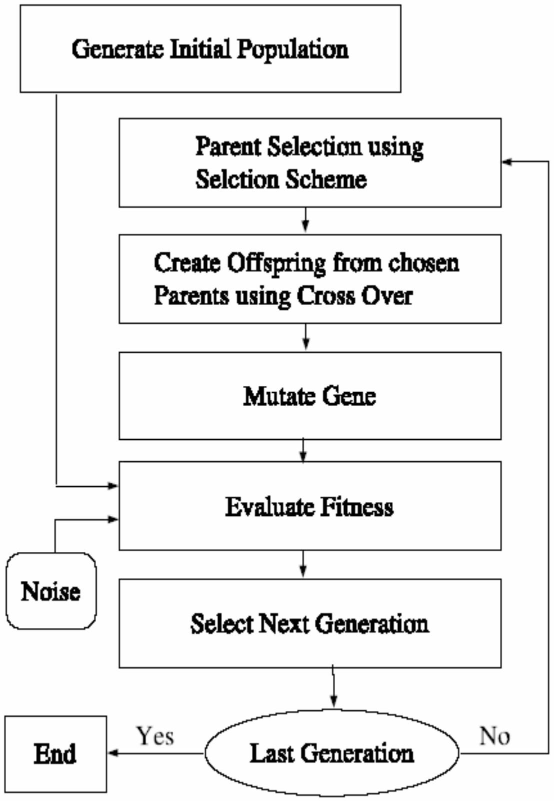
- Two basic processes from evolution:

  - inheritance

  - natural selection

- What must be developed:

  - encoding scheme: map solutions to chromosomes to which genetic operators can be applied



Generate Initial Population
Parent Selection using Selction Scheme
Create Offspring from chosen Parents using Cross Over
Mutate Gene
Evaluate Fitness
Noise
Select Next Generation
End
Yes
Last Generation
No

(Antonsson and Cagan,

Formal Engineering Design Synthesis, 2001)

Introduction

- What must be developed:
  - Initialization of population: randomly to sample the search space uniformly without bias (robustness)
  - evaluation function: all chromosomes are evaluated to see how fit they are as solutions
- Genetic operators:
  - must be designed based on the problem

(Antonsson and Cagan,
Formal Engineering Design Synthesis, 2001)

Terminology

- Encoding:
  representation, a linkage between the solutions in the phenotype space and the chromosomes is the genotype space (binary or real)

- Selection:
  Darwinian natural evolution, fitter individuals have a greater chance to reproduce offspring “survival of the fittest”
  - Parent selection to create offspring
  - Generational selection from both parents and offspring

Terminology

- Crossover:
  operator to be applied to two chromosomes to generate two offspring. Combines the schemata from two different solutions in various combinations (most important)

- Mutation:
  after creation of new individuals via crossover, mutation is applied usually with a low probability to introduce random changes into the population
  - replace gene values lost from the population or not initially present
  - evaluate more regions of the search space
  - avoid premature convergence
  - makes the entire search space reachable

Terminology

- Linkage:
  - majority of DNA is non-coding
  - modular genetic structure can develop due to the variability in non-coding segment length (or linkage)
  - certain blocks become more likely to stay together during recombination
  - integral part of GA

Terminology

- Linkage:
  - in GA: probability that two genes will be separated after recombination
  - adjacent genes have tighter linkage
  - crossover is more likely between non-adjacent genes
  - If linkages are allowed to adapt -> modularity
  - evolution of modularity allows exploitation of good building blocks while exploring other viable solutions

Basic operators: selection

- Fitness Proportionate Selection (FPS)
  - scaling
- Rank-Based Selection
- Tournament Selection
- Deterministic Selection

Basic operators: selection

(1) Fitness proportionate selection (FPS)

$$p_k = \frac{F_k}{\sum_{j=1}^{popsize} F_j}$$

for chromosome k with fitness value F_k and selection probability p_k

FPS methods assign selection probability to an individual based on its fitness (must scale)

- Fitness scaling is needed:

  - if range is too large …

  - if range is too small …

Basic operators: selection

- Scaling mechanisms:

(1) Linear:

$$F'_k = a \times F_k + b$$

- a and b are normally selected such that an average chromosome receives one offspring copy on average, and the best receives the specified number of copies (usually two)

- best chromosome gets a fixed number of expected offspring! prevents it from reproducing too many offspring

Basic operators: selection

- Scaling mechanisms:

(2) Sigma Truncation (Goldberg, 1989):

$$F_k' = F_k - (F - c \times \sigma)$$

- c is the sigma scaling factor, F is the average raw fitness value
- negative scaled fitness values are set to zero
- any individual worse than c standard deviations is not selected
- value of c in the literature is between 1 and 5

Basic operators: selection

- Scaling mechanisms (continued)

(3) Boltzmann Selection (Michalewicz, 1994):

$$F'_k = e^{(F_k/T)}$$

- nonlinear scaling method for proportionate selection
- T is a user-defined control parameter.
- The selection pressure can be adjusted by assigning T high or low.

Basic operators: selection

- Rank-Based Selection:

  - selection probability based on rank instead of its raw fitness.

  - Two methods to map rank into reproductive fitness:

  (1) linear ranking

  (2) exponential ranking

- (1) Linear ranking:

$$F'_k = q - (k-1) \times \frac{q - q_0}{popsize - 1}$$

  - *q* is the reproductive fitness for the best chromosome, and q_0 is the reproductive fitness for the worst chromosome

  - setting q_0 to zero ! maximum selective pressure

Basic operators: selection

- (2) Exponential ranking (Michalewicz, 1994):

$$F_k' = q(1-q)^{k-1}$$

  - larger value for q implies larger selective pressure

  - Hancock proposed the following exponential ranking method:

$$F_k' = q^{k-1}$$

  - where q is typically ~0.99. The best chromosome has a fitness of 1, and the last one receives $q^{(popsize-1)}$

Basic operators: selection

- Tournament selection:
  - Two kinds: binary and standard

  (1) Binary:
  - two individuals taken at random and the better is selected from the two.
  - if done without replacement, the two individuals are set aside for the next selection operation
  - the original population size is restored after the new population is half filled
  - the best individual will be selected twice and the worst won’t be selected
  - the number of copies selected of any individual cannot be predicted (either zero, one, or two)

Basic operators: selection

- Tournament selection:

  (2) Standard Tournament Selection:
  - random uniform sample of a certain size $q > 1$ is taken from the population
  - select the best of these q individuals to survive the next generation
  - very popular due to easy implementation, computational efficiency, and fine-tuning capability by increasing or decreasing q

Basic operators: selection

- Deterministic Selection:

  (1) Generational replacement:

  - replaces the entire parent generation with their offspring

  (2) Elitist selection:

  - problem with FPS: no guarantee for best to be passed
  - elitism: elitist selection guarantees asymptotic convergence

  (3) The (μ , λ) evolution strategy:

  - μ parents create $\lambda > \mu$ offspring by means of recombination and mutation
  - the best offspring individuals are deterministically selected to replace the parents
  - this method allows for the best member of generation t+1 to perform worse than the best individual at generation t, is that good?

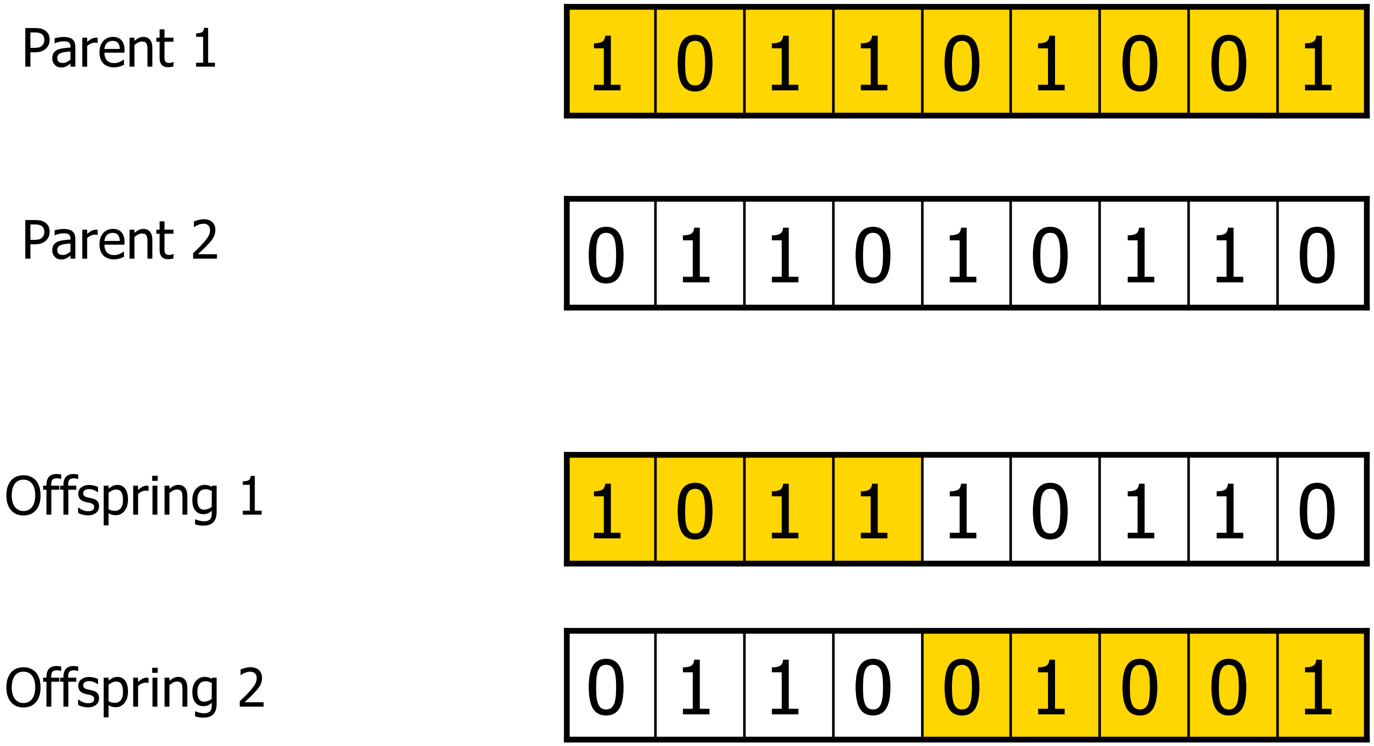
Basic operators: crossover

- Binary crossover

- Real crossover:
  (1) Simple crossover
  (2) Random crossover
  (3) Arithmetic crossover

Basic operators: crossover

- Binary crossover:



Parent 1	1	0	1	1	0	1	0	0	1
Parent 2	0	1	1	0	1	0	1	1	0
Offspring 1	1	0	1	1	1	0	1	1	0
Offspring 2	0	1	1	0	0	1	0	0	1

Basic operators: crossover

- Real crossover

  - each chromosome is encoded as a vector of real numbers with the same length as the solution vector

  - most belong to three categories:
  (1) simple crossover
  (2) random crossover
  (3) arithmetic crossover

Basic opertaors: crossover

(1) Simple crossover:

- analogous to binary (one or multi-point)
- only difference real numbers are used instead of binary bits

(2) Random crossover:

- Flat:

  produces an offspring by uniformly picking a value for each gene from the range formed by the values of two corresponding parents' genes
- Blend (BLX-α):

  also picks values from a range but not formed by the parents genes (extends the search)

Basic opertaors: crossover

(3) Arithmetic crossover:

- Let x_1 and x_2 be two parent chromosomes (real vectors):

$$x_1' = \lambda_1 x_1 + \lambda_2 x_2$$

$$x_2' = \lambda_1 x_2 + \lambda_2 x_1$$

linear: λ_1, λ_2 are real
affine: $\lambda_1 + \lambda_2 = 1$
convex: $\lambda_1 + \lambda_2 = 1$, $\lambda_1 > 0$, $\lambda_2 > 0$

Basic operators: mutation

- Crossover limitations
- Replace a gene (real number) with a randomly selected real number within a specified range
- Another kind is “boundary mutation” which replaces a gene with either the lower bound or the upper bound
- Mutation is implemented as a background operator and is usually applied with low probability
- If applied at high probability, the offspring will lose their resemblance to their parents and the ability of the algorithm to learn from the parents will be lost

Why do GA's work?

- Schemata: similarity templates or hyperplanes in l-dimensional bit space B^l
- A schema describes a subset of strings that have similarities at certain string positions
- To capture such similarities, schemata are defined to be strings of length l over the alphabet {0,1,*}
- For a given schemata H 2 $\{0,1,*\}^l$, a string a 2 B^l is called an instance of H iff whenever a position in H is specified (i.e. is either 0 or 1), this bit is identical to the corresponding bit in a
- The * serves as a wildcard and matches both 0 and 1

Why do GA's work?

- More formally, the set I(H) of all instances of a particular schema H = {h_1,…, h_l} is given by I: {0,1,*}^{l} ! 2^I
  according to:
  I(H) = {(a_1,..., a_l)| 8 i 2 {1,…,l}:h_i2{0,1})a_i = h_i}

- Example of a schema: H =(01*1*) having
  the set of all instances
  I(H) = {(01010),(01011),(01110),(01111)}

Why do GA's work?

- Two measures defined on schemata by Holland to measure its specifity:

  (1) The order:
  $O(H) = |\{I \mid h_i\ 2\ \{0,1\}\}|$
  example: $O(0{*}{*}1{*}) = 2$

  (2) Maximum distance (defining length of a schema):
  $\Delta(H) = max\{i \mid h_i 2\ \{0,1\}\} - \min\{i \mid h_i 2\{0,1\}\}$
  example: $\Delta(0{*}{*}1{*}) = 4 - 1 = 3$

Why do GA’s work?

- The measures are important in calculating the survival probabilities of that schema under mutation and crossover.

- The survival probability under mutation is:
  $\mathrm{P}\{m'_{\{pm\}}\ (\boldsymbol{a}_i(t))2\ \mathrm{I}(H(t))|\ \boldsymbol{a}_i(t))2\ \mathrm{I}(H(t))\}$
  which reduces to:
  $(1 - p_m)^{oH(t)}$

- The survival probability under one-point crossover is:
  $\mathrm{P}\{r'_{\{pc,1\}}\ (\boldsymbol{a}_i(t),\ \boldsymbol{a}_j(t))2\ \mathrm{I}(H(t))|\ \boldsymbol{a}_i(t))2\ \mathrm{I}(H(t))\} =$
  $\mathrm{P}\{r'_{\{pc,1\}}\ (\boldsymbol{a}_i(t),\ \boldsymbol{a}_j(t))2\ \mathrm{I}(H(t))|\ \boldsymbol{a}_j(t))2\ \mathrm{I}(H(t))\}$

Why do GA's work?

- Schema theorem:

$$N(H(t+1)) \geq N(H(t)).\frac{\Phi(H(t))}{\overline{\Phi}(t)}.[1-p_c.\frac{\Delta(H(t))}{l-1}.(1-\frac{N(H(t))}{\mu})].(1-p_m)^{\circ(H(t))}$$

- The basic statement is that short, low-order, above-average schemata (so-called binary blocks) receive exponentially increasing trials in the following generations
- GA's explore the search space by building blocks, subsequently combined into larger blocks by crossover.

Recommended Parameters

- Crossover rate:

Crossover rate should be high generally, about 80%-95%. (However some results show that for some problems crossover rate about 60% is the best.)

- Mutation rate:

On the other side, mutation rate should be very low. Best rates seems to be about 0.5%-1%

Recommended Parameters

- Population size:

  - Very big population size usually does not improve performance of GA (in the sense of speed of finding solution)
  - Good population size is about 20-30, however sometimes sizes 50-100 are reported as the best.
  - Some research also shows, that the best population size depends on the size of encoded string (chromosomes). It means that if you have chromosomes with 32 bits, the population should be higher than for chromosomes with 16 bits.

Recommended Parameters

- Selection:

  Basic roulette wheel selection can be used, but sometimes rank selection can be better.

- Encoding:

  Encoding depends on the problem and also on the size of instance of the problem

- Crossover and mutation type:

  Operators depend on the chosen encoding and on the problem

Comparison with other methods

- Gradient methods:

  - Good:
  for well behaved functions: use information about the gradient to guide the direction of the search

  - Problems:
  (1) fail if the derivative of the function can’t be computed
  (2) fail easily in problems with local optima

Comparison with other methods

- Simulated Annealing:

  - Process:

  (1) a random starting point is chosen

  (2) a random move is made

  (3) accepted if move takes to higher point

  (4) accepted with a probability p(t) if it takes to
  a lower point p(t) begins close to 1 and ! 0
  (analogy with the cooling of a solid)

  (5) t is reduced and another iteration begins

Comparison with other methods

- Simulated Annealing:

  - Problems:

    (1) one candidate solution at a time (no overall picture of the search space)

    (2) no information is saved from previous moves to guide new ones

Applications

- Nonlinear dynamical systems - predicting, data analysis
- Designing neural networks, both architecture and weights
- Robot trajectory
- Strategy planning
- Finding shape of protein molecules
- Sequence scheduling
- Functions for creating images
- Soon, laminates in composite shells!

Applications: Standard Solid Model

- Ortiz and Stainier introduced a time discretization of the following finite-deformation implementation of the standard solid model; the free energy takes the form:

$$RA\text{Y}F,T,F^{p}?=W^{e}\text{Y}FF^{p?1},T?+W^{p}\text{Y}F^{p},T?\ ^{-}\ W^{e}\text{Y}F^{e},T?+W^{p}\text{Y}F^{p},T?$$

- The corresponding equilibrium relations are:

$$P_{iJ}=R\frac{/A}{/F_{iJ}}=\frac{/W^{e}}{/F^{e}_{iB}}F^{p?1}_{JB}$$

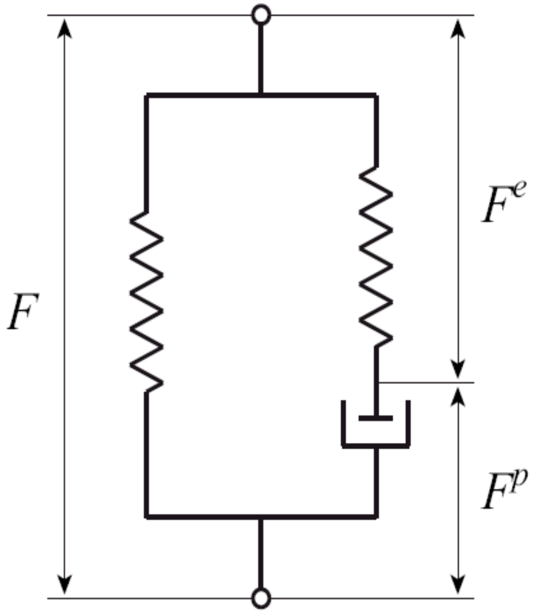
- The kinetic equations for F^{p} may be taken to have the potential form:

$$Y_{AJ}=\frac{/f^{\text{D}}_{6}}{/F^{p}}\text{Y}D^{p},T?$$

Model Parameters

- Six parameters needed for the model:

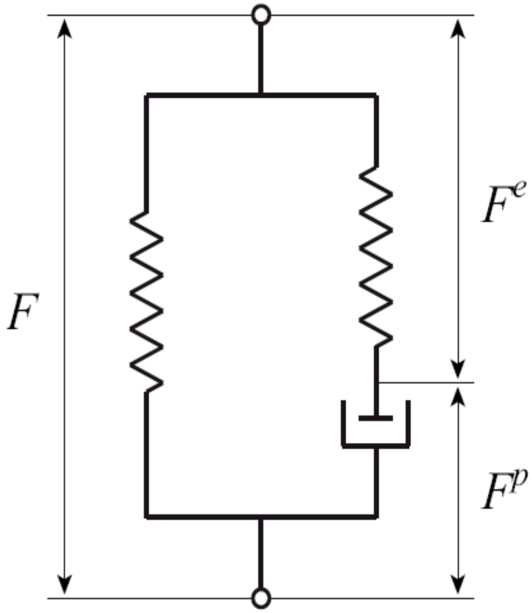
  (1) External spring bulk modulus
  (2) External spring shear modulus
  (3) Internal spring bulk modulus
  (4) Internal spring shear modulus
  (5) Dashpot bulk viscosity
  (6) Dashpot shear viscosity



Schematic Representation of finite-deformation viscoelastic standard solid model.

Model Parameters

- Model needs to be calibrated against experimental data
- The calibration process is a non-convex optimization problem
- Metric used is 1-norm (absolute value of maximum difference between experimental and computational Cauchy Stress values)
- The variables are the finite-deformation viscoelastic standard solid model.



Schematic Representation of finite-deformation viscoelastic standard solid model.

Population Formulation

- Each individual contains the six parameters and the metric value.
- One original individual is created with initial values and then mutated using mutation scheme to create 200 unique individuals.
- Parents from the population are selected randomly and a crossover scheme is used to create 100 more individuals.
- Another randomly chosen 100 individuals are mutated and the mutated individuals are added to the population to total 400 individuals.
- The metric is evaluated for the whole population using the finite-deformation viscoelastic standard solid model with the experimental strain values as input.
- Population is sorted based on the metric value and then enters the natural selection iteration process.

Natural Selection

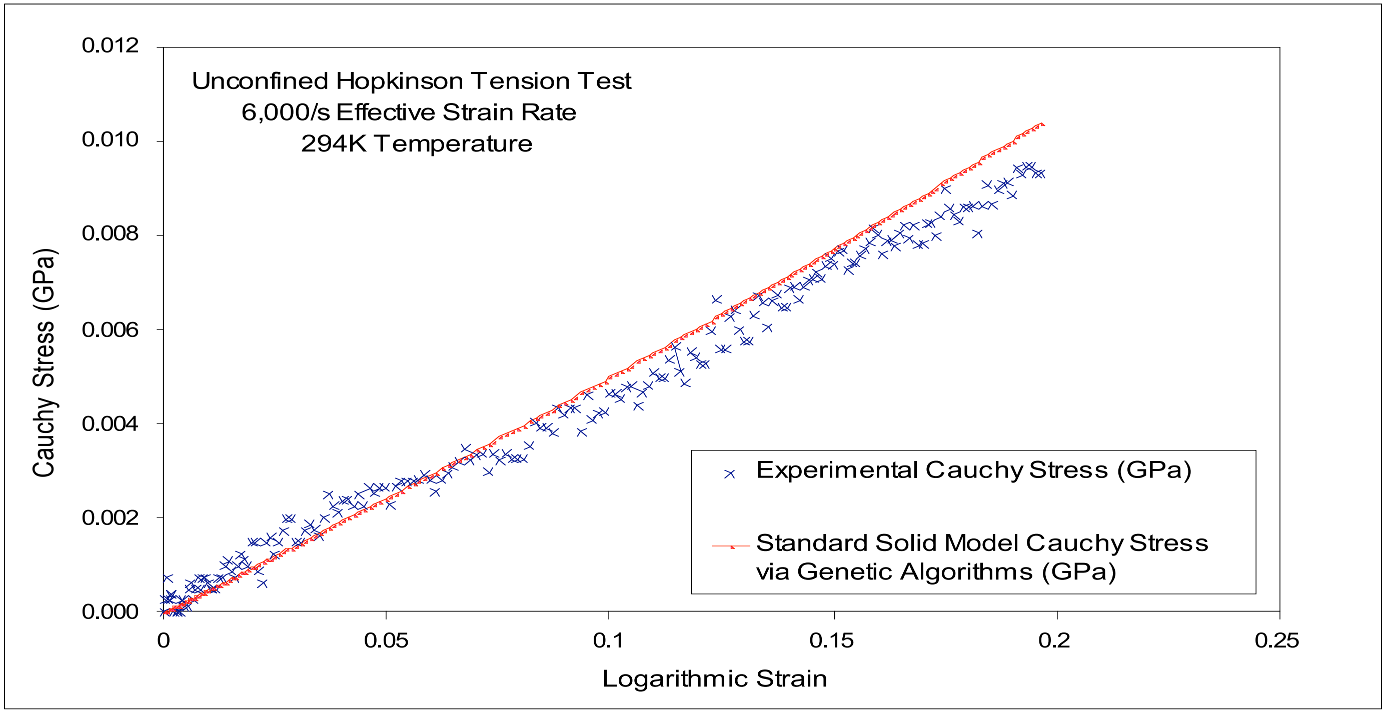
- Crossover is performed on 100 randomly chosen individuals and the resulting children are added to the old generation.
- Mutation is performed on 100 randomly chosen individuals and the resulting children are added to the old generation.
- The entire new population is now sorted using the metric evaluator.
- Bad individuals are eliminated keeping the total population size at 400.
- Repeat entire process.
- Best individual is always at first position in population.
- After suitable iterations Genetic Algorithms converges to local optima.

Unconfined Tension Tests

- Actual experimental strain values were used as input, not the effective strain rate
- The unconfined tension test data were used as input simultaneously
- Genetic Algorithms converged in 500 generations.
- Model parameter results via Genetic algorithms are:

  (1) External spring bulk modulus = 5.22E10 Pa
  (2) External spring shear modulus = 1.57E7 Pa
  (3) Internal spring bulk modulus = 6.79E8 Pa
  (4) Internal spring shear modulus = 4.92E8 Pa
  (5) Dashpot bulk viscosity = 6.86E8 Pa sec
  (6) Dashpot shear viscosity = 8.70E8 Pa sec

Unconfined Hopkinson Tension Test (6000/s)



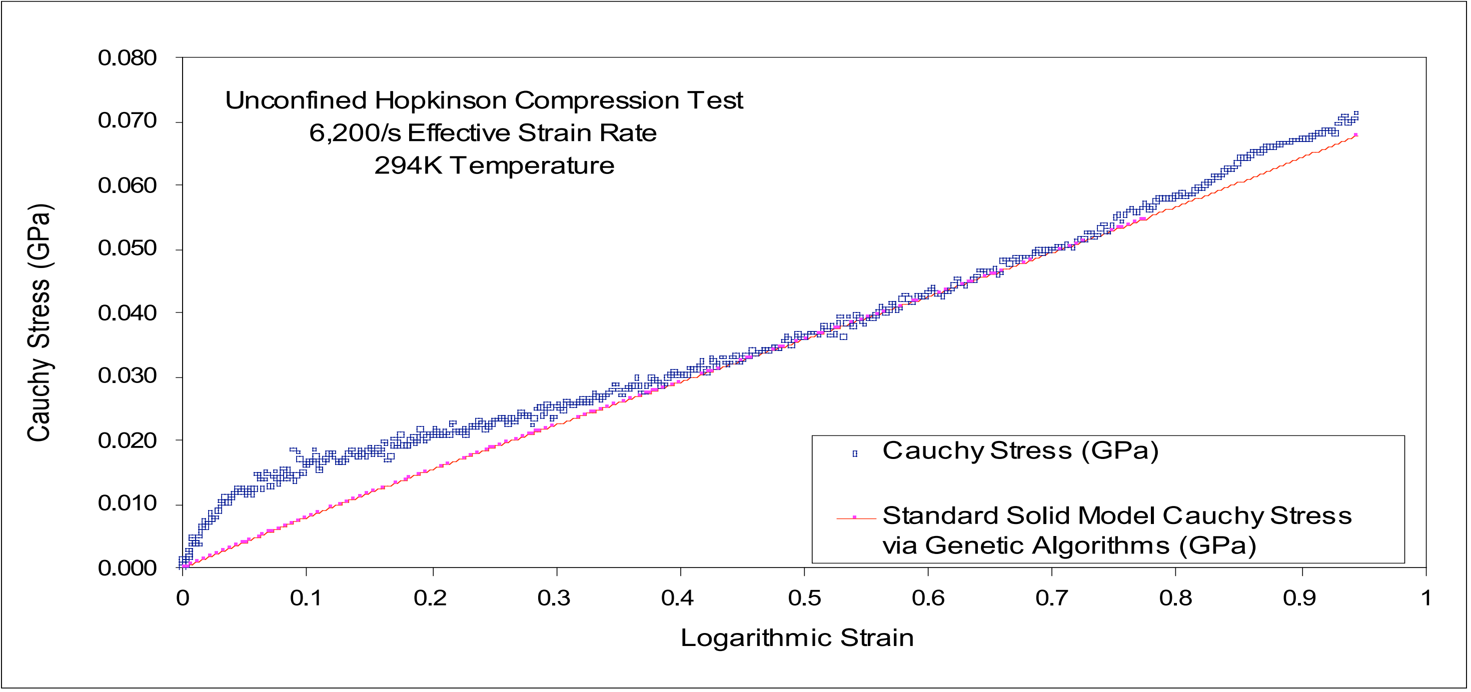
Hopkinson test data on polyurea obtained at UCSD's CEAM: Sia Nemat-Nasser, Jon Isaacs, Ali V. Amirkhizi, et al.; summarized on 07/20/05

Unconfined Compression Tests

- Actual experimental strain values were used as input, not the effective strain rate
- The unconfined compression test data were used as input simultaneously
- Genetic Algorithms converged in 45 generations starting from the previous parameters
- Model parameter results via Genetic algorithms are:

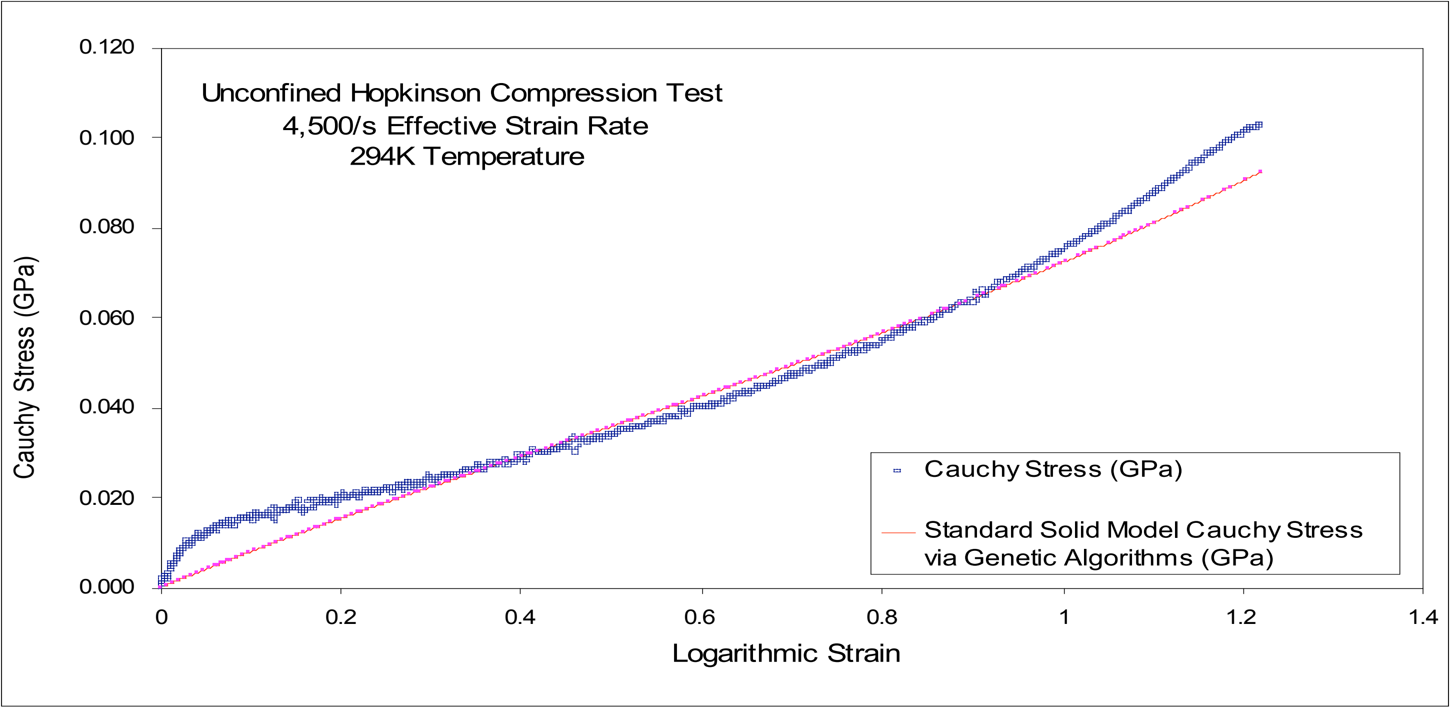
  (1) External spring bulk modulus = 4.30E10 Pa
  (2) External spring shear modulus = 2.82E7 Pa
  (3) Internal spring bulk modulus = 6.87E8 Pa
  (4) Internal spring shear modulus = 4.73E8 Pa
  (5) Dashpot bulk viscosity = 7.99E8 Pa sec
  (6) Dashpot shear viscosity = 9.01E8 Pa sec

Unconfined Hopkinson Compression (6,200/s)



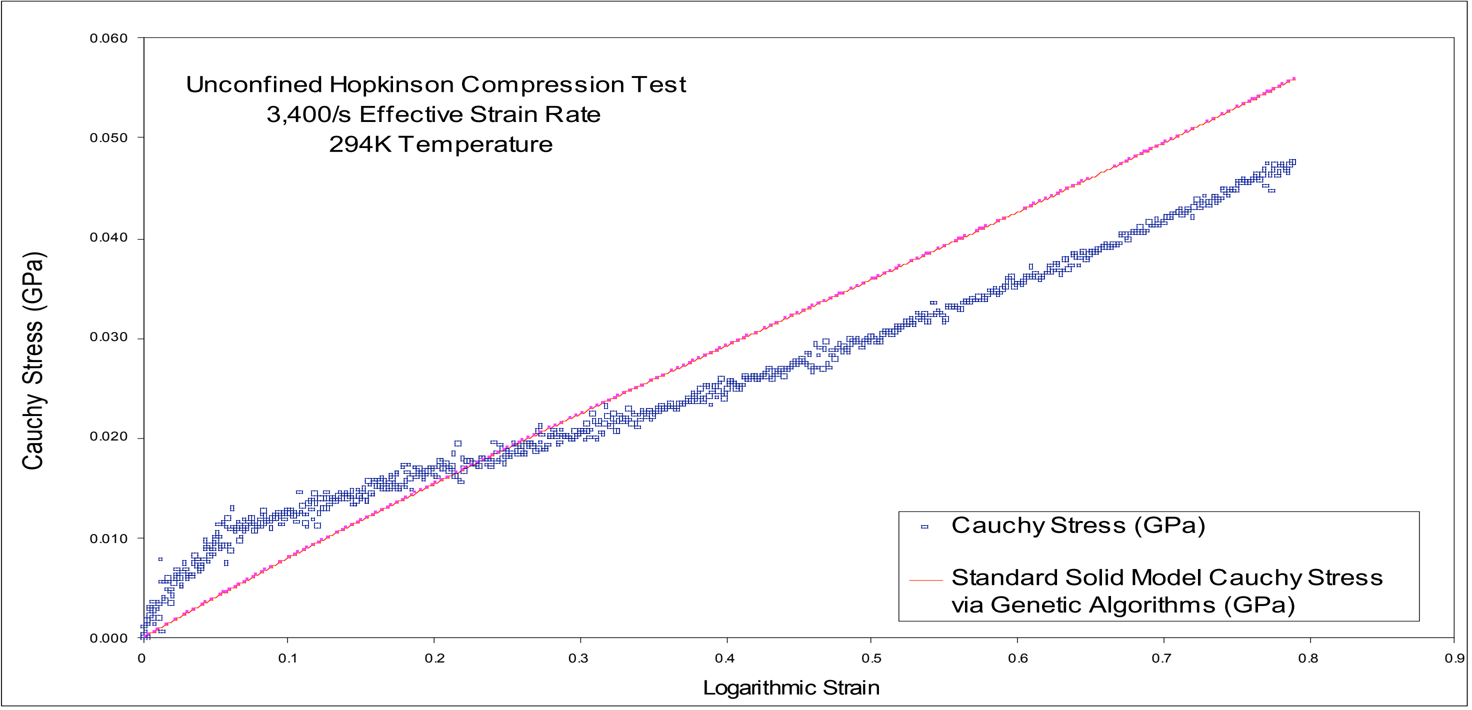
Hopkinson test data on polyurea obtained at UCSD's CEAM: Sia Nemat-Nasser, Jon Isaacs, Ali V. Amirkhizi, et al.; summarized on 07/20/05

Unconfined Hopkinson Compression (4,500/s)



Hopkinson test data on polyurea obtained at UCSD's CEAM: Sia Nemat-Nasser, Jon Isaacs, Ali V. Amirkhizi, et al.; summarized on 07/20/05

Unconfined Hopkinson Compression (3,400/s)



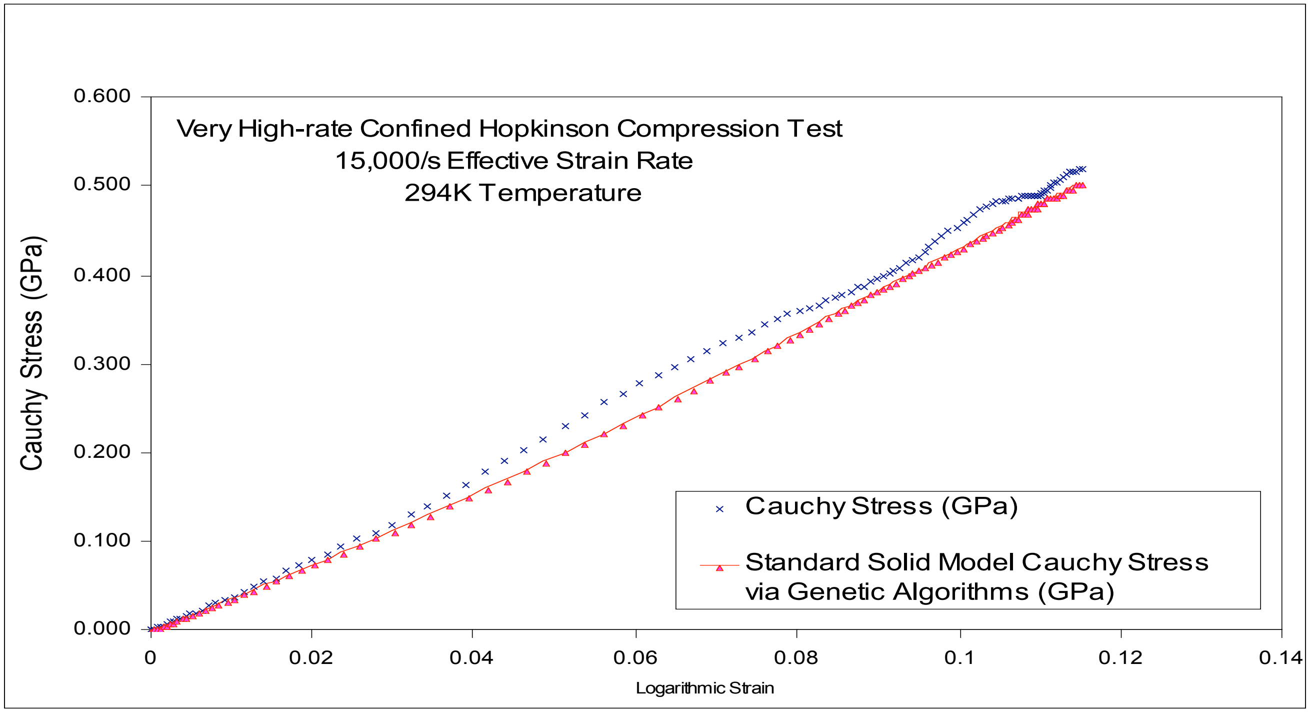
Hopkinson test data on polyurea obtained at UCSD's CEAM: Sia Nemat-Nasser, Jon Isaacs, Ali V. Amirkhizi, et al.; summarized on 07/20/05

Confined Compression Tests

- Actual experimental strain values were used as input, not the effective strain rate.
- The confined compression test data were used as input simultaneously.
- Genetic Algorithms converged in 65 generations starting from the unconfined compression tests parameters.
- Model parameter results via Genetic algorithms are:

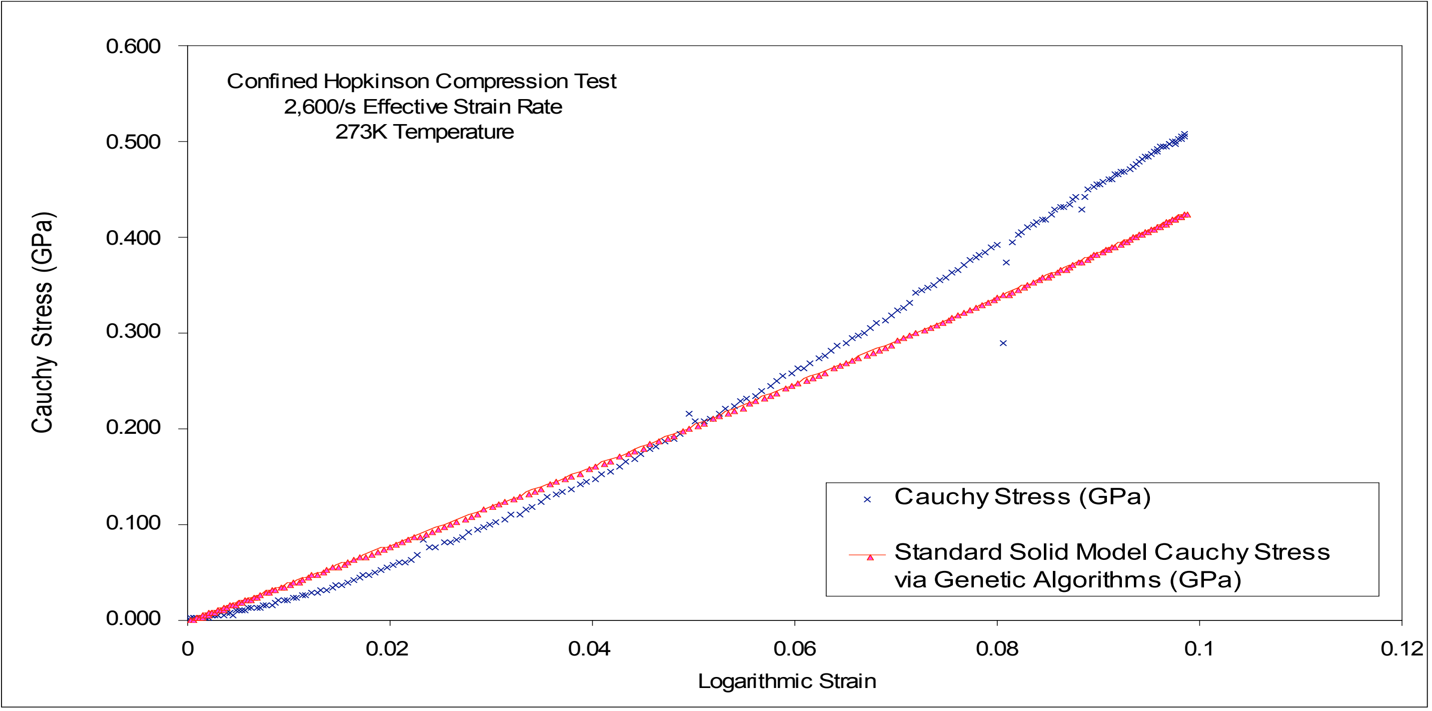
  (1) External spring bulk modulus = 3.91eE9 Pa
  (2) External spring shear modulus = 1.23E7 Pa
  (3) Internal spring bulk modulus = 6.57E8 Pa
  (4) Internal spring shear modulus = 3.77E8 Pa
  (5) Dashpot bulk viscosity = 9.11E8 Pa sec
  (6) Dashpot shear viscosity = 9.94E8 Pa sec

Confined Hopkinson Compression (15,000/s)



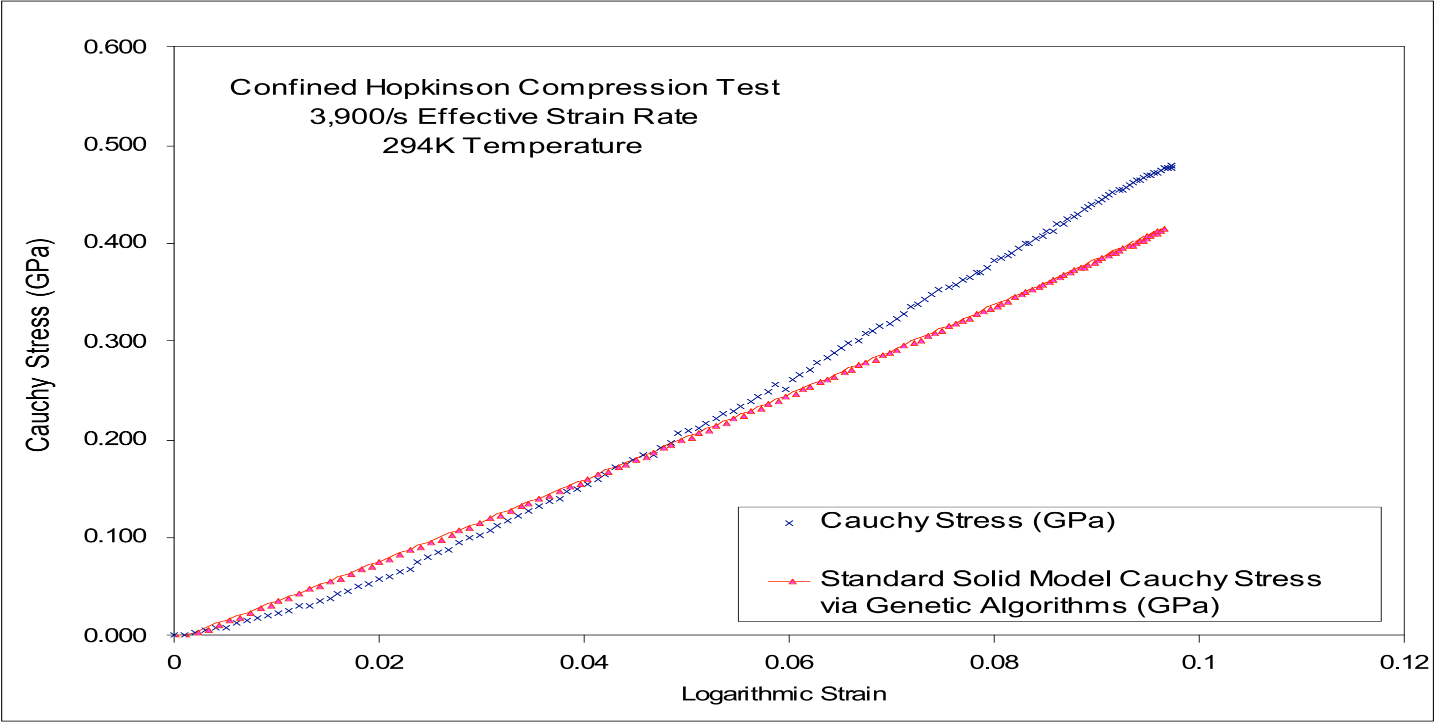
Hopkinson test data on polyurea obtained at UCSD's CEAM: Sia Nemat-Nasser, Jon Isaacs, Ali V. Amirkhizi, et al.; summarized on 07/20/05

Confined Hopkinson Compression (2,600/s)



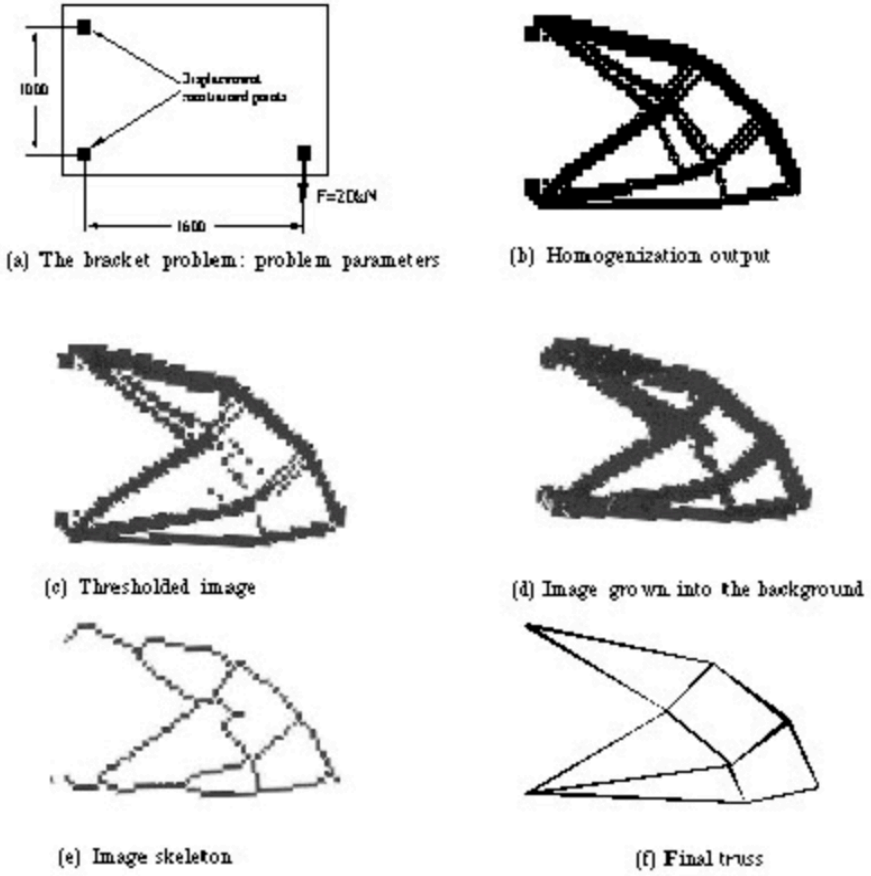
Hopkinson test data on polyurea obtained at UCSD's CEAM: Sia Nemat-Nasser, Jon Isaacs, Ali V. Amirkhizi, et al.; summarized on 07/20/05

Confined Hopkinson Compression (3,900/s)



Hopkinson test data on polyurea obtained at UCSD's CEAM: Sia Nemat-Nasser, Jon Isaacs, Ali V. Amirkhizi, et al.; summarized on 07/20/05

Structures



(a) The bracket problem: problem parameters

(b) Homogenization output

(c) Thresholded image

(d) Image grown into the background

(e) Image skeleton

(f) Final truss

Conclusion

- Genetic algorithms have been used for difficult problems (such as NP-hard problems), for machine learning and also for evolving simple programs. They have been also used for some art, for evolving pictures and music.

- The advantage of GA's is in their parallelism. GA is traveling in a search space using more individuals (and with genotype rather than phenotype) so that they are less likely to get stuck in a local extreme like the other methods.

Conclusion

- They are also easy to implement. Once you have the basic GA algorithm implemented, you have just to write a new chromosome (just one object) to solve another problem. However, for some problems, choosing and implementation of encoding and fitness function can be difficult.

- The disadvantage of GA's is in the computational time. GA's can be slower than other methods. But sice we can terminate the computation in any time, the longer run is acceptable (especially with faster and faster computers.)

- Website example

Crossover & Mutation

- Choose β as random double between 0 and 1 for crossover scheme
- Parameter = β x Parent1Parameter + $(1 - \beta)$ x Parent2Parameter
- Create crossover child using above parameters
- Choose β as random double between 0 and 1 for mutation scheme
- Mutated parameter is based on a suitable probability distribution
- Mutated parameter = old parameter + old parameter/15 x sign(β – 0.5) x exp(-1 x abs(β – 0.5))
- Create mutated child using above parameters