Distributed Design and Hierarchical Optimization

Sigurd A. Nelson II
Design Laboratory
Department of Mechanical Engineering and Applied Mechanics
University of Michigan
Outline of Talk

1. Nonlinear Programs and Hierarchy

2. NLP Algorithms


4. Numerical Examples
Nonlinear Programs

Formal description of an optimization problem.

\[
\begin{align*}
\min_{x \in \mathbb{R}^n} & \quad f(x) \\
\text{s. t.} & \quad g(x) \leq 0 \\
& \quad h(x) = 0
\end{align*}
\]

Gives a consistent manner with which to compare different decisions/engineering designs.

Solutions are difficult to find when \( n \) is large and \( f(x) \), \( g(x) \) or \( h(x) \) are very nonlinear, have many components, or are expensive to compute.
Hierarchical NLP

Hierarchical Form

\[
\min_{x \in \mathbb{R}^n} \quad f_0(x_0) + \sum_{j=1}^p f_j(x_0, x_j)
\]

\[
g_0(x_0) \leq 0
\]

\[
h_0(x_0) = 0
\]

s. t.

\[
g_j(x_0, x_j) \leq 0
\]

\[
h_j(x_0, x_j) = 0
\]

Linking Variables: A vector of variables \( x_0 \) common to all groups of functions.

Subproblems A group of functions which depends only on the vector of linking variables and upon a single subvector \( x_j \). Often written as a small NLP

\[
\min_{x_j \in \mathbb{R}^n} \quad f_j(x_0, x_j)
\]

s. t.

\[
g_j(x_0, x_j) \leq 0
\]

\[
h_j(x_0, x_j) = 0
\]
Properties of Hierarchies

- Transforms a large problem into smaller manageable pieces.
- Allows for parallel implementation.
- Each subproblem is autonomous.
- Smaller problems are easier to solve.
Using general NLP algorithms

There are several very successful algorithms for solving general NLP.

Experience and proven track record

Substantial theoretical analysis
  Global Convergence Properties
  Local Convergence Rates.

Will not have the desirable hierarchical properties.

Can be difficult to apply successfully for large NLP.

Can general NLP algorithms be modified in order to accommodate hierarchical NLP while still retaining the theoretical properties of the original algorithm?
Modifying general NLP algorithms

Add a step within the existing NLP algorithm, during which the linking variables are held constant and the subproblems are solved.

1. Start at some initial point

2. Treat the linking variables as parameters and solve each subproblem

3. Solve an approximate problem, giving a candidate point.

4. If the candidate point is acceptable, move to the candidate point and go to step 2. Otherwise, modify the approximate problem and go to step 3.

This step can not be added without taking care to preserve the theoretical properties of the original algorithm.
Guidelines for Modifying General NLP algorithms

1. In order to retain any established local convergence properties, decomposition (i.e., solving subproblems separately) is not performed when near a solution. That is to say, the additional step is not used.

2. In order to retain the global convergence properties of the original algorithm, coordination between subproblems is performed using an approximate problem which has the same form as the unmodified algorithm. That is, the coordination uses all of the constraints and all of the variables in the original problem.

3. Subproblems are used not only to improve their respective objective functions while maintaining or obtaining feasibility, but also to give better estimates of other quantities used in the algorithm, such as penalty parameters, Hessian estimates, and trust region radii.
A Trust Region Example-Yuan '91

Nonsmooth Penalty Function

\[ \Phi(x) = f(x) + \sigma \| c(x)^+ \|_\infty \]

Candidate Points

\[ \min_{s \in \mathbb{R}^n} \varphi(s) = \nabla f(x^k)s + \frac{1}{2} s^T B^k s + \sigma \| (\nabla c(x^k)s + c(x^k)^+ \|_\infty \]

s. t. \[ \| s \|_\infty \leq \Delta^k \]

Altering the Trust Region

\[ \Delta^{k+1} = \begin{cases} \max \left\{ 2\Delta^k, 4\| s \|_\infty \right\} & 0.9 < r \\ \Delta^k & 0.1 \leq r \leq 0.9 \\ \max \left\{ \frac{\Delta^k}{4}, \frac{\| s \|_\infty}{2} \right\} & r < 0.1 \end{cases} \]

where \[ r = \frac{\Phi(x^k) - \Phi(x^k + s)}{\varphi(0) - \varphi(s)} \]
The Modified Algorithm

1. Set the iteration counter $k = 1$. Choose some $x^1$ as an initial point and a set of $B^1_j$ as initial sparse Hessian estimates. Also choose some $\sigma^1 > 0$ and $\Delta^1 > 0$.

2. If the current iterate $x^k$ is not near the solution $x^*$, then each subproblem will be solved using the current value for $\sigma$ and $\Delta$ and applying Yuan’s algorithm to each subproblem. Upon completion, set the parameters for the coordination problem by defining $\sigma^k = \max(\sigma^1_j^\dagger)$, $\Delta^k = \min(\Delta^1_j^\dagger)$ where the superscript $^\dagger$ is used to denote parameter values upon completion of each subproblem.

3. Using $B^k = \sum_{j=0}^{p} B^k_j$ solve the approximate problem and denote the solution $s$. If $s = 0$ then stop.

4. Calculate the penalty function $\Phi(x)$, approximate penalty function $\varphi(s)$ and the ratio of the two $r$. If $r > 0$ go to Step 5, otherwise set $\Delta = \|s\|_\infty/4$ and go to Step 3.

5. Alter the trust region radius $\Delta^k$. Generate $B_{j}^{k+1}$ for each $j = 0..p$.

6. If appropriate, modify then modify $\sigma$ and $\Delta$. Set $x^{k+1} = x^k + s^k$, $k = k + 1$ and go to Step 2.
Using Subproblems for Better Estimates for Parameters

The Penalty Parameter $\sigma$ and Trust Region Radii $\Delta$

Each subproblem uses the current value to start with.

Each subproblem alters their own values of $\sigma$ and $\Delta$ as appropriate.

When each subproblem is finished, redefine $\sigma$ and $\Delta$ before coordination.

$$\sigma = \max(\sigma_j^\dagger)$$

$$\Delta = \min(\Delta_j^\dagger)$$
Updating Hessians (Griewank & Toint)

Each group of functions is dependent only on two sets of variables

\[ \nabla L_j(x_0, x_j, \lambda_j) = \begin{bmatrix} \nabla x_0 L_j \\ 0 \\ \nabla x_j L_j \\ 0 \end{bmatrix} \text{ and } s = \begin{bmatrix} s_0 \\ 0 \\ s_j \\ 0 \end{bmatrix} \]

\[ y = \nabla L(x_0 + s_0, x_j + s_j, \lambda_j) - \nabla L(x_0, x_j, \lambda_j) \]

\[ B_j^{k+1} = B_j^k + \frac{yy^T}{y^Ts} - \frac{(B_j^k)(s^TB_j^k)}{(s^TB_j^k)} \]

\[ B_j^{k+1} = B_j^k + \frac{(y - B_j^k)s^T + s(y - B_j^k)^T}{s^Ts} - \frac{s^T(y - B_j^k)s}{(s^Ts)^2} \]

\[ B_j^{k+1} = B_j^k + \frac{(y - B_j^k)y^T + y(y - B_j^k)^T}{y^Ts} - \frac{y^T(y - B_j^k)yy^T}{(y^Ts)^2} \]

This is done even at the subproblem level.
Testing for Closeness

Only solve the subproblems as separate problems if the current iterate is not close enough to the solution to see the theoretical q-superlinear convergence rate.

Need a test which depends upon definitions of q-superlinear convergence and on the sufficient conditions in the proof of q-superlinear convergence.

From the definition of q-superlinear convergence:

$$\|x^{k+1} - x^k\| \leq \rho \|x^k - x^{k-1}\|$$

Trust Region constraint is inactive:

$$\|s^k\|_\infty < \Delta^k$$

If both of these conditions are satisfied for two consecutive iterations, the subproblems are not solved independently.
Example 1- The Combustion of Propane

\[
\begin{align*}
\min_{\mathbf{x}} & \quad \sum_{j = 1}^{13} f_j^2(\mathbf{x}) \\
\text{(The ICASE MDO collection)}
\end{align*}
\]

\[
\begin{align*}
f_1 &= x_1 + x_4 - 3 \\
f_2 &= 2x_1 + x_2 + x_4 + x_7 + x_8 + x_9 + 2x_{10} - R \\
f_3 &= 2(x_2 + x_5) + x_6 + x_7 - 8 \\
f_4 &= 2x_3 + x_9 - 4R \\
f_5 &= K_5 x_2 x_4 - x_1 x_5 \\
f_6 &= K_6 \sqrt{x_2 x_4} - x_6 \sqrt{(P x_1)/x_{11}} \\
f_7 &= K_7 \sqrt{x_1 x_2} - x_7 \sqrt{(P x_4)/x_{11}} \\
f_8 &= K_8 x_1 - (P x_4 x_8)/11 \\
f_9 &= K_9 x_{11} \sqrt{x_3 - x_4 x_9 \sqrt{(P x_4)/x_{11}}} \\
f_{10} &= K_{10} x_1^2 - (x_4 x_{10} P)/11 \\
f_{11} &= x_{11} - (x_1 + x_2 + x_4 + x_7 + x_{12} + x_{13}) \\
f_{12} &= x_{12} - (x_5 + x_6) \\
f_{13} &= x_{13} - (x_3 + x_8 + x_9 + x_{10})
\end{align*}
\]
Example 1 - The Decomposition

Subproblem 1

\[
\min_{x_5, x_6} \sum_{j = \{3, 5, 6, 12\}} f_j^2(x_0, x_5, x_6)
\]

Subproblem 2

\[
\min_{x_3, x_8, x_9, x_{10}} \sum_{j = \{2, 4, 8, 9, 10, 13\}} f_j^2(x_0, x_3, x_8, x_9, x_{10})
\]

Remaining Functions (Subproblem 0)

\[
\sum_{j = \{1, 7, 11\}} f_j^2(x_0)
\]

Linking Variables

\[
x_0 = [x_1, x_2, x_4, x_7, x_{11}, x_{12}, x_{13}]
\]
Example 1- Algorithm Performance

Iteration history for Yuan’s algorithm and the modified algorithm when applied to the propane combustion example. Gradient evaluations are marked by either a circle, an asterisk, or a cross. One increment along the abscissa represents either a function evaluation or a gradient evaluation of the functions in subproblem 0, subproblem 1, or subproblem 2.
### Example 1 - Summary of Results

<table>
<thead>
<tr>
<th></th>
<th>Yuan's Algorithm</th>
<th>Modified Algorithm (SDP)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of function calls to subproblem 0</td>
<td>145</td>
<td>31</td>
</tr>
<tr>
<td>Number of function calls to subproblem 1</td>
<td>145</td>
<td>45</td>
</tr>
<tr>
<td>Number of function calls to subproblem 2</td>
<td>145</td>
<td>77</td>
</tr>
<tr>
<td>Number of gradient calls to subproblem 0</td>
<td>80</td>
<td>24</td>
</tr>
<tr>
<td>Number of gradient calls to subproblem 1</td>
<td>80</td>
<td>39</td>
</tr>
<tr>
<td>Number of gradient calls to subproblem 2</td>
<td>80</td>
<td>51</td>
</tr>
<tr>
<td>Number of times the approximate problem was solved</td>
<td>80</td>
<td>24</td>
</tr>
</tbody>
</table>
Example 2 - A Parking Brake

\[
\begin{align*}
\min_{x \in \mathbb{R}^n} & \quad f_1(x) + f_2(x) \\
\text{s. t.} & \quad g(x) \leq 0 \\
& \quad h(x) = 0
\end{align*}
\]

(Krishnamachari)

\[
\begin{align*}
f_1 &= 100x_4x_5 - x_9 - x_{11} \\
f_2 &= -30x_6x_7^3 + 100x_6x_7 \\
g_1 &= x_9 - 2K_1x_8x_{10}x_{13} \\
g_2 &= x_1x_8x_{10} + \frac{5}{9}(x_2(x_3 - x_1) - x_3(x_2 - x_1)) \\
g_3 &= x_{11} - x_4x_5^3/12 \\
g_4 &= x_{12} - K_4(x_2 - x_1) \\
g_5 &= x_{13} - x_8x_{14} \\
g_6 &= x_{14} - 0.8(x_3/x_8)
\end{align*}
\]

\[
\begin{align*}
h_1 &= 2.94 - 2K_7x_8x_{10}x_{13} \\
h_2 &= K_8 - 2K_7x_8x_{10}x_{13} \\
h_3 &= x_1 - x_3 + 0.45x_8 \\
h_4 &= (x_5x_{12})/2 - K_{10}x_{11} \\
h_5 &= K_{11}x_2 - x_1x_6x_7 \\
h_6 &= \frac{K_{12}x_2x_9}{x_1} - \frac{x_6x_7^3}{0.45x_8\left(1.25 + \frac{0.15x_1}{x_3 - 0.45x_8}\right)^2} \\
h_7 &= x_1 - x_2 + 5
\end{align*}
\]
Example 2 - The Decomposition

Subproblem 1
\[
\begin{align*}
\text{min} & \quad f_1(x_4, x_5, x_9, x_{11}) \\
\text{s. t.} & \quad g_i(x_0, x_1) = 0 \quad i = 1, 3, 5 \\
& \quad h_i(x_0, x_1) \leq 0 \quad i = 7, 8, 10 \\
& \quad 5 \leq x_4 \leq 20 \\
& \quad 5 \leq x_5 \leq 30 
\end{align*}
\]

Subproblem 2
\[
\begin{align*}
\text{min} & \quad f_2(x_6, x_7) \\
\text{s. t.} & \quad g_i(x_0, x_2) = 0 \quad i = 2, 4, 6 \\
& \quad h_i(x_0, x_2) \leq 0 \quad i = 9, 11, 12, 13 \\
& \quad 10 \leq x_1 \leq 24 \\
& \quad 10 \leq x_2 \leq 102 \\
& \quad 3 \leq x_3 \leq 183 \\
& \quad 5 \leq x_6 \leq 20 \\
& \quad 5 \leq x_7 \leq 30 \\
& \quad 160 \leq x_8 \leq 240 
\end{align*}
\]

Linking Variables \( x_0 = [x_8, x_9, x_{10}, x_{14}] \)
Example 2 - Algorithm Performance

Iteration history for Yuan’s algorithm and the modified algorithm when applied to the parking brake example. Gradient evaluations are marked by either a circle or a cross. One increment along the abscissa represents either a function evaluation or a gradient evaluation of the functions in either subproblem 1 or subproblem 2.
## Example 2 - Summary of Results

<table>
<thead>
<tr>
<th></th>
<th>Yuan's Algorithm</th>
<th>Modified Algorithm (SDP)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of function calls to subproblem 1</td>
<td>146</td>
<td>55</td>
</tr>
<tr>
<td>Number of function calls to subproblem 2</td>
<td>146</td>
<td>67</td>
</tr>
<tr>
<td>Number of gradient calls to subproblem 1</td>
<td>47</td>
<td>23</td>
</tr>
<tr>
<td>Number of gradient calls to subproblem 2</td>
<td>47</td>
<td>29</td>
</tr>
<tr>
<td>Number of times the approximate problem was solved</td>
<td>47</td>
<td>11</td>
</tr>
</tbody>
</table>
Conclusions

Algorithms intended for general NLP can be modified to accommodate hierarchical NLP without compromising the theoretical properties of the original algorithm.

Even for a serial implementation, the modified algorithms (SDP) outperform the original algorithms in terms of number of iterations, number of function calls, and number of times coordination must be performed.