Linearization in Mathematical Programming

22 May, 2012 math blog

Linear programming problem has the form of

$$\begin{aligned} \text{minimize} \quad & \sum_{j\in J} c_j x_j \\ \text{subject to} \quad & \sum_{j\in J} a_{ij}x_j > b_i \quad \forall i\in I \\ & x_j \ge 0 \quad \forall j \in J \end{aligned}$$

If the formulation contains anything nonlinear (strictly speaking, non-convex), it cannot be solved easily in general. This is helpful to write the problem in a modelling language like AMPL and feed into a LP solver.

Minima and Maxima

If a minimum or maximum function is involved, e.g.

$$\max_{i\in I} x_i$$

it can be linearized by introducing an additional variable $x_{\max}$ and the constraints

$$x_{\max} \ge x_i \quad \forall i \in I$$

Minimax objective

Minimizing the objective function of $z = \max_k \sum_j f_k(x_j)$ can be transformed into minimizing the decision variable z with extra constraints defining z:

$$\sum_j f_k(x_j) \le z \quad \forall k$$

Absolute values

In case any absolute value of a free variable (i.e. $\in \mathbb{R}$) is involved, say $|x|$, we may convert it by introducing additional variables x^+, x^-, and additional constraints:

$$\begin{aligned} x &= x^+ - x^- \\ |x| &= x^+ + x^- \\ x^+, x^- &\ge 0 \end{aligned}$$

If we minimize the *sum* of x^+ and x^- in objective function, one of them will be equal to zero. This will not work for maximization problem or problem without objective functions. For the latter, simple minimize for the sum of non-negative parts converted. For the former, switch the problem into minimization of negative of objective function.

Fractions in objective function

This is a method introduced by A. Charnes and W.W. Cooper in their paper, *Programming with Linear Fractional Functional*, Naval Research Logistics Quaterly, Vol.9 pp.181–186, 1962.

Consider the case that a faction of polynomials is at the objective, e.g.

$$\begin{aligned} \text{minimize} \quad & \frac{a_0 + \sum_{i\in I} a_i x_i}{b_0 + \sum_{i\in I} b_i x_i} \\ \text{subject to} \quad & \sum_{i\in I} c_{ij} x_i > d_j \quad \forall j\in J \\ & x_i \ge 0 \quad \forall i\in I \end{aligned}$$

We define

$$t = \frac{1}{b_0 + \sum_{i\in I} b_i x_i}$$

and change the problem into:

$$\begin{aligned} \text{minimize} \quad & a_0 t + \sum_{i\in I} a_i x_i t \\ \text{subject to} \quad & \sum_{i\in I} c_{ij} x_i t > d_j t \quad \forall j \in J \\ & b_0 t + \sum_{i\in I} b_i x_i t = 1 \\ & t \ge 0 \\ & x_i \ge 0 \quad \forall i \in I \end{aligned}$$

In other words, we linearize the objective function by substituting for t, and convert all existing constraints by multiplying them with t. Finally, we introduce the additional constraint that t multiply by the denominator of the original fraction shall equal to one.

In such a converted problem, the decision variables x_j are transformed into $y_j = x_j t$, together with the newly introduced variable t, we can recover x_j after the solution is found.

Either-or constraints

Either $f(x) \le 0$ or $g(x) \le 0$ but not necessarily both as constraints: We can find some fairly large bounds M_i such that we are quite sure $f(x) < M_1$ and $g(x) < M_2$ in the domain we concerned. Then set up:

$$\begin{aligned} f(x) &\le M_1 b \\ g(x) &\le M_2(1-b) \end{aligned}$$

for an extra binary variable b. Similar binary variable trick can be applied to objective functions that takes different value based on some conditions. If the two constraints are mututally exclusive, however, we should add

$$\begin{aligned} f(x) &\ge -M_3(1-b) \\ g(x) &\ge -M_4 b. \end{aligned}$$

The bounds with binary variable trick can help imposing constraints like both-or-nothing as well. Below is an example of imposing exactly m out of N constraints active:

$$\begin{aligned} A_k x - M_k b_k \le c_k \quad k = 1, \dots, N \\ \sum_k b_k = m \end{aligned}$$

Multiplication

Multiplication in general cannot be linearized. However, if one of the two multiplicands is range-bounded, their product can be converted into linear form.

Let's introduce for the product $x_i y_i$:

$$\begin{aligned} u &= \frac{x_i - y_i}{2} \\ v &= \frac{x_i + y_i}{2} \end{aligned}$$

and notice that

$$\begin{aligned} v^2 - u^2 &= \frac{1}{4}(x_i^2 + 2x_i y_i + y_i^2) - \frac{1}{4}(x_i^2 - 2x_i y_i + y_i^2) \\ &= \frac{1}{4}(4x_i y_i) \\ &= x_i y_i. \end{aligned}$$

Then we can convert a multiplication into difference of variables u^2 and v^2, on the condition that their introduction can make the system remain linear.

Alternatively, consider $x_i y_i$, which y_i is a free variable (or lower-bounded only) but $m_i \le x_i \le M_i$. Then we have

$$\begin{aligned} m_i y_i \le x_i y_i \le M_i y_i \quad \forall y_i \ge 0 \\ M_i y_i \le x_i y_i \le m_i y_i \quad \forall y_i \le 0 \end{aligned}$$

or, by realizing that $\frac{1}{2}(m_i + M_i) - \frac{1}{2}(M_i - m_i) \le x_i \le \frac{1}{2}(m_i + M_i) - \frac{1}{2}(M_i - m_i)$,

$$\begin{aligned} x_i y_i &\le \frac{1}{2}(m_i + M_i) y_i + \frac{1}{2}(M_i - m_i)|y_i| \quad \forall y_i \\ x_i y_i &\ge \frac{1}{2}(m_i + M_i) y_i - \frac{1}{2}(M_i - m_i)|y_i| \quad \forall y_i \end{aligned}$$

Thus we can replace the product $x_i y_i$ by using the RHS of the above inequalities, optionally with some modification to the corresponding constraint to reflect the substitution is not exact.

In the particular case of binary variables, say, the product $\delta_1\delta_2$, it can be replaced by a new binary variable x such that:

$$\begin{aligned} & x \le \delta_1 \\ & x \le \delta_2 \\ & x \ge \delta_1 + \delta_2 - 1 \\ & x \quad \text{binary} \end{aligned}$$

In another case of a binary variable δ multiply by a bounded real number $m \le x \le M$, the product δx can be replaced by a new variable x' such that:

$$\begin{aligned} & x' \le M\delta \\ & x' \le x \\ & x' \ge x - M(1-\delta) \\ & x' \ge \min(0, m) \end{aligned}$$

For other cases, such as multiplication of two unbounded real number, we may need approximation techniques, i.e. approximate a convex curve by piecewise linear functions.

Discontinuity, Indicator Variables, and Upper-Bounds (MILP)

Let the range of a decision variable x_i to be either $a \le x \le b$ or $c \le x \le d$ which $b < c$. We may introduce an indicator variable δ to represent this:

$$\begin{aligned} & x \ge a\delta + c(1-\delta) && (1) \\ & x \le b\delta + d(1-\delta) && (2) \\ & \delta \quad \text{binary} && (3) \end{aligned}$$

Another case that indicator variables can apply is a conditional equation. Let there be a quantity

$$z = \begin{cases} c & \text{if } x = 0, \\ ax + b & \text{if } x > 0. \end{cases}$$

Then we can replace the definition of z by

$$\begin{aligned} & z = ax + b\delta + c(1-\delta) \\ & x \le M\delta \\ & x \ge 0 \\ & \delta \quad \text{binary} \end{aligned}$$

for a fairly large value M to serve as the upper-bound of x.

The introduction of an upper-bound is useful to incorporate exotic constraints in the linear programming. For example, either one of two constraints must be satisfied, but not necessarily both. Say, we have the following constraints:

$$\begin{aligned} \sum_{i\in I} a_i x_i \le b \\ \sum_{i\in I} a'_i x'_i \le b' \end{aligned}$$

where at least one of them must be satisfied. Then we can introduce an upper-bound M and an indicator variable δ to invalidate one of them, i.e.

$$\begin{aligned} & \sum_{i\in I} a_i x_i \le b + M\delta \\ & \sum_{i\in I} a'_i x'_i \le b' + M(1-\delta)\delta \qquad \text{binary} \end{aligned}$$

In general, we can mandate at least k out of n such constraints must be satisfied:

$$\begin{aligned} \sum_{i\in I} a_{ji} x_i \le b_j + M\delta_j \quad & j = 1, \cdots, n \\ \sum_{j=1}^{n} \delta_j < n - k \quad & \\ \delta_j \quad \text{binary} \quad & j = 1, \cdots, n \end{aligned}$$

This provide each constraint with an independent indicator variable δ_j, which tells if it have to be relaxed. Thus we can control the least number of constraints satisfied (k) by controlling the most number of constraints *not* satisfied ($n-k$).

A variation of such either-or constraints is the all-or-nothing constraints. Consider the two constraints case above, if it is an all-or-nothing rule, it means an either-or rule for the below:

$$\begin{aligned} \sum_{i\in I} a_i x_i > b \\ \sum_{i\in I} a'_i x'_i \le b' \end{aligned}$$

Similarly, for the case of n-constraints case, the all-or-nothing rule means

$$\begin{aligned} \sum_{i\in I} a_{ji} x_i \le b_j + M\delta \quad & j = 1, \cdots, n \\ \sum_{i\in I} a_{ji} x_i > b_j - M(1-\delta) \quad & j = 1, \cdots, n \\ \delta \quad \text{binary} \quad & \end{aligned}$$

Reference

H.P. Williams, *Model building in mathematical programming*, 5th ed., John Wiley & Sons, 2013.

Last edit: 20 October, 2017