GADO: A Genetic Algorithm for Design Optimization

Khaled Rasheed
Computer Science Dept.
University of Georgia
khaled@cs.uga.edu
http://www.cs.uga.edu/~khaled
Presentation outline

- Introduction
- Architecture of GADO
- Comparison to other methods
- Ongoing work
  - Generating and using reduced models
  - Multi-objective optimization
- Conclusion
The engineering design optimization problem

**Objective**
- Given a tool that evaluates designs, find the best design according to some measure of merit and subject to some constraints
- Parametric design

**Example**
- Given an aircraft simulator
- Design a supersonic aircraft capable of taking 70 passengers from Chicago to Paris in 3 hours
- The aircraft should have the minimum takeoff mass (measure of merit)
- The wings should be strong enough to hold the weight of the aircraft in all stages (constraint)
Objective: Optimization Method Tailored to Design

- **Properties of complex design domains:**
  - Many unevaluable points
    - Simulators are designed for use by humans
  - Many infeasible points
  - Expensive evaluation functions
  - Discontinuity of design space
  - Many local optima
    - Physical or numerical
Domain 1: Supersonic aircraft design

- 12 parameters
- 37 inequality constraints
- 0.6% of the space is evaluable
Aircraft search space cross section

Exhaust Nozzle Design: Isosurface Visualization
Domain 2: Missile inlet design (NIDA)

- 8 parameters
- 20 inequality constraints
- 3% evaluable, 0.147% feasible
NIDA search space cross section

Feasible
Genetic Algorithm Based Design Optimization

- Maintains a population of potential designs (individuals)
- Better designs are generated using:
  - Crossover: 2 designs from the current population combine attributes
  - Mutation: 1 design changes attributes
- Fitness of a design is based on measure of merit and constraint violation (penalty)
Elements of a steady state genetic algorithm

- Representation
- Fitness function
- Initialization strategy
- Selection strategy
- Crossover operators
- Mutation operators
- Replacement strategy

Population

Newborn

C1

C2

Selection

Crossover

Replacement

Mutation
## GADO: Genetic Algorithm for Design Optimization

<table>
<thead>
<tr>
<th>Representation</th>
<th>Fitness</th>
<th>Initialization</th>
<th>Selection</th>
</tr>
</thead>
<tbody>
<tr>
<td>Floating point</td>
<td>Adaptive Penalty</td>
<td>Repeated Random</td>
<td>Rank-Based</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Crossover</th>
<th>Mutation</th>
<th>Replacement</th>
<th>Search-Control</th>
</tr>
</thead>
<tbody>
<tr>
<td>Point</td>
<td>Uniform</td>
<td>Crowding</td>
<td>Screening Module</td>
</tr>
<tr>
<td>Line</td>
<td>Non-Uniform</td>
<td></td>
<td>Diversity</td>
</tr>
<tr>
<td>Double Line</td>
<td>Greedy</td>
<td></td>
<td>Maintainance</td>
</tr>
<tr>
<td>Uniform</td>
<td>Shrinking-Window</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Guided</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
GADO: Genetic Algorithm for Design Optimization

• **Most Novel ideas:**
  • Guided crossover
  • Screening module
  • Diversity maintenance module
  • Adaptive penalty functions
Guided Crossover

**Method:**
- Select one point
- Find second point in "best" direction
- Pick a point along the line connecting them

**Motivation:**
- Add gradient-like functionality without expense of computing gradients
Screening Module

• **Method:**
  - Find $k$ nearest neighbors
  - Discard if all $k$ are below threshold
  - Threshold = Function of current population

• **Motivation:**
  - Decreases number of evaluations by avoiding unevaluable regions, as identified in past evaluations
  - Can eliminate >30% of evaluations
  - Negligible overhead
Diversity Maintenance Module

• **Method:**
  • At start compute inter-solution distances
  • If inter-solution distances are too small relative to this, reseed from earlier population elements
  • Reject points near past points

• **Motivation:**
  • Maintains diversity
  • Fewer evaluations
Adaptive Penalties

**Method:**
- Fitness = Measure of merit + Penalty
- Penalty = $c(t) \times \sum$ constraint violations
- $c(t)$ increases whenever the best element of the population does not have the least constraint violation
- $c(t)$ can also decrease to inject "slightly" infeasible points into the population

Optimum

Feasible

Infeasible
Comparison of methods: Conceptual Design of Aircraft

- **Random probes:**
  - No feasible points in 50,000 tries

- **Multistart CFSQP:**
  - Inferior on average
  - High variance in quality of solutions

- **Genocop III (GENetic algOrithm for Constrained OPtimization),**
- **ASA (Adaptive Simulated Annealing):**
  - Require feasible starting points
  - Inferior from "good" starting points
GADO vs. CFSQP in Aircraft design (domain 1)
GADO runs

Evaluations

Take off mass(ton)

'GADO_Aircraft_traces'
Multistart CFSQP runs
GADO vs. Genocop III and ASA in Aircraft design domain
Results in Missile Inlet Design (domain 2)
Case Study: Redesign of a two-dimensional supersonic inlet

- Original designs by ITAM (Russia), redesign by Michael Blaize (Aérospatiale, France)

- First inlet
  - ITAM design: Total pressure recovery = 0.134
  - GADO: Total pressure recovery = 0.194 (1.25 CPU hours)
  - CFSQP:
    - From GADO’s optimum: no improvement
    - From original (ITAM) design: Total pressure recovery = 0.160
    - Multistart: no better than the original design (1 CPU day)
GADO achieved

• Faster optimizations
• Better final designs
• Lower variance in final design quality
• Low sensitivity to internal parameters and setup
Ongoing research directions

- New application domains
- Using reduced models for speedup
- Multi-objective GADO
Generating and using reduced models for design optimization

- Reduced models and their sources
- Generation of reduced models
- Using reduced models through informed operators
- Future directions
Reduced models

- **Pre-existent:**
  - Simpler physical models
  - Coarse grids
- **Generated:**
  - Functional Approximations (Response Surfaces)
    - Least Squares
    - Neural Networks
    - Genetic Programming
Observation

- Previous methods do not take properties of design domains into consideration
  - Unevaluable points
  - Numerical problems: discontinuity, high non-linearity
- Some approaches make strong assumptions about reduced model accuracy
Generating reduced models by incremental approximate clustering

- Maintain previously encountered points divided into dynamic clusters
- Periodically introduce new clusters and refresh all clusters
- Periodically compute quadratic approximations
  - Separate approximations for measure of merit and constraints
  - Global approximation: all points
  - Cluster approximations: large enough clusters
Approximate evaluation of a new point

- If point’s cluster has approximations, use them, otherwise use global approximations
- Two phase approach:
  - Classify point using K nearest neighbors (feasible, infeasible, unevaluable)
  - Use classification and proper approximation functions to form fitness
Informed operators

- **Idea**: replace randomness with decisions informed by the reduced model
- **Examples**:
  - Informed initialization
  - Informed crossover (parents, method)
  - Informed mutation (type, amplitude)
Informed mutation

- Crossover done, followed by several random mutations
- Random mutations are evaluated using reduced model best becomes newborn
Utility of informed operators in aircraft design
Speedup with informed operators in aircraft design

![Graph showing the comparison between 'GADO_Aircraft_long' and 'GADO_Aircraft_informed' with respect to takeoff mass and evaluations.](image-url)
Utility of informed operators in missile inlet design

![Graph showing total pressure recovery over evaluations for 'GADO_NIDA' and 'GADO_NIDA_informed'.]
Ongoing and future directions in Reduced Model Utilization

- Dynamic adjustment of degree of reliance on the reduced model
  - Cost
  - Recent accuracy
- More systematic ways of using very cheap reduced models
- Use of decomposability and sensitivity information for speedup
- Other strategies (genetic engineering)
- Other approximation methods (NNs)
Multi-objective optimization

• Most realistic problems are multi-objective
• The goal is to sample the set of non-dominated solutions (Pareto front)
  • A solution dominates another if it outperforms it in at least one objective and is not outperformed by it in any objective
  • On the Pareto front no solution is better than another when all objectives are considered simultaneously
• A better multi-objective optimizer is one that provides a dense, well-spread, accurate sampling of the Pareto front
Genetic Algorithms for multi-objective optimization

- A natural approach (population-based)
- Several methods exist in two basic categories
  - Methods that evaluate all-objectives for each solution
  - Methods that evaluate only one objective for each solution
- One of the best known methods is NSGA-II by Kalyanmoy Deb
OEGADO for two objective optimization

- Idea: to use one GADO optimizer for each objective
- The optimizer of each objective forms a reduced model of it (native model)
- The optimizers exchange their reduced models at specified intervals
- Each optimizer uses the imported reduced model (instead of the native model) with informed operators
Comparison of methods in the Welded Beam design domain

6808 objective evaluations

8000 objective evaluations

OEGA

NSGA-II
Comparison of methods in the Truss design domain

5882 objective evaluations

6000 objective evaluations

OEGA

NSGA-II
Conclusion

- GADO is a GA tailored for design optimization
- Its merit was demonstrated in several realistic and benchmark domains
- Further improvement expected using reduced models
- Several extensions (example: OEGADO)