K out of *N* Constraints Must Hold

Consider the case where the overall model includes a set of N possible constraints such that only some K of these constraints *must* hold. (Assume that $K < N$.) Part of the optimization process is to choose the *combination* of K constraints that permits the objective function to reach its best possible value. The $N - K$ constraints *not* chosen are, in effect, eliminated from the problem, although feasible solutions might coincidentally still satisfy some of them.

Denote the N possible constraints by

$$\begin{aligned} f_1(x_1, x_2, \ldots, x_n) &\le d_1 \\ f_2(x_1, x_2, \ldots, x_n) &\le d_2 \\ &\vdots \\ f_N(x_1, x_2, \ldots, x_n) &\le d_N. \end{aligned}$$

Then, applying the same logic as for the preceding case, we find that an equivalent formulation of the requirement that some K of these constraints *must* hold is

$$\begin{aligned} f_1(x_1, x_2, \ldots, x_n) &\le d_1 + My_1 \\ f_2(x_1, x_2, \ldots, x_n) &\le d_2 + My_2 \\ &\vdots \\ f_N(x_1, x_2, \ldots, x_n) &\le d_N + My_N \\ \sum_{i=1}^{N} y_i &= N - K, \end{aligned}$$

and

$$y_i \text{ is binary}, \qquad \text{for } i = 1, 2, \ldots, N,$$

where M is an extremely large positive number. For each binary variable y_i ($i = 1, 2, \ldots, N$), note that $y_i = 0$ makes $My_i = 0$, which reduces the new constraint i to the original constraint i. On the other hand, $y_i = 1$ makes $(d_i + My_i)$ so large that (again assuming a bounded feasible region) the new constraint i is automatically satisfied by any solution that satisfies the other new constraints, which has the effect of eliminating the original constraint i. Therefore, because the constraints on the y_i guarantee that K of these variables will equal 0 and those remaining will equal 1, K of the original constraints will be unchanged and the other $(N - K)$ original constraints will, in effect, be eliminated. The choice of *which* K constraints should be retained is made by applying the appropriate algorithm to the overall problem so it finds an optimal solution for *all* the variables simultaneously.

Functions with N Possible Values

Consider the situation where a given function is required to take on any one of N given values. Denote this requirement by

$$f(x_1, x_2, \ldots, x_n) = d_1 \quad \text{or} \quad d_2, \ldots, \quad \text{or} \quad d_N.$$

One special case is where this function is

$$f(x_1, x_2, \ldots, x_n) = \sum_{j=1}^{n} a_j x_j,$$

as on the left-hand side of a linear programming constraint. Another special case is where $f(x_1, x_2, \ldots, x_n) = x_j$ for a given value of j, so the requirement becomes that x_j must take on any one of N given values.

The equivalent IP formulation of this requirement is the following:

$$f(x_1, x_2, \ldots, x_n) = \sum_{i=1}^{N} d_i y_i$$

$$\sum_{i=1}^{N} y_i = 1$$

and

$$y_i \text{ is binary,} \qquad \text{for } i = 1, 2, \ldots, N.$$

so this new set of constraints would replace this requirement in the statement of the overall problem. This set of constraints provides an *equivalent* formulation because exactly one y_i must equal 1 and the others must equal 0, so exactly one d_i is being chosen as the value of the function. In this case, there are N yes-or-no questions being asked, namely, should d_i be the value chosen ($i = 1, 2, \ldots, N$)? Because the y_i respectively represent these *yes-or-no decisions,* the second constraint makes them *mutually exclusive alternatives.*