Definition of separability

Piecewise-Linear Optimization

Restricted Basis Entry rules

Example

Refining the Grid
A function \( f(x_1, x_2, \cdots, x_n) \) is **separable** if it can be written as a sum of terms, each term being a function of a **single** variable:

\[
 f(x_1, x_2, \cdots, x_n) = \sum_{i=1}^{n} f_i(x_i)
\]

**separable**

\[
\sqrt{x_1} + 2 \ln x_2
\]

**not separable**

\[
x_1x_2 + x_3
\]

**examples**

\[
x_1^2 + 3x_1 + 6x_2 - x_2^2
\]

\[
5x_1/x_2 - x_1
\]

© Dennis Bricker, U. of Iowa, 1996

**Piecewise-Linear (separable) Programming**

We approximate a nonlinear separable function by a piecewise-linear function:

\[ f(z) \]

© Dennis Bricker, U. of Iowa, 1996
There are two ways to formulate the piecewise-linear programming problem as a Linear Programming problem:

- "LAMBDA" formulation
- "DELTA" formulation

Suppose that \( f(z) \) is a convex function. Let \( \zeta_0, \zeta_1, \ldots \) be specified "grid points", and \( \lambda_0, \lambda_1, \ldots \) be "weights" where

\[
\sum_i \lambda_i = 1, \quad \lambda_i \geq 0
\]
Any value of \( z \) in the interval between the left-most and the right-most grid point may be expressed as a "convex combination" of the grid points:

\[
z = \lambda_0 \zeta_0 + \lambda_1 \zeta_1 + \lambda_2 \zeta_2 + \lambda_3 \zeta_3
\]

where

\[
\sum_{i} \lambda_i = 1, \lambda_i \geq 0
\]

With the same "weights" used in writing the convex combination of the grid points,

\[
z = \lambda_0 \zeta_0 + \lambda_1 \zeta_1 + \lambda_2 \zeta_2 + \lambda_3 \zeta_3
\]

we approximate \( f(z) \) as a convex combination of the function values at the grid points:

\[
f(z) \approx \lambda_0 f(\zeta_0) + \lambda_1 f(\zeta_1) + \lambda_2 f(\zeta_2) + \lambda_3 f(\zeta_3)
\]
Suppose that $f(z)$ is piecewise linear and \textit{convex}... 

By the definition of "convex", this means that every chord of the graph of $f(z)$ lies \textit{on} or \textit{above} the graph!

Consider now the various convex combinations of grid points yielding $z=1.75$.....

\begin{align*}
1.75 & = \frac{5}{12} (0) + \frac{7}{12} (3) \\
1.75 & = \frac{5}{8} (1) + \frac{3}{8} (3) \\
1.75 & = \frac{1}{2} (1) + \frac{1}{4} (2) + \frac{1}{4} (3) \\
1.75 & = \frac{1}{4} (1) + \frac{3}{4} (2) \\
\text{etc.}
\end{align*}

©Dennis Bricker, U. of Iowa, 1996
Each set of "weights" in the convex combinations (which yield the same z) when used to weight the function values, will result in a different approximation to f(z).

\[ 1.75 = \frac{5}{12} f(0) + \frac{7}{12} f(3) \]

\[ f(1.75) \approx \frac{5}{12} f(0) + \frac{7}{12} f(3) = \frac{15}{4} \]

The point \( \left( \sum \lambda_i \xi_i, \sum \lambda_i f(\xi_i) \right) \) lies on a chord of the graph which is, of course, on or above the graph. That is, \( \sum \lambda_i f(\xi_i) \) is in general an \textbf{overestimate} of \( f(z) \).
\[ 1.75 = \frac{5}{8} (1) + \frac{3}{8} (3) \]
\[ f(1.75) \approx \frac{5}{8} f(1) + \frac{3}{8} f(3) = \frac{5}{2} \]
\[ 1.75 = \frac{1}{4} (1) + \frac{3}{4} (2) \]

\[ f(1.75) \approx \frac{1}{4} f(1) + \frac{3}{4} f(2) = \frac{7}{4} \]

Of the various ways to express \( z \) as a convex combination of grid points, the way which results in the \textit{minimum} value for an approximation of \( f(z) \) is that which assigns positive weights only to the grid points immediately to the left and right of \( z \).

\textit{This is the convex combination which best approximates} \( f(z) \)!
When minimizing a convex function $f(z)$ by choosing the weights in the convex combination, then,

...at most TWO $\lambda_i$'s will be positive, and these will be weights of adjacent grid points!

*What happens if $f(z)$ is NOT convex?*
When \( f(z) \) is not convex, the chords do not all lie on or above the graph, and one can choose convex combinations of grid points yielding approximations of \( f(z) \) which are underestimates of the function.

For example, in this figure, the lowest (and the worst!) estimate of \( f(\zeta_3) \) would be obtained by expressing \( \zeta_3 \) as a convex combination of \( \zeta_1 \) and \( \zeta_4 : \quad \zeta_3 = \lambda_1 \zeta_1 + \lambda_4 \zeta_4 \) with \( f(\zeta_3) \) approximated by \( \lambda_1 f(\zeta_1) + \lambda_4 f(\zeta_4) \) whereas the "best" approximation is obtained by

\[
\zeta_3 = 0\zeta_1 + 0\zeta_1 + 1\zeta_3 + 0\zeta_4
\]
In the "lambda" formulation, a special variable (λ) was defined for each grid point. In the "delta" formulation, a special variable (δ) will be defined for each interval between grid points, i.e., for each linear piece.

There are two variations....

Define constants:

\[ \Delta \xi_i = \xi_i - \xi_{i-1} \]
\[ \Delta f_i = f(\xi_i) - f(\xi_{i-1}) \]

Define variables:

\[ 0 \leq \delta_i \leq 1 \text{ OR } 0 \leq \Delta i \leq \Delta \xi_i \]
variation #1

each variable is bounded between zero and 1.00

\[ z = \zeta_0 + \sum_{i=1}^{p} (A \zeta_i) \delta_i \]

\[ f(z) \approx f(\zeta_0) + \sum_{i=1}^{p} (A f_i) \delta_i \]

\[ 0 \leq \delta_p \leq \cdots \leq \delta_1 \leq 1 \]

© Dennis Bricker, U. of Iowa, 1996

variation #2

each variable has an upper bound equal to the length of the interval

\[ \Delta_i = (A \zeta_i) \delta_i \]

\[ f(z) \approx f(\zeta_0) + \sum_{i=1}^{p} \left( \frac{A f_i}{A \zeta_i} \right) \Delta_i \]

\[ 0 \leq \Delta_i \leq A \zeta_i \]

© Dennis Bricker, U. of Iowa, 1996
"Delta" form of Separable Programming

In either variation, at most ONE variable is allowed to be at an intermediate value (not a bound), i.e., BASIC when we use UBT (upper bounding technique)

variation #1

\[ z = \zeta_0 + \sum_{i=1}^{p} (\Delta \zeta_i) \delta_i \]

\[ f(z) \approx f(\zeta_0) + \sum_{i=1}^{p} (\Delta f_i) \delta_i \]

\[ 0 \leq \delta_p \leq \ldots \leq \delta_1 \leq 1 \]

variation #2

\[ z = \zeta_0 + \sum_{i=1}^{p} \Delta_i \]

\[ f(z) \approx f(\zeta_0) + \sum_{i=1}^{p} \left( \frac{\Delta f_i}{\Delta \zeta_i} \right) \Delta_i \]

\[ 0 \leq \Delta_i \leq \Delta \zeta_i \]

© Dennis Bricker, U. of Iowa, 1996

If we are:

• minimizing a non-convex function

&/or

• optimizing over a nonconvex region

e.g., \( g(x) \leq 0 \) where \( g \) is non-convex,

Then the simplex method will NOT yield a basic solution in which

• at most two (adjacent) \( \lambda \)'s are basic
  \((\lambda\text{-formulation})\)

• only one \( \delta \) is basic
  \((\delta\text{-formulation})\)

© Dennis Bricker, U. of Iowa, 1996
In these cases, a "restricted basis entry" rule may be implemented, which will guarantee that the solution satisfies the desired properties,

- at most 2 $\lambda$'s are in the basis, in which case they have consecutive indices ( $\lambda$-formulation)
- at most one $\delta$ is in the basis ( $\delta$-formulation)

but unfortunately will not guarantee an optimal solution!

"Lambda" formulation
Special set: $[\lambda_{i0}, \lambda_{i1}, \ldots \lambda_{ip}]$

$\lambda_{ij}$ is positive for at most TWO values of $j$, in which case they are consecutive indices.

How can we modify the simplex method so as to impose this restriction?
"Lambda" formulation
Special set: \( \{ \lambda_{i0}, \lambda_{i1}, \ldots \lambda_{ip} \} \)

RBE Rule

If 2 adjacent weights are in the basis, then no other weight from the same set may be considered for basis entry;
if only one weight \( \lambda_{ij} \) is basic, then only \( \lambda_{i,j-1} \) & \( \lambda_{i,j+1} \) are considered as candidates for basis entry.

Note that this modification of the simplex method does not guarantee optimality, unless the function being minimized is a convex function!
"Delta" formulation
Special set: \([\delta_{i1}, \delta_{i2}, \ldots \delta_{ip}]\)

Constraint
\(\delta_{ij}\) is at an intermediate level (neither lower nor upper bound) for at most a single \(j\) (i.e., if UBT is used, at most one variable in the set is basic.)

\[\begin{align*}
\text{How can we modify the simplex method so as to impose this restriction?}
\end{align*}\]

"Delta" formulation
Special set: \([\delta_{i1}, \delta_{i2}, \ldots \delta_{ip}]\)

RBE Rule
\(\delta_{ij}\) is not considered for basis entry unless:
- no other variable in the set is basic
- \(\delta_{i,j-1}\) is at upper bound
- \(\delta_{i,j+1}\) is at lower bound
"Delta" formulation
Special set: \( \{\delta_{i1}, \delta_{i2}, \ldots \delta_{ip}\} \)

**Example**

\[
\begin{align*}
1, & 1, 1, 1, \frac{3}{8}, 0, 0, 0, 0, 0 \\
U & B & L
\end{align*}
\]

no variable may enter the basis

---

"Delta" formulation
Special set: \( \{\delta_{i1}, \delta_{i2}, \ldots \delta_{ip}\} \)

**Example**

considered for basis entry

\[
\begin{align*}
1, & 1, 1, 1, 1, 0, 0, 0, 0, 0, 0 \\
U & L
\end{align*}
\]

In this case, no variable in the set is in the basis set \( B \);
one variable in \( L \) and one variable in \( U \) may enter \( B \)
Example

A company manufactures three products, using three limited resources:

<table>
<thead>
<tr>
<th>resources</th>
<th>product</th>
<th>available supply</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>ingredient #1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>ingredient #2</td>
<td>10</td>
<td>4</td>
</tr>
<tr>
<td>ingredient #3</td>
<td>3</td>
<td>2</td>
</tr>
</tbody>
</table>

Because of various factors (e.g., quantity discounts, use of overtime, etc.) the profits per unit decrease as sales increase:

<table>
<thead>
<tr>
<th>product A</th>
<th>sales</th>
<th>profit ($/unit)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0-40</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>40-100</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>100-150</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>over 150</td>
<td>7</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>product B</th>
<th>sales</th>
<th>profit ($/unit)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0-50</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>50-100</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>over 100</td>
<td>3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>product C</th>
<th>sales</th>
<th>profit ($/unit)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0-100</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>over 100</td>
<td>4</td>
</tr>
</tbody>
</table>

Determine the most profitable mix of products.
\textbf{Separable Programming}  

$P_A(X_A)$  

\begin{align*}
\text{Profit} & \quad P_A(X_A) \\
$1000$ & \quad \text{slope } $7/\text{unit} \\
$500$ & \quad \text{slope } $9/\text{unit} \\
& \quad (150,1340) \\
& \quad \text{slope } $10/\text{unit} \\
& \quad (40,400) \\
& \quad 50 \quad 100 \quad 150 \\
\end{align*}

\textbf{©Dennis Bricker, U. of Iowa, 1996}

$P_B(X_B)$  

\begin{align*}
\text{Profit} & \quad P_B(X_B) \\
$500$ & \quad \text{slope } $6/\text{unit} \\
& \quad (50,300) \\
& \quad \text{slope } $4/\text{unit} \\
& \quad (100,500) \\
& \quad 50 \quad 100 \quad 150 \\
\end{align*}

\textbf{©Dennis Bricker, U. of Iowa, 1996}

$P_C(X_C)$  

\begin{align*}
\text{Profit} & \quad P_C(X_C) \\
$500$ & \quad \text{slope } $5/\text{unit} \\
& \quad (100,500) \\
& \quad 50 \quad 100 \quad 150 \\
\end{align*}

\textbf{©Dennis Bricker, U. of Iowa, 1996}
Maximize \( p_A(x_A) + p_B(x_B) + p_C(x_C) \)
subject to
\[
\begin{align*}
x_A + 2x_B + x_C & \leq 1000 \\
10x_A + 4x_B + 5x_C & \leq 7000 \\
3x_A + 2x_B + x_C & \leq 4000 \\
x_A \geq 0, x_B \geq 0, x_C \geq 0
\end{align*}
\]

Each profit function \( p_A, p_B, \) & \( p_C, \)
is piecewise linear.

We can reformulate this as a linear programming problem in two ways:

- \( \Delta \) "delta" formulation
  one variable for each interval

- \( \Lambda \) "lambda" formulation
  one variable for each grid point
Define
\[ \Delta_{A1} = \text{quantity of } A \text{ produced at } \$10/\text{unit profit,} \]
\[ \Delta_{A2} = \text{quantity of } A \text{ produced at } \$9/\text{unit profit,} \]
... etc.

so that
\[ p_A(x_A) = 10\Delta_{A1} + 9\Delta_{A2} + 8\Delta_{A3} + 7\Delta_{A4} \]
\[ 0 \leq \Delta_{A1} \leq 40 \]
\[ 0 \leq \Delta_{A2} \leq 60 \quad = 100-40 \]
\[ 0 \leq \Delta_{A3} \leq 50 \quad = 150-100 \]
\[ 0 \leq \Delta_{A4} \]

Since the simplex algorithm will maximize, the optimum will NOT use a positive value for \( \Delta_{A2} \) unless the more profitable \( \Delta_{A1} \) has reached its upper limit (40), etc.

Thus, the simplex algorithm will naturally impose the restricted basis entry (RBE) rules.

(These profit functions exhibit "decreasing returns to scale".... )
<table>
<thead>
<tr>
<th>Max</th>
<th>$\Delta A_1$</th>
<th>$\Delta A_2$</th>
<th>$\Delta A_3$</th>
<th>$\Delta A_4$</th>
<th>$\Delta B_1$</th>
<th>$\Delta B_2$</th>
<th>$\Delta B_3$</th>
<th>$\Delta C_1$</th>
<th>$\Delta C_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>4</td>
<td>3</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>lower bounds</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>upper bounds</td>
<td>40</td>
<td>60</td>
<td>50</td>
<td>$\infty$</td>
<td>50</td>
<td>50</td>
<td>$\infty$</td>
<td>100</td>
<td>$\infty$</td>
</tr>
</tbody>
</table>

We require an upper bound (right-most grid point) for each product A, B, and C. Let's arbitrarily use 1000 for each.

Define a weight for each grid point:

- $\lambda_{A_0} \leftrightarrow 0$
- $\lambda_{A_1} \leftrightarrow 40$
- $\lambda_{A_2} \leftrightarrow 100$
- $\lambda_{A_3} \leftrightarrow 150$
- $\lambda_{A_4} \leftrightarrow 1000$

©Dennis Bricker, U. of Iowa, 1996
Substitute

\[ p_A(x_A) = 0 \lambda_{A0} + 400 \lambda_{A1} + 940 \lambda_{A2} + 1340 \lambda_{A3} + 6590 \lambda_{A4} \]

and

\[ x_A = 0 \lambda_{A0} + 40 \lambda_{A1} + 100 \lambda_{A2} + 150 \lambda_{A3} + 1000 \lambda_{A4} \]

... etc.

©Dennis Bricker, U. of Iowa, 1996