CS 524: Introduction to Optimization
Lecture 25 : Piecewise linear modeling

Michael Ferris

Computer Sciences Department
University of Wisconsin-Madison

November 1, 2023

Overview

- Consider real functions of a single variable $f : \mathbb{R} \mapsto \mathbb{R}$.
- Approximate these functions for optimization by a piecewise-linear function.
- Two cases:
  1. continuous, bounded domain
  2. possibly multi-valued, infinite domains
- In model, impose constraints $(x, y) \in \text{graph}(f)$, i.e. $vy = f(x)$.
- Then build remaining feature of model around this, e.g.:

$$\min \sum_i y_i \text{ s.t. } (x_i, y_i) \in \text{graph}(f_i), Ax + By \leq d$$

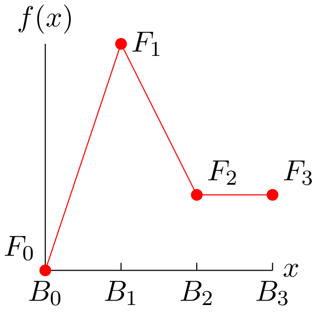
Piecewise Linear Functions

- A very common structure used in modeling is a piecewise-linear function of a scalar variable $x \in [B_0, B_n]$

$$f(x) = m_i x + c_i, \quad x \in [B_{i-1}, B_i] \quad \forall i = 1, \dots, n$$

Sample applications using PLFs

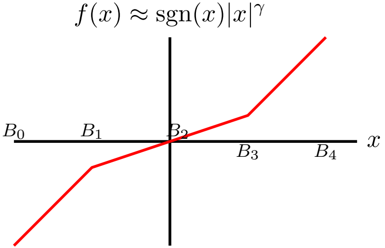
- Gas network optimization [Martin et al., 2006]
- Transmissions expansion planning [Alguacil et al., 2003]
- Oil field development [Gupta and Grossmann, 2012]
- Hydro Scheduling [Borghetti et al., 2008]
- Thermal unit commitment [Carrion and Arroyo, 2006]
- Sales resource allocation [Lodish, 1971]



Multiple Choice Model

- Want to model $(\forall i)$

$$f(x) = m_i x + c_i, \; x \in [B_{i-1}, B_i]$$



Introduce New Variables

- $b_i = 1$ if $x \in [B_{i-1}, B_i]$
- $w_i = x$ if $x \in [B_{i-1}, B_i]$
- $y \approx f(x)$

$$x = \sum_{i=1}^{n} w_i$$

$$y = \sum_{i=1}^{n} (m_i w_i + c_i b_i)$$

$$B_{i-1} b_i \leq w_i \leq B_i b_i, \forall i$$

$$1 = \sum_{i=1}^{n} b_i$$

But Wait, There's More

- In many applications there is an additional binary indicator variable
- If $z = 0$, then $x = 0$, and we would like the relationship $y = f(x)$ "turned off"
- Specifically (assuming WLOG $f(0) = 0$)

$$z = 0 \implies x = 0, y = f(x) = 0$$

- The standard way is to introduce a "big-M" constraint:

$$B_0 z \leq x \leq B_n z$$

A Simple Trick

- Instead of modeling the piecewise-linear indicator in the standard way:

$$B_0 z \leq x \leq B_n z,\ 1 = \sum_{i=1}^{n} b_i$$

- Model it as

$$z = \sum_{i=1}^{n} b_i$$

- Resulting formulation is provably stronger (locally-ideal)
- All piecewise-linear functions turned on/off by an indicator have a similar modeling trick, see Sridhar et al. [2013]

Multiple Choice Model

Not "Locally ideal", note $c_1 = 0$

$$x = \sum_{i=1}^{n} w_i$$
$$y = \sum_{i=1}^{n} (m_i w_i + c_i b_i)$$
$$B_{i-1} b_i \leq w_i \leq B_i b_i, \forall i$$
$$B_0 z \leq x \leq B_n z$$
$$1 = \sum_{i=1}^{n} b_i$$

"Locally ideal"

$$x = \sum_{i=1}^{n} w_i$$
$$y = \sum_{i=1}^{n} (m_i w_i + c_i b_i)$$
$$B_{i-1} b_i \leq w_i \leq B_i b_i, \forall i$$
$$z = \sum_{i=1}^{n} b_i$$

- A formulation is locally ideal if every extreme point of the LP relaxation satisfies the discrete requirements

Example

- S: Set of suppliers
- B: Set of Cost Breakpoints
- c_{si}: Per Units cost of item from supplier $s \in S$ in cost region $i \in B$
- v_{si}: Maximum number of item from supplier $s \in S$ to purchase in region $i \in B$
- R: Number of purchase
- α_s: Maximum percentage to purchase from any supplier

Example:

```
table CBR(SUPPL,BREAK0) 'Total cost at break points'
            0           1           2           3
1     30.0000    950.0000   1850.0000   7450.0000
2      8.0000    458.0000   2158.0000  16683.0000
3     10.0000   1110.0000   2810.0000  30560.0000
;

table  BR(SUPPL,BREAK0) 'Breakpoints (quantities at which unit cost changes)'
$ondelim
BR 0 1   2   3
1  0 100 200 1000
2  0 50  250 2000
3  0 100 300 4000
$offdelim
;
```

Compute c_{si} which is cost intercept of ith segment and m_{si} which is slope of ith segment.

MCM Model: Suppliers s

$$\min \sum_{s \in S} \sum_{i \in B} c_{si} b_{si} + m_{si} w_{si}$$

$$\sum_{i \in B} w_{si} = x_s \quad \forall s \in S$$

$$B_{si-1} b_{si} \leq w_{si} \leq B_{si} b_{si} \quad \forall s \in S, i \in B$$

$$\sum_{i \in B} b_{si} = 1 \quad \text{or } z_s \text{ if fixed cost} \quad \forall s \in S$$

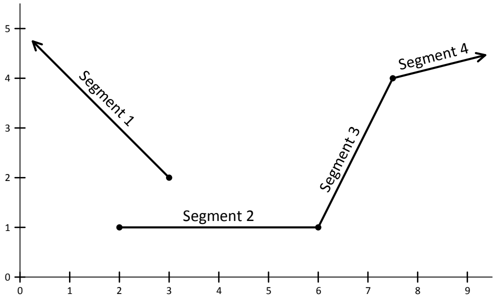
$$0 \leq x_s \leq \alpha_s R \quad \forall s \in S$$

$$\sum_{s \in S} x_s \geq R$$

$$b_{si} \text{ binary/sos1} \quad \forall s \in S$$

Infinite domains

- We explain this approximation for a function f of a single variable x.
- The piecewise-linear function is described by a collection of segments $\mathcal{S}$.
- In the case where the domain of the function is an unbounded set, or the function is not continuous, the segment approach has proven effective.
- Each segment i has an (x_i, f_i) coordinate point, a (potentially infinite) length l_i, and a slope g_i, the rate of increase or decrease of the function from (x_i, f_i).



	x	f	l	g
Seg. 1	3	2	-∞	-1
Seg. 2	2	1	4	0
Seg. 3	6	1	$\sqrt{11.25}$	2
Seg. 4	7.5	4	∞	¼

An example of a (multi-valued) piecewise-linear function described by segments.

The sign of the l_i determines if the segment expands to the left (negative length) or the right (positive length) of the (x_i, f_i) point. These segment definitions allow more than pure piecewise-linear functions. Segments can overlap, meaning we can have multi-valued functions, and there can be holes in the x coordinate space. There is also no order requirement of the segment x_i coordinates.

Each segment has two variables associated with it. The first is a binary variable b_i that chooses the segment to be used. In order that we have a single value for the function at x, only one segment can be active, which is modeled using:

$$\sum_{i \in \mathcal{S}} b_i = 1.$$

The other segment variable is a nonnegative variable λ_i whose upper bound is the absolute value of the length of the segment: $\lambda_i \leq |l_i|$. This variable measures how far we move into segment i from the starting point (x_i, f_i). A particular choice of the vectors b and λ formed from these components determines a point of evaluation $x \in \mathbb{R}$ and the value of the approximation f at x by the following formulae ($\mathrm{sgn}(l_i)$ denotes the "sign" of the parameter l_i):

$$x = \sum_{i \in \mathcal{S}} (x_i b_i + \mathrm{sgn}(l_i)\lambda_i), \quad f = \sum_{i \in \mathcal{S}} (f_i b_i + \mathrm{sgn}(l_i) g_i \lambda_i).$$

For each segment that has finite length $|l_i| < \infty$, we enforce the constraint that $\lambda_i > 0$ implies $b_i = 1$ using the "Turn on indicator":

$$\lambda_i \leq |l_i| \, b_i.$$

If the piecewise-linear function contains segments of infinite length, this constraint does not work. Instead, for these segments, we form a `SOS1` set containing the variables λ_i and $1 - b_i$, that is at most one of these two variables is positive. This has the same effect as the M constraint, but is independent of the length of the segment and hence also works with infinite length.

$$(\lambda_i, 1 - b_i) \in SOS1$$

implies that if $\lambda_i > 0$, then $1 - b_i = 0$ and hence $b_i = 1$.

How to use pwlfunc.inc

Example `purchase-pwl.gms`

- Generate parameter to specify segments: note that length is measured along the "x" axis

```
set sl  segment labels / x, y coordinates, l length, g slo
parameter costp(s,b,sl) cost segment definition;
costp(s,b,'x') = BR(s,b);
costp(s,b,'y') = CBR(s,b);
costp(s,b,'l')$BREAK1(b) = BR(s,b-1) - BR(s,b);
costp(s,b,'g')$BREAK1(b) = (CBR(s,b-1) - CBR(s,b))/costp(s
```

- Add `$batinclude pwlfunc.inc costp b x y l g s`
- Check function evaluator at various points

```
parameter fx(s); fx(s) = costp_FUNC(50,s); display fx;
```

Changes to existing model

- Add tie to existing problem variables x to define binaries and λ_i
  `x(s) =e= costp_x(s);`
- Add tie to function values for evaluation within a normal variable
  `MinCost =e= sum(s, costp_y(s));`
- Add equation list into model definition
  `model m / defobj, defx, ..., %costp_EquList% /;`
- Use mip (or minlp) since model is nonlinear
- Can use the fixed cost trick as well using pwlfuncF.inc instead and setting `pwlfcostvar` to the fixed cost variable name
- Another example is provided as `gamslib trnspwlx`

Gams models

- `purchase-multi.gms` for multi choice model (options –fcost=1 to turn on binary (not sos1) y variables)
- Model using segments: `purchase-seg.gms` (options –fcost=1 to turn on binary y variables, –size=big for larger instance, mip=gurobi)
- Model using segments: `purchase-pwl.gms` (code shows how to use pwlfuncF to do fixed cost as well).
- Have versions for sos1 and sos2 (convex combination) and incremental model, but these do not perform as well.

Also a transport example notebook on canvas.

Modeling Piecewise-Linear Functions

- There are many ways to model piecewise linear functions using integer variables. I really recommend Vielma et al. [2010], Vielma [2015] as references

Take Your Pick

- Multiple Choice Model [Jeroslow and Lowe, 1984]
- SOS2 Model [Beale and Tomlin, 1970, Beale and Forrest, 1976]
- Incremental Model [Markowitz and Manne, 1957]
- Convex Combination Model [Dantzig, 1960, Padberg, 2000]
- Disaggregated Convex Combination Model [Meyer, 1976]
- Logarithmic Model [Vielma et al., 2010]

Separable Programming

- We are going to deal with functions like

$$f(x) = \sum_i f_i(x_i),$$

  where each f_i is a function of a single variable
- Quite often these tricks are used to model a “piecewise-linear” structure
- These are typically modeled with SOS variables: $\cdots - - - \cdots$

If the problem is not separable, there are a number of tricks that can be used to substitute out non-separable terms and convert the model into a separable one. For example, we can deal with terms like $x_i x_j$ by using the fact that

$$4x_i x_j = (x_i + x_j)^2 - (x_i - x_j)^2$$

or terms like

$$\prod_{i=1}^{m} x_i$$

(with $x_i > 0$) can be replaced with y where

$$\ln y = \sum_{i=1}^{m} \ln x_i.$$

Note that linear functions are separable, so functions like $f(\sum_j a_j x_j)$ can be reformulated in a separable manner using $f(y)$ where $y = \sum_j a_j x_j$. The GAMS solver ANTIGONE does some of these reformulations automatically.