Large-Scale Multi-Disciplinary Mass Optimization in the Auto Industry

Don Jones
Technical Fellow
General Motors Product Development

Presented at the MOPTA 2008 Conference, August 20, 2008
Outline

- The complexity of multi-disciplinary mass minimization
- A new MDO benchmark problem
- Four promising approaches
  - Evolutionary Strategies
  - Local search with search directions from local response surfaces
  - Local search with search directions from alternative physics models
  - Global search using kriging response surfaces
- Conclusion
Complexity of Multi-Disciplinary Mass Minimization

Typical Full Vehicle Problem is Large

Variables:

- Up to 200 part gages, 50 part materials, 100 shape variables

Simulations of Vehicle Performance

- Crash — Front, Side, Rear (several flavors of each)
- Noise Vibration and Harshness
- Durability

Performance constraints from the above simulations

- Up to 100 constraints

Computation Time

- Submit, run, post-process crash simulation can be 1-3 days

Desired Mass Reduction

- Usually about 10-20% of body and chassis mass

Goal
Choose variables to minimize the objective (mass), subject to meeting performance constraints
**Complexity of Multi-Disciplinary Mass Minimization**

No analytic crash derivatives; numerical ones problematic

- Thickness #1
- Thickness #2

Hinging / bending mode

Axial crush mode

Source: Mark Neal, GM R&D
Complexity of Multi-Disciplinary Mass Minimization

No analytic crash derivatives; numerical ones problematic

Source: Mark Neal, GM R&D
New MDO Benchmark Problem: The “MOPTA08” problem

- 1 objective function to be minimized
- 124 variables normalized to [0,1]
- 68 inequality constraints of form $g_i(x) \leq 0$.
- Constraints well normalized: 0.05 means 5% over requirement, etc.
- Based on kriging response surfaces to real automotive problem
- In real problem, can optimistically compute about 60 points/day
- Highly desirable to solve in $\leq 1$ month (30 days). Hence:
  - Max evaluations = $60 \times 30 = 1800 \sim 15$ times number variables
- Test problem comes with initial feasible point with objective $\sim 251.07$
- Optimum appears to have objective $\sim 222.74$
New MDO Benchmark Problem: The “MOPTA08” problem

Option 1: Use all the FORTRAN code except mopta08.f90. This gives you a subroutine func(n,m,x,f,g) that, for a given 124-vector of inputs x, gives you the objective f and the 68-vector of constraints g. The starting point is given in the file start.txt. Link this to your optimization code, calling func(n,m,x,f,g) whenever you need to evaluate a point (n=124, m=68).

Option 2: Compile all the FORTRAN code, including mopta08.f90. This gives you an executable that reads a file “input.txt” with the x values and writes a file “output.txt” with the objective and constraint values. Again the starting point is in start.txt.

input.txt

| 0.0666666666666667 |
| 0.0666666666666667 |
| 0.0666666666666667 |
| 0.3333333333333330 |
| 0.0666666666666667 |
| ... |
| 0.5500000000000000 |

124 input values

output.txt

| 251.0705961399711157 |
| -0.0132811254476478 |
| -0.6952387065097247 |
| -0.6030937399571389 |
| -0.0004037478531724 |
| ... |
| -0.0403392384222215 |

Objective value followed by the 68 constraints

Note: The FORTRAN files are huge and take a long time to compile. It is best to turn off compiler optimization.
What is “Good” Performance on the Benchmark?

- Good performance would be on a par with Powell’s COBYLA:
  - Number evaluations = 15 x number variables
  - Fully feasible solution (no constraint violations)
  - Objective ≤ 228 (at least 80% of potential reduction)

- Anything better is exciting

- Truly impressive would be
  - Something twice as fast (evaluations = 8x); or
  - Same speed (15x), but more accurate (objective ≤ 225)
Performance of “Standard” Gradient-Based Methods on the Benchmark
Generalized Reduced Gradient (using iSIGHT-FD)

Multiple of (n+1) Evaluations

Green: 0 violations
Yellow: 1-5 violations
Red: > 5 violations

Stays mostly feasible, but makes slow progress on the objective.
After 12 X to it isn’t even close to the optimum of 223
Performance of “Standard” Methods on the Benchmark

SQP (Harwell routine VF13)

Multiple of (n+1) Evaluations

Obj.

Number Evaluations

Green: 0 violations
Yellow: 1-5 violations
Red: > 5 violations

At 2X evaluations, constraints violated up to 12% of performance requirement.

At 5X evaluations, violations are as high as 6%

Fast progress on the objective, but slow to resolve constraint violations.

Final solution is shown green, but still has violations up to 0.5%
Finite difference methods are troublesome

- For gradient methods using finite differences:
  - Either the progress on objective will be slow
  - Or progress is fast but substantial constraint violations remain
- Noisiness of real simulation codes make finite differences tricky:
  - Finite difference step must be large enough to filter noise
  - But also small enough to give a good local approximation
Four Promising Approaches

1. Evolutionary Strategies
Promising new methods
Evolutionary Strategies

- Evolutionary Strategies (ESs) versus Genetic Algorithms (GAs)
  - GA evolves “x values” through crossover and mutation
  - ES evolves:
    - “x values” through crossover and mutation, AND
    - “strategy parameters” specifying direction & amount of mutation
    - As a result, an ES “evolves” good downhill directions for mutation, making mutation more effective than in a GA

- Big claim by Thomas Baeck
  - “An ES implicitly finds the gradient in order sqrt(n) evaluations”
  - If true, this is MUCH better than finite differences!
Promising new methods

How an Evolutionary Strategy Works

An individual $= \left( x_1, x_2, \ldots, x_n, \sigma_1, \sigma_2, \ldots, \sigma_n, \alpha_{12}, \alpha_{13}, \ldots, \alpha_{n-1,n} \right)$

Mutation “ellipsoid”

A point $(x_1, x_2)$

Amount of mutation in direction of first axis (after rotation)

Axis rotation $\alpha_{12}$

Amount of mutation in direction of first axis (after rotation)
Promising new methods

How an Evolutionary Strategy Works

\[
\begin{bmatrix}
  x_1^{(1)} & x_2^{(1)} & \sigma_1^{(1)} & \sigma_2^{(1)} & \alpha_{12}^{(1)} \\
  x_1^{(2)} & x_2^{(2)} & \sigma_1^{(2)} & \sigma_2^{(2)} & \alpha_{12}^{(2)} \\
  x_1^{(3)} & x_2^{(3)} & \sigma_1^{(3)} & \sigma_2^{(3)} & \alpha_{12}^{(3)} \\
  x_1^{(\mu)} & x_2^{(\mu)} & \sigma_1^{(\mu)} & \sigma_2^{(\mu)} & \alpha_{12}^{(\mu)}
\end{bmatrix}
\]

Parent population of \( \mu \) individuals

Select Two Parents At Random

\[
\begin{bmatrix}
  x_1^{(I)} & x_2^{(I)} & \sigma_1^{(I)} & \sigma_2^{(I)} & \alpha_{12}^{(I)} \\
  x_1^{(J)} & x_2^{(J)} & \sigma_1^{(J)} & \sigma_2^{(J)} & \alpha_{12}^{(J)}
\end{bmatrix}
\]

Recombine

\[
\begin{bmatrix}
  x_1^{(I)} & x_2^{(J)} & \frac{\sigma_1^{(I)} + \sigma_1^{(J)}}{2} & \frac{\sigma_2^{(I)} + \sigma_2^{(J)}}{2} & \frac{\alpha_{12}^{(I)} + \alpha_{12}^{(J)}}{2}
\end{bmatrix}
\]

\[\equiv \begin{bmatrix}
  x_1' & x_2' & \sigma_1' & \sigma_2' & \alpha_{12}'
\end{bmatrix}\]

Continued
Promising new methods

How an Evolutionary Strategy Works

Result of recombination (previous slide)

\[
\begin{bmatrix}
    x_1' & x_2' & \sigma_1' & \sigma_2' & \alpha_{12}' \\
    x_1' & x_2' & \sigma_1'' & \sigma_2'' & \alpha_{12}''
\end{bmatrix}
\]

Mutate strategy parameters

\[
\begin{bmatrix}
    x_1'' & x_2'' & \sigma_1'' & \sigma_2'' & \alpha_{12}''
\end{bmatrix}
\]

Mutate variables using new strategy parameters

Repeat all the above \( \lambda > \mu \) times to get \( \lambda \) offspring

Evaluate the offspring

Replace the entire population with the best \( \mu \) offspring
Promising new methods
Simple test problem to see if ES “finds the gradient”

- Simple Test Problem
  - Minimize \((1-x1)^2 + (1-x2)^2\)
  - Solution is \((1,1)\)
  - Start from \((-10, 1.5)\)
  - *Will an evolutionary strategy learn the gradient direction?*

- Use hand-coded “textbook” ES
  - Based on section 2.1 of Thomas Baeck’s textbook *Evolutionary Algorithms in Theory and Practice.*
  - Use several values of \((\mu, \lambda)\): \((15,35), (1,7), (5,10)\)
Promising new methods

Result of (15,35) ES on the simple test problem

We show the best individual after each iteration.

It seems the ES does indeed evolve a gradient direction.
Promising new methods
Result of (1,7) ES on the simple test problem

A (15,35) ES will use a lot of function evaluations.

On big problems, NUTECH suggests using (1,7).

But it seems the ability to evolve the gradient is compromised with a small population.

The “gradient property” comes with a cost!
Promising new methods

Result of (5,10) ES on the simple test problem

A (5,10) ES does an OK job of getting directions, but not as well as one with bigger (15,35) ES.
**Promising new methods**

The rate of convergence of ES seems linear in number variables.

I ran a (5,35) ES starting from (0,0,...,0) until all components of the best point were within 0.1 of optimum of (1,1,...,1).

Size of “success region” is going down *exponentially* as the number of variables increases, but evaluations are going up *linearly*. This is GOOD.
Promising new methods
Evolutionary Strategies

- BMW evaluated several optimizers and settled on an ES
  - They use the “ClearVu” optimizer from NUTECH
  - Technical development driven by Dr. Thomas Baeck

- In Reference [1], BMW and NUTECH describe problem with
  - 136 parameters (gages of body parts)
  - 47 performance constraints
  - 180 evaluations were made
  - Mass was reduced 13.5 kg

- Seems promising ... so let’s try it on the MOPTA08 benchmark!
  - Explored three constraint handling methods (see next slide)
  - Use ES with ($\mu, \lambda$) = (15,35)
Promising new methods

With constraints, which $\mu$ children do we select from $\lambda$ offspring?

1. Select best of those with minimum number of violations
   - Find minimum number of constraint violations among children
   - Kill all children with more violations; suppose $\gamma$ remain
   - If $\gamma \leq \mu$, use just these $\gamma$ children for the next parents
   - If $\gamma > \mu$, select the best $\mu$ on the objective for the next parents

2. Select best children ranked by objective + very high penalty

3. Filter method
   - Kill all children dominated by another child on both function value and max constraint violation; suppose $\gamma$ remain
   - If $\gamma \leq \mu$, use just these $\gamma$ children for the next parents
   - If $\gamma > \mu$, select the best $\mu$ on the objective for the next parents
**Promising new methods**

**Constraint Handling:** Select best of those with min violations

Promising new methods

Constraint Handling: Select best of those with min violations

- Green: 0 violations
- Yellow: 1-5 violations
- Red: > 5 violations

Takes long time to “learn” the feasible direction.

Pressure to stay feasible slows convergence.

Little progress from initial solution of 251
Promising new methods

Constraint Handling: Select best on f + heavy penalty

Green 0 violations
Yellow 1-5 violations
Red > 5 violations

Takes long time to “learn” the feasible direction.
Pressure to stay feasible slows convergence.
Little progress from initial solution of 251
Promising new methods

Constraint Handling: Select best of non-dominated children

Multiple of (n+1) Evaluations

0 2 4 6 8 10 12 14 16 18 20 22 24 26 28

Obj.

0 2 4 6 8 10 12 14 16 18 20 22 24 26 28

Filter method based on objective and max violation works better...

But filter method has trouble removing constraint violations.

At 24X, have solution with objective 227 with constraints violated up to 7% of performance requirement.
**Promising new methods**

**Constraint Handling: NUTECH ClearVu optimizer**

Constraints are handled with a proprietary method (seems to be a penalty).

A longer run may find a better solution.
Four Promising Approaches

2. Local Search With Search Directions From Local Surface Approximations
Local Search With Search Directions From Local Surface Approximations

LS-OPT

1. Sample in trust region current point (D-OPTIMAL, LHS, etc.)
2. Fit linear approximations to objective and constraints.
3. Solve problem with linear approximations and move current iterate to solution.
4. Adjust move limits for $x(k)$, $k=1,n$
   - Multiply move limit by 0.65 if $x(k)$ oscillated over last three iterations
   - Multiply move limit by factor = $0.65 + 0.35 \times \text{[percent x(k) moved to boundary]}$
Local Search With Search Directions From Local Surface Approximations

LS-OPT

Points for Fitting in Iter 2
Iterates
Optimum
Local Search With Search Directions From Local Surface Approximations

LS-OPT
1. Sample in trust region current point (D-OPTIMAL, LHS, etc.)
2. Fit linear approximations to objective and constraints.
3. Solve problem with linear approximations and move current iterate to solution
4. Adjust move limits for $x(k)$, $k=1,n$
   Multiply move limit by 0.65 if $x(k)$ oscillated over last three iterations
   Multiply move limit by factor $= 0.65 + 0.35 \times $[percent $x(k)$ moved to boundary]
**Local Search With Search Directions From Local Surface Approximations**

**LS-OPT**

At 6X evaluations, have solution with $F=228$ with constraints violated up to 10% of performance requirement.

At 24X evaluations, also have $F=228$ with just 0.2% violation.
Observations on LS-OPT

- At each iterate, need at least n+1 points to fit surface
  - No faster than finite differences
  - But can help if function is noisy

- Constraint violations handled in more detail
  - Constraints modelled separately (not lumped into a penalty)
  - My version of ES didn’t do this, and so constraints were difficult

- Does not expand trust region based on “success”
  - Hard to define “success” in a constrained problem

- Possible opportunity for improvement:
  - Reuse previous points more... this is what COBYLA does
Local Search With Search Directions From Local Surface Approximations

Importance of oscillation in LS-OPT

Uses oscillation to force shrinkage of trust region when near optimum.
Local Search With Search Directions From Local Surface Approximations

Powell’s COBYLA

Start with simplex. Each iteration drops one (bad) point and adds one that optimizes a merit function within a trust region. Merit function uses estimated gradients of objectives and constraints (from linear fit to simplex)

Never expands trust region, only shrinks it.

When the simplex becomes ill conditioned, and add point to “improve geometry”

When fail to get improvement, shrink simplex toward current best point.

Min \((1 - x_1)^2 + (1 - x_2)^2\) subject to \(x_2 \leq 4(1 - x_1)\)
Local Search With Search Directions From Local Surface Approximations

Powell’s COBYLA

Close up view of last triangle shrinking towards the best point.
Local Search With Search Directions From Local Surface Approximations

Powell’s COBYLA

Usually do one point per iteration in improving direction. This is GOOD.

But sometimes have to add geometry improving points that may not directly move towards optimum.

But this is better than evaluating N+1 on each iteration!
Good initial progress, but takes time to resolve constraints. Takes 15X to get feasible solution of 228.
Local Search With Search Directions From Local Surface Approximations

ORBIT algorithm of Wild / Regis / Shoemaker

- Same basic idea as COBYLA
  - Use approximation within trust region to generate points
  - Shrink trust region when fail to make improvement
  - Add points when needed to “improve geometry” used for interpolation

- Advantages over COBYLA:
  - Instead of linear approximation, uses radial-basis functions (RBF). RBF is more accurate, so can defer shrinking trust region and take longer steps
  - RBF interpolation can use all previous points (subject to well conditioning)
    So we can use more information than just N+1 points.

- Disadvantage: not yet extended to handle nonlinear constraints
  - A heuristic extension is trivial (solve constrained problem in trust region)
  - Proving the constrained version will converge is tricky
The amazing accuracy of radial basis functions

Contours of Branin Test Function

Contours of Kriging Interpolator Based on 21 Points
Local Search With Search Directions From Local Surface Approximations

Performance of ORBIT on test functions

Two flavors of ORBIT beat Powell’s latest NEWUOA on the Rosenbrock function.

Powell’s NEWUOA is based on quadratic approximations.
Local Search With Search Directions From Local Surface Approximations

Performance of ORBIT on test functions

This slide plots

**Success rate on 5 test problems with many random starting points**

**versus**

**Number of evaluations used, expressed as a multiple of \((N+1)\)**

Gets 80% success with about 8X evaluations.

**Outstanding Question:**

*Would this performance carry over to larger, constrained problems?*
Four Promising Approaches

3. Local Search With Search Directions From Alternative Physics Models
Local Search With Search Directions From Alternative Physics Models

Basic Idea

- Develop a simpler & faster model of the automobile, airplane, using:
  - Simpler discretization (coarser mesh, etc.); or
  - Simpler physics (linear static model instead of nonlinear dynamic)
- Match the simple and high-fidelity models at the current iterate
  - Make models agree on function values and, if available, gradients
- Optimize the simple model within a trust region
- The vector from current point to this optimum is our “search direction”
- Make a move in this direction on the high-fidelity model
- If don’t get improvement, try smaller step
First-order model management (Alexandrov/Lewis)

- Multiply the low-fidelity model by a correction factor
  - At the current iterate, the correction factor makes low-fidelity model agrees with the high-fidelity model on value and gradient
- Not relevant to automotive, because it require derivatives
- However, their results show possible improvement:
  - Aircraft problem
    - High fidelity = Nvier-Stokes
    - Low fidelity = Euler (8 times faster)
  - Using only high-fidelity model: 31 high fidelity evaluations
  - Using low & high fidelity models: 4 high + 51 low \sim 10.4 high
  - About a 3 times speed up
Local Search With Search Directions From Alternative Physics Models

Equivalent loads method (Gyun-Jin Park)

- High-fidelity model = fully dynamic model with plastic deformation
  Low-fidelity model = same model with only static, linear deformation

- Matching the models:
  - Run non-linear dynamic model and get final displacements
  - Find forces that -- if applied to the linear model -- give same displacements.
  - Apply these “equivalent loads” to the linear model
  - Linear model then agrees on function value (zero order model management)

- Convergence not proved, but whenever step direction computed with the low-fidelity model fails, can resort to a finite difference step on high-fidelity model

- Extreme speed ups reported by Park:
  - Up to 30 fold reduction in number of high fidelity evaluations
Four Promising Approaches

4. Global Search Using Kriging Response Surfaces
Global Search Using Kriging Response Surfaces

Kriging

- The kriging predictor smoothly interpolates the sampled points.
- Kriging also provides a computed “standard error” that estimates how much the predictor may be wrong.
Global Search Using Kriging Response Surfaces

Simplest way to use kriging for optimization

1. Do initial sampling
2. Fit surfaces
3. Optimize on surfaces
4. Add optimum to data set and go to Step 2.
Global Search Using Kriging Response Surfaces

Pitfalls with the simple approach in higher dimensions

- Looks great, but in high dimensions, there will be a lot of error
- Error will be especially high near boundary, hence
  - Optimization on the surface may yield “wild” points on boundary that look good only due to surface error (don’t “confirm”)
  - Adding wild points to data may not improve surface where needed
- Big idea in literature
  - Use the “standard error” to help manage possible surface error
  - Can avoid sampling where error is high (to avoid wild points); or
  - Can deliberately sample where the error is high, but where there is good chance to find a better point (to avoid local optimum)
Global Search Using Kriging Response Surfaces

Probability of Improvement criterion

The Probability of Improvement criterion specifies a target value of the objective $T$ (better than the current best point) and samples where the probability of exceeding $T$ is maximized.

Under mild conditions, it will generate search points that fill the space (i.e., are dense). Hence it guarantees ultimate convergence.
Global Search Using Kriging Response Surfaces

Optimizing using Probability of Improvement

This auxiliary function is highly multi-modal and hard to optimize!
Global Search Using Kriging Response Surfaces

Optimizing using Probability of Improvement
Global Search Using Kriging Response Surfaces

Optimizing using Probability of Improvement

Can get even better results by using several T targets to generate several points per iteration.

But this requires even more work on the auxiliary problems!
Global Search Using Kriging Response Surfaces

Status of literature

- Most of literature looks at small unconstrained problems
  - But methods that work well on small problems may fail on big ones due to higher error. Need to use better benchmarks, like MOPTA08.
  - Can handle constraints by factoring in probability of meeting them
    - You then maximize probability of *feasible* improvement
- Most existing methods have very high overhead
  - Need to solve one or several complex auxiliary problems
  - Complexity/number of these problems goes up with dimensionality
Global Search Using Kriging Response Surfaces

New developments in literature

- Avoid difficulty of auxiliary problems by using “candidate sets”
  - Somehow generate many candidate points for the next iterate
  - Use kriging to select *which* candidates to sample

- Examples
  - Regis & Shoemaker “Stochastic Radial Basis Function Method”
    - Best candidate selected based on kriging prediction and distance to nearest sampled point
  - Villmonteix, Vazquez, Walter’s “IAGO” method
    - Best candidate selected based on how sampling the point is expected to reduce the uncertainty of where the optimum is
    - But these methods may fail in high dimensions, since candidate set may not have any points that fall in “good” region of space.
Conclusions

- Evolutionary Strategies
  - May have trouble with heavily constrained problems...jury is out
- Local search with directions based on local response surfaces
  - Radial basis function methods are very promising!
    But need to explore ways handle nonlinear constraints
- Local search with directions based on alternative physics models
  - Very promising where alternative physics models exist
- Global search with kriging models
  - Need more focus on constraints and tractability in higher dimensions
  - Hopefully, the MOPTA08 test problem will focus efforts to address these gaps and provide a benchmark for comparing new approaches