Manual No. 002

Business Practices Manual

Energy and Operating Reserve Markets

Attachment A

Market Optimization Techniques

Disclaimer

This document is prepared for informational purposes only to support the application of the provisions of the Tariff and the services provided thereunder. MISO may revise or terminate this document at any time at its discretion without notice. While every effort will be made by MISO to update this document and inform its users of changes as soon as practicable, it is the responsibility of the user to ensure use of the most recent version of this document in conjunction with the Tariff and other applicable documents, including, but not limited to, the applicable NERC Standards. Nothing in this document shall be interpreted to contradict, amend, or supersede the Tariff. MISO is not responsible for any reliance on this document by others, or for any errors or omissions or misleading information contained herein. In the event of a conflict between this document, including any definitions, and either the Tariff, NERC Standards, or NERC Glossary, the Tariff, NERC Standards, or NERC Glossary shall prevail. In the event of a conflict between the Tariff and the NERC Standards, or NERC Glossary, the Tariff shall prevail until or unless the Federal Energy Regulatory Commission orders otherwise. Any perceived conflicts or questions should be directed to the Legal Department.

TABLE OF CONTENTS

List of Exhibits:

1. Purpose of this Attachment A

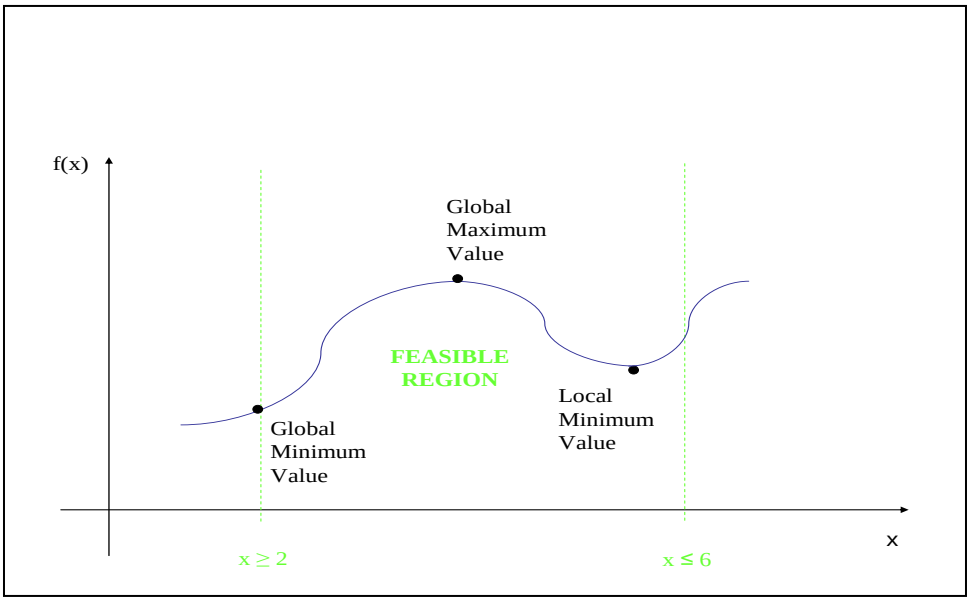
The goal of the MISO is to commit Resources and clear the Energy and Operating Reserve Markets in an optimal manner to produce the lowest cost power, subject to physical limitations, reliability requirements and good utility practice. To achieve this goal, the MISO uses simultaneously co-optimized Security Constrained Unit Commitment ("SCUC") and Security Constrained Economic Dispatch ("SCED") algorithms. To understand how these algorithms perform their optimization function, it is useful to understand some basic optimization concepts. This Attachment A to the *BPM for Energy and Operating Reserve Markets* outlines some basic optimization concepts and provides a high level description of the SCED and SCUC algorithms that are utilized by the MISO to achieve its objectives.

2. Basic Optimization Theory

2.1 The Optimization Problem

An optimization problem is a mathematical problem where the goal is to maximize or minimize the value of some mathematical function of one or more variables subject to one or more constraints on those variables. The simplest example of an optimization problem is the single variable problem illustrated in Exhibit 2-1 below:

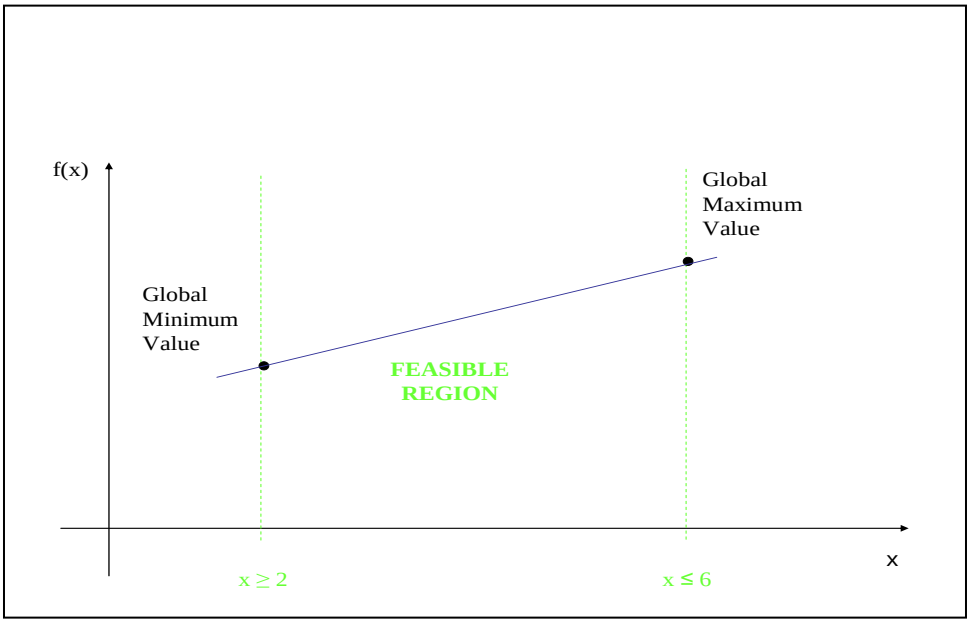
Exhibit 2-1: Single Variable Example



In Exhibit 2-1, the curve represents the mathematical function f(x) to be optimized and the vertical dotted lines are constraints restricting the value of x to be not greater than 6 nor less than 2. The mathematical function f(x) to be optimized is referred to as the objective function. The allowable region defined by the constraints is referred to as the feasible region. There is a global maximum value, a global minimum value, and a local minimum value. The global maximum value represents the maximum value of the objective function within the feasible region and the global minimum value represents the minimum value of the objective function within the feasible region. The local minimum value does not represent the minimum value of the objective function within the feasible region. Given the objective function f(x) is a continuous and differentiable function within the feasible region, maximum and minimum values (global and local) generally occur at points where the derivative of the objective function is zero or at the points where the objective function intersects the constraints (boundary points).

Consider the single variable linear objective function shown in Exhibit 2-2.

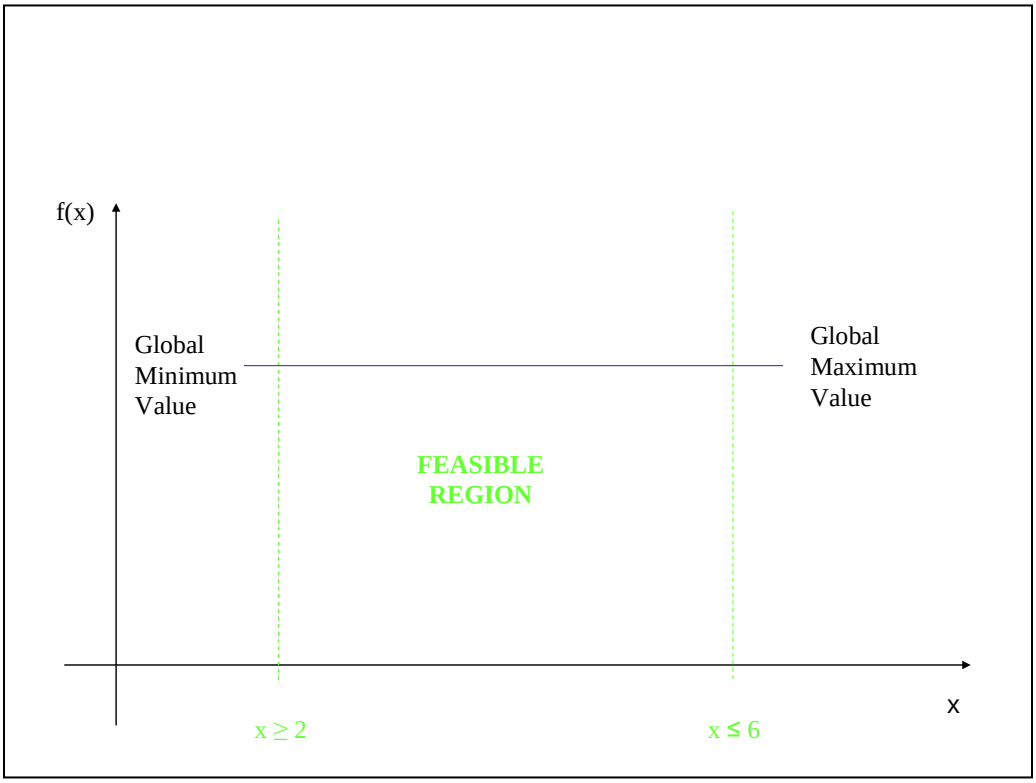
Exhibit 2-2: Linear Objective Function – Example 1



In the special case of a linear objective function, there are two important observations that can be made. First of all, there is no local minimum or maximum value. Secondly, the global minimum and maximum values lie at the boundary of the feasible region. These important features of a linear optimization problem can be used to solve large linear optimization problems within a reasonable time frame.

A special case of a single variable linear optimization problem is illustrated in Exhibit 2-3:

Exhibit 2-3: Linear Objective Function – Example 2



In Exhibit 2-3, the linear objective function represents a line with zero slope. Therefore, all points on the line represent the maximum and minimum value of the objective function. That is, there are an infinite number of optimum solutions. That is not to say there are local maxima and local minima, since local maxima and minima are generally different values than global maxima and minima. Instead, there are simply an infinite number of global maxima and minima, all of which are the same value and any of which represent an optimum solution.

2.2 Multi-Variable Optimization Problems

Most optimization problems in the real world include more than one variable. The MISO SCUC and SCED problems generally contain well over 100,000 variables. A simple example of a multi-variable optimization problem is shown in Exhibit 2-4 below:

Exhibit 2-4: Multi-Variable Objective - Example 1

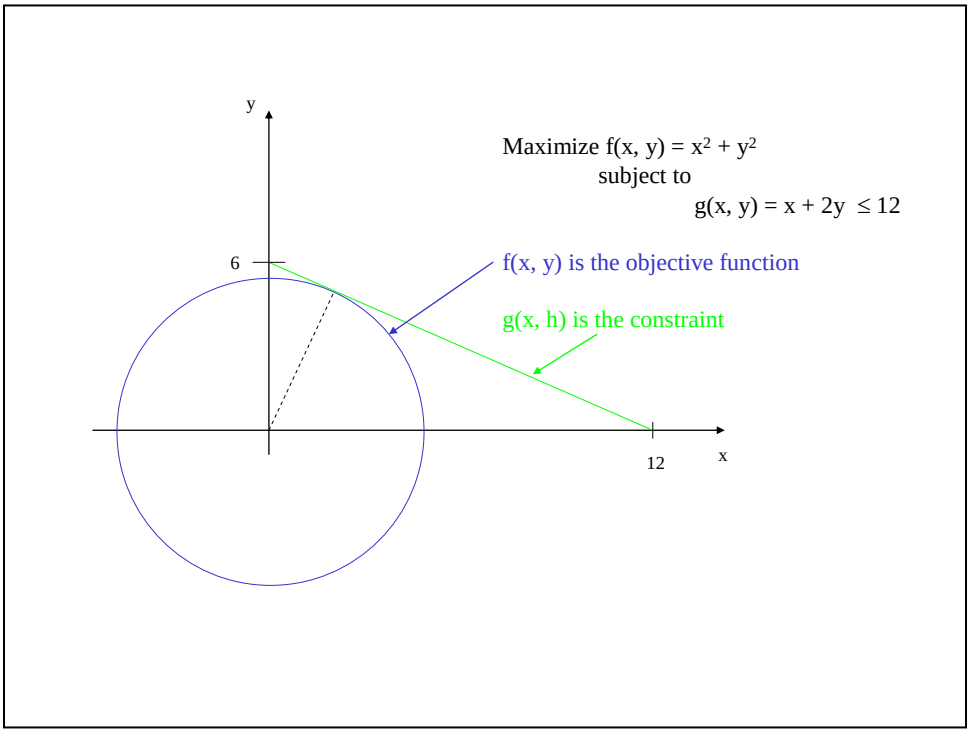


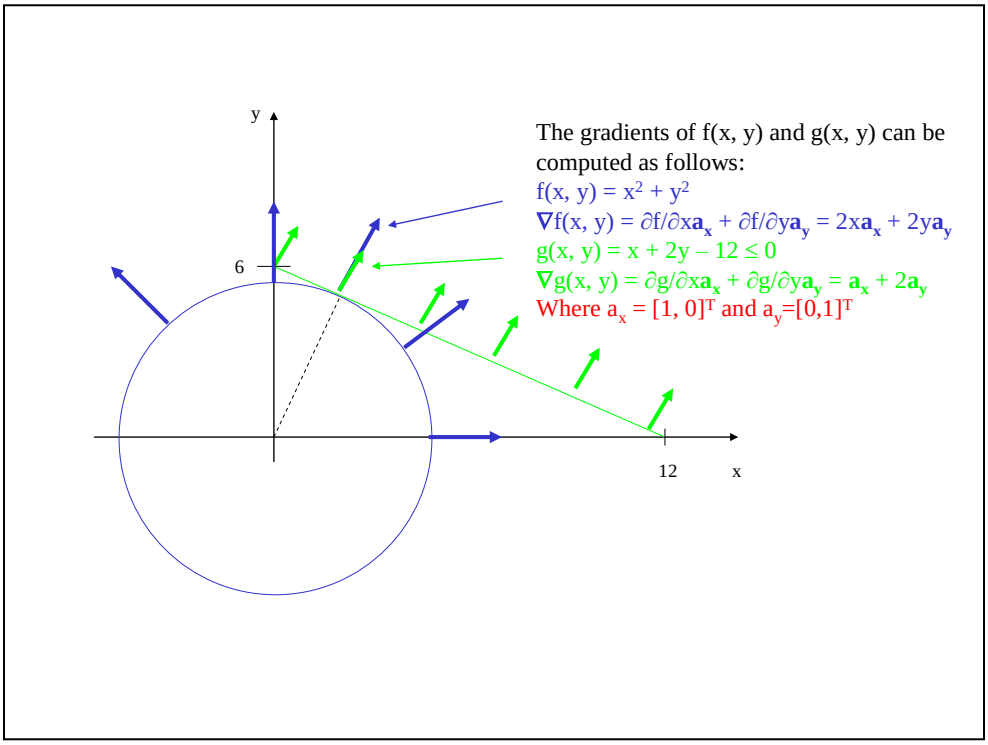
Exhibit 2-4 contains a non-linear objective function (the circle) of variables x and y subject to a linear constraint (the line) that is a linear function of variables x and y. It is important to note that the objective function is actually a three-dimensional surface (elliptic paraboloid) and the constraint is actually a plane in three-dimensional space. In general, linear objective functions and constraints of two variables can be represented by planes in three-dimensional space whereas non-linear objective functions and constraints of two variables can be represented by three-dimensional surfaces.

The objective above is to maximize the square of the radius of the circle (or the height of the elliptic paraboloid measured along the z axis out of the page), subject to the constraint that the circle not cross the line (or elliptic paraboloid not cut through the plane). Exhibit 2-4 above

illustrates the maximum value of the objective function as the point where the circle is tangent to the line (or the elliptic paraboloid just touches the plane).

In Exhibit 2-5 below, the gradients of the objective function and constraint, which represent the magnitude and direction of the largest change in the objective function and constraint respectively for an incremental increase in each of the variables x and y, are calculated and illustrated:

Exhibit 2-5: Multi-Variable Optimization - Example 2



In Exhibit 2-5 above, the gradients are parallel at the point where the objective function reaches its maximum value subject to the constraint. A rigorous mathematical proof (that is beyond the scope of this document) will show that when the maximum or minimum value of an objective function subject to a single constraint is limited by that constraint, the maximum or minimum value of the objective function will be the solution where the objective function is tangent to the constraint (i.e., pushes up against the constraint), which is the point where the gradient of the objective function is parallel to the gradient of the constraint. That is, the maximum or minimum value of an objective function is on the boundary of the feasible region.

2.2.1 Lagrange Multipliers

Given that the gradient of the objective function and the gradient of the constraint are parallel at the maximum or minimum value of the objective function when the maximum or minimum value is limited by the constraint, the maximum or minimum value of the objective function f(x, y) subject to a single constraint g(x, y) can be expressed mathematically as follows:

$$\nabla f(x, y) = \lambda * \nabla g(x, y) \quad \text{(A-1)}$$

where λ is a scalar since the gradients are parallel

The scalar λ is known as a Lagrange multiplier. The mathematical expression above implies the following:

$$\partial f/\partial x = \lambda * \partial g/\partial x \quad \text{(A-2)}$$

$$\partial f/\partial y = \lambda * \partial g/\partial y \quad \text{(A-3)}$$

In the simple example above, solving for the optimum values of x and y in terms of λ is trivial,

$$\partial f/\partial x = \lambda * \partial g/\partial x \Rightarrow 2x = \lambda \Rightarrow x = \lambda/2 \quad \text{(A-4)}$$

$$\partial f/\partial y = \lambda * \partial g/\partial y \Rightarrow 2y = 2\lambda \Rightarrow y = \lambda \quad \text{(A-5)}$$

Therefore,

$$x = y/2 \text{ or } 2x - y = 0 \quad \text{(A-6)}$$

In this simple example, the optimum values of x and y are found by simultaneously solving the following two equations:

$$2x - y = 0 \quad \text{(determined using } \lambda\text{)} \quad \text{(A-7)}$$

$$x + 2y = 12 \quad \text{(constraint g)} \quad \text{(A-8)}$$

The solution is as follows:

x = 2.4 (A-9)

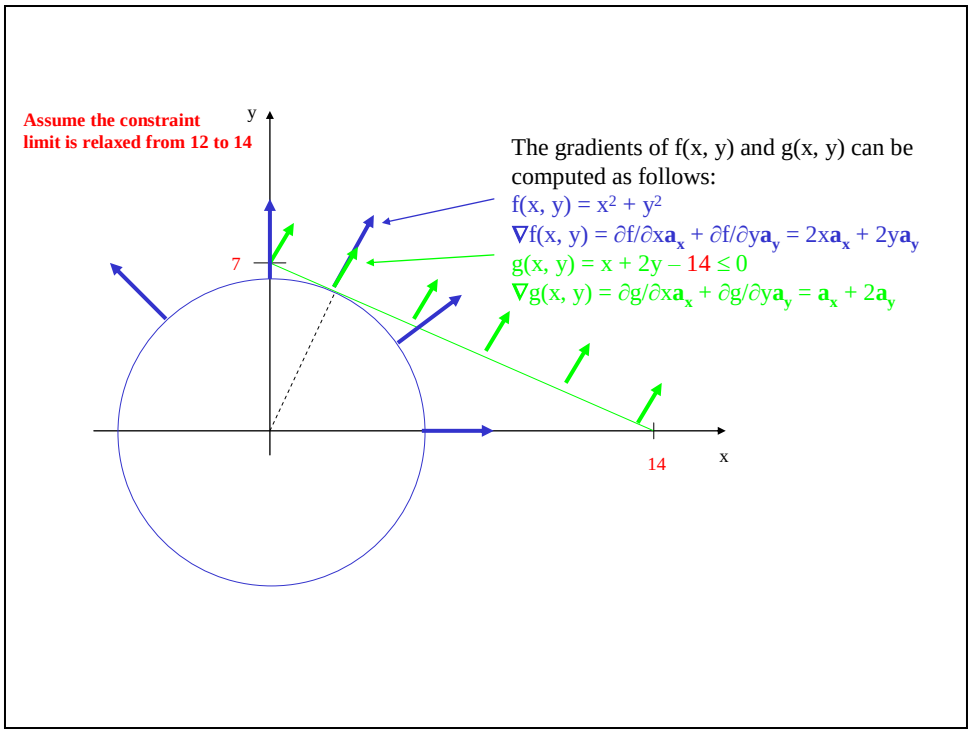
y = 4.8 (A-10)

λ = 4.8 (A-11)

f(x, y) = 28.8 (A-12)

Now assume the constraint limit in the example above is relaxed from 12 to 14 as illustrated in Exhibit 2-6 below:

Exhibit 2-6: Multi-Variable Optimization – Example 3



With the relaxed constraint limit of 14, the optimum values of x and y are found by simultaneously solving the following two equations:

$2x - y = 0$ (determined using λ - same as before) (A-13)

$x + 2y = 14$ (revised constraint g) (A-14)

The solution is as follows:

$x = 2.8$ (A-15)

$y = 5.6$ (A-16)

$\lambda = 5.6$ (A-17)

$f(x, y) = 39.2$ (A-18)

The following comparison of the results of the simple optimization problem with a constraint limit of 12 and 14 provides significant insight.

Constraint Limit	12	14
Lagrange Multiplier λ	$\lambda_{12} = 4.8$	$\lambda_{14} = 5.6$
Objective Function f(x, h)	$f_{12}(x, y) = 28.8$	$f_{14}(x, y) = 39.2$

The difference in the objective function for the two constraints limits can be calculated as follows:

$\Delta f(x, y) = f_{14}(x, y) - f_{12}(x, y) = 39.2 - 28.8 = 10.4$ (A-19)

The average value of the Lagrange Multiplier can be calculated as follows:

$\text{Average } \lambda = [\lambda_{12} + \lambda_{14}] / 2 = [4.8 + 5.6] / 2 = 5.2$ (A-20)

If the change in the objective function is divided by the change in the constraint limit, the following result is obtained:

$$\Delta f(x, y) / \Delta \text{Limit} = 10.4 / 2 = 5.2 \qquad \text{(A-21)}$$

For a relaxation of the constraint limit, the average value of the Lagrange Multiplier for the two constraint limits is found to be equal to the change in the objective function divided by the change in the constraint limit, or

$$\text{Average } \lambda = \Delta f(x, y) / \Delta \text{Limit} = 5.2 \qquad \text{(A-22)}$$

The following observation can be made:

As $\Delta \text{Limit} \rightarrow 0$,

$\Delta f(x, y) \rightarrow 0$,

λ_{12} and $\lambda_{14} \rightarrow \lambda = df(x, y) / d\text{Limit}$ at the optimum point (A-23)

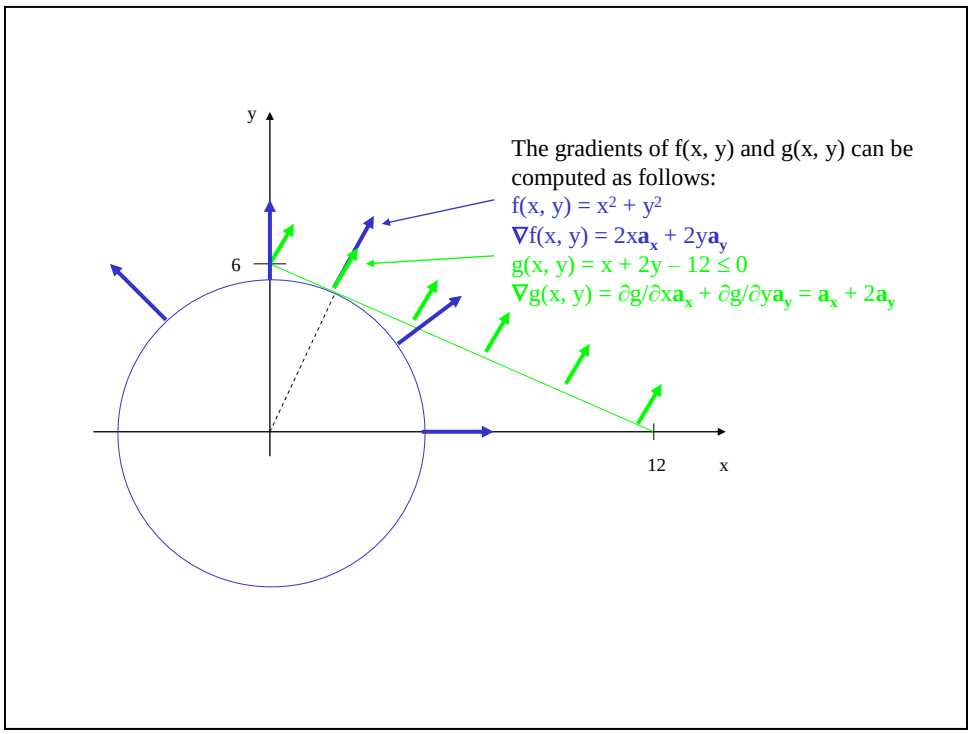
Based on the observation above, the Lagrange multiplier corresponds to the incremental change in the objective function with respect to an incremental change in the constraint limit at the optimum point, which is the incremental cost to bind the constraint at the optimal point (assuming the objective function is a cost function), better known as the constraint Shadow Price.

In general the Lagrange multiplier of a constraint is the same thing as the constraint Shadow Price when the objective function is a cost function.

2.2.2 Binding vs. Non-Binding Constraints

In general, if the constraint limits the maximum or minimum value of an objective function, the constraint is said to be binding. If the constraint does not limit the maximum or minimum value of an objective function, the constraints is said to be non-binding. To illustrate this concept, refer to Exhibit 2-7 below:

Exhibit 2-7: Multi-Variable Optimization – Binding Constraint



Per equations A-9 and A-10, the values of x and y that maximize f(x, y) subject to g(x, y) were found to be 2.4 and 4.8 respectively.

This is an example of a binding constraint with a non-zero Shadow Price (or Lagrange Multiplier) of $\lambda = 4.8$ per equation A-11. Now consider the problem of finding the minimum value of the objective function. The minimum value is represented by the base or point of the elliptic paraboloid which is located where x and y both equal zero. The gradient of f(x, y) and g(x, y) at the minimum point can be calculated as follows:

$$\nabla f(x, y) = 2x\mathbf{a}_x + 2y\mathbf{a}_y = 0 \qquad \text{(A-24)}$$

$$\nabla g(x, y) = \mathbf{a}_x + 2\mathbf{a}_y \qquad \text{(A-25)}$$

The only value of λ that will satisfy equation A-1 at this point is zero since $\nabla f(x, y) = 0$ and $\nabla g(x, y) \neq 0$. This is an example of a minimum value that is not on the boundary of the feasible region. This is not to imply that the Shadow Price of a constraint is only zero when the solution is not on the boundary of the feasible region. This is only true in the special case of a single constraint. In general, the Shadow Price of a constraint is only zero when the constraint is not binding, which means the optimum value of the objective function does not intersect (or push up against) the particular constraint.

2.2.3 Lagrangian

If equation A-1 is slightly rearranged, the following equation results:

$$\nabla f(x, y) = \lambda * \nabla g(x, y)$$

$$\Rightarrow$$

$$\nabla f(x, y) - \lambda * \nabla g(x, y) = 0 \qquad \text{(A-26)}$$

Assume the Lagrangian of a single-variable optimization problem is defined as follows:

$$L(x, y) = f(x, y) - \lambda * g(x, y) \qquad \text{(A-27)}$$

Since λ is a scalar, the optimization problem can be solved by calculating the gradient of the Lagrangian and then setting the gradient equal to zero, or

$$\nabla L(x, y) = \nabla f(x, y) - \lambda * \nabla g(x, y) = 0 \qquad \text{(A-28)}$$

This method of solving optimization problems is generally restricted to very trivial problems involving only a few variables.

2.3 Multi-Constraint Optimization Problems

Most optimization problems, including the MISO SCUC and SCED problems, contain a single objective function subject to multiple constraints, such as the following:

Maximize {f(x, y)}

subject to

$g_1(x, y) \leq C_1$
$g_2(x, y) \leq C_2$

$g_N(x, y) \leq C_N$ (A-29)

In the case of multiple constraints, there are N Lagrange multipliers (one per constraint) that satisfy the following at the maximum value of f(x, y):

$$\nabla f(x, y) = [\lambda_1 * \nabla g_1(x, y)] + [\lambda_2 * \nabla g_2(x, y)] + ... + [\lambda_N * \nabla g_N(x, y)] \quad \text{(A-30)}$$

For any constraint that is not binding at the optimum point, the Lagrange multiplier or Shadow Price is equal to zero.

For any constraint that is binding at the optimum point, the Lagrange multiplier or Shadow Price is equal to the incremental cost to bind that constraint.

The Lagrangian of an optimization problem with multiple constraints can be expressed as follows:

$$L(x, y) = f(x, y) - [\lambda_1 * g_1(x, y)] - [\lambda_2 * g_2(x, y)] - ..... - [\lambda_N * g_N(x, y)] \quad \text{(A-31)}$$

The gradient of the above Lagrangian is then calculated as follows:

$$\nabla L(x, y) = \nabla f(x, y) - [\lambda_1 * \nabla g_1(x, y)] - [\lambda_2 * \nabla g_2(x, y)] - ..... - [\lambda_N * \nabla g_N(x, y)] \quad \text{(A-32)}$$

Setting the gradient equal to zero then yields the following:

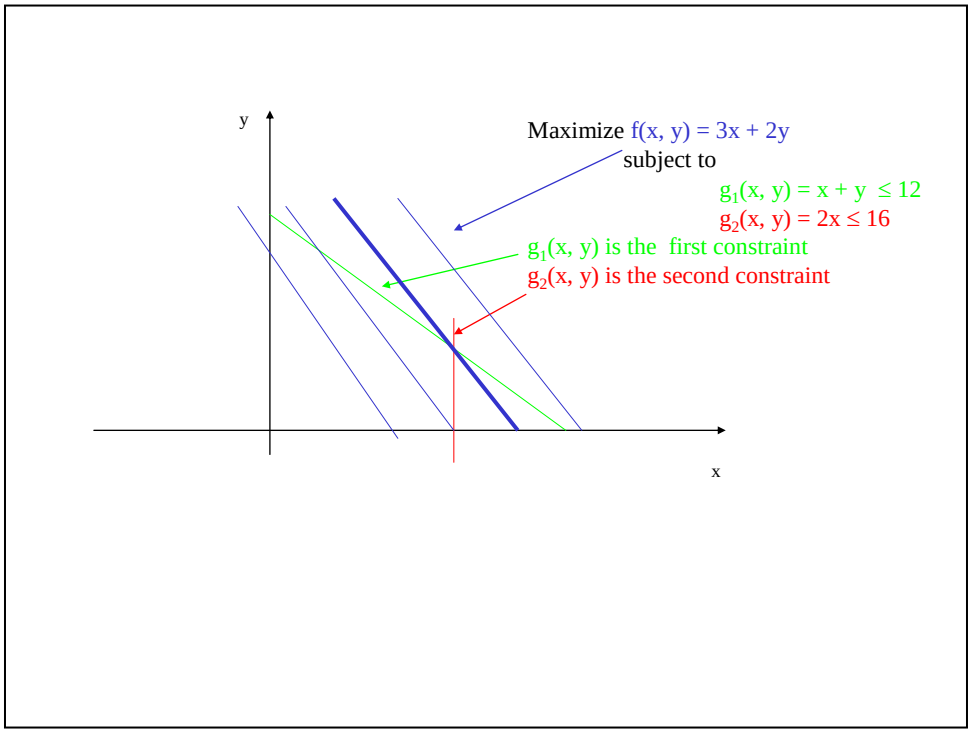
$$\nabla L(x, y) - [\lambda_1 * \nabla g_1(x, y)] - [\lambda_2 * \nabla g_2(x, y)] - ..... - [\lambda_N * \nabla g_N(x, y)] = 0$$

thus,

$$\partial f/\partial x - [\lambda_1][\partial g_1/\partial x] - [\lambda_2][\partial g_2/\partial x] - .....- [\lambda_N][\partial g_N/\partial x] = 0$$
$$\partial f/\partial y - [\lambda_1][\partial g_1/\partial y] - [\lambda_2][\partial g_2/\partial y] - .....- [\lambda_N][\partial g_N/\partial y] = 0 \quad \text{(A-33)}$$

To illustrate graphically an optimization problem with multiple constraints, consider the example in Exhibit 2-8 below that contains a linear objective function subject to two linear constraints:

Exhibit 2-8: Multi-Constraint Optimization



The gradients of g_1 and g_2 can be calculated as follows:

$$\nabla g_1(x, y) = \mathbf{a}_x + \mathbf{a}_y \quad \text{(A-34)}$$

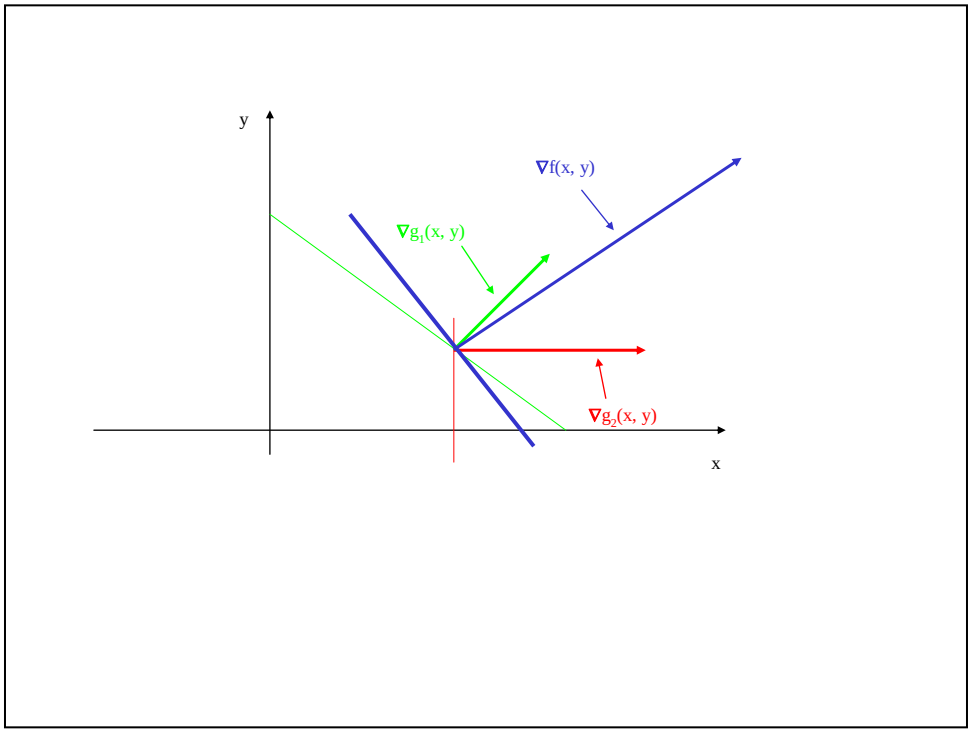
$$\nabla g_2(x, y) = 2\mathbf{a}_x \quad \text{(A-35)}$$

The gradient of f is calculated as follows:

$$\nabla f(x, y) = 3\mathbf{a}_x + 2\mathbf{a}_y \quad \text{(A-36)}$$

An illustration of these gradients is provided in Exhibit 2-9 below:

Exhibit 2-9: Multi-Constraint Optimization Gradients



The values of the Lagrange Multipliers in this trivial example can be calculated as follows:

$$\partial f/\partial x = \lambda_1 \partial g_1/\partial x + \lambda_2 \partial g_2/\partial x \quad \text{(A-37)}$$

$$\partial f/\partial y = \lambda_1 \partial g_1/\partial y + \lambda_2 \partial g_2/\partial y \quad \text{(A-38)}$$

Substituting the vector components of equations A-34, A-35 and A-36 into equations A-37 and A-38 yields the following:

$$3 = \lambda_1 + 2\lambda_2 \quad \text{(A-39)}$$

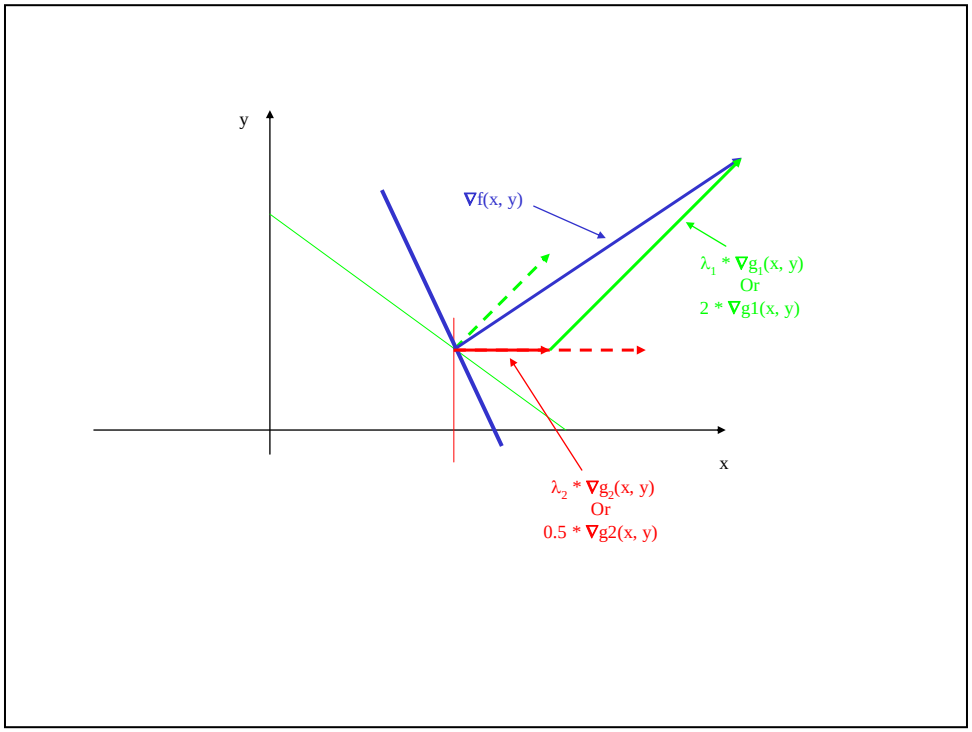
$$2 = \lambda_1 \quad \text{(A-40)}$$

Substituting equation A-40 into equation A-39 yields the following result:

$$\lambda_2 = 0.5 \quad \text{(A-41)}$$

The Lagrange multipliers λ_1 and λ_2 are illustrated in Exhibit 2-10 below:

Exhibit 2-10: Multi-Constraint Optimization Lagrange Multipliers



The validity of the results above can be checked by relaxing each of the constraints one at a time. Assuming the constraint g_1 limit is relaxed by an amount dz and the constraint g_2 limit is not, the following constraints result:

$$g_1(x, y) = x + y \leq 12 + dz \quad \text{(A-42)}$$

$$g_2(x, y) = 2x \leq 8 \quad \text{(A-43)}$$

Because $g_2(x, y)$ does not change but continues to bind, x must remain at 4.

Because $g_1(x, y)$ is relaxed by dz, y may increase by dz. Therefore,

$$f(x, y) = 3x + 2y \quad \text{(A-44)}$$

$$f(x, y + dz) = 3x + 2(y + dz) \quad \text{(A-45)}$$

$$df(x, y) = f(x, y + dz) – f(x, y) = 3x + 2y + 2dz – 3x –2y = 2dz \quad \text{(A-46)}$$

$$\lambda_1 = df(x, y)/dz = 2dz/dz = 2 \quad \text{(A-47)}$$

Assuming the constraint g_2 limit is relaxed by an amount dz and the constraint g_1 limit is not, the following constraints result:

$$g_1(x, y) = x + y \leq 12 \quad \text{(A-48)}$$

$$g_2(x, y) = 2x \leq 8 + dz \quad \text{(A-49)}$$

Because $g_2(x, y)$ is relaxed by dz, x may increase by dz/2. Because $g_1(x, y)$ does not change but continues to bind, x + y must remain at 12, implying that y must decrease by dz/2. Therefore,

$$f(x, y) = 3x + 2y \quad \text{(A-50)}$$

$$f(x + dz/2, y – dz/2) = 3(x + dz/2) + 2(y – dz/2) \quad \text{(A-51)}$$

$$df(x, y) = f(x + dz/2, y – dz/2) – f(x, y)$$
$$= 3x + 1.5dz + 2y - dz – 3x –2y = 0.5dz \quad \text{(A-52)}$$

$$\lambda_2 = df(x, y)/dz = 0.5dz/dz = 0.5 \quad \text{(A-53)}$$

3. Security Constrained Economic Dispatch ("SCED") Algorithm Overview

The MISO uses a simultaneously co-optimized Security Constrained Economic Dispatch ("SCED") algorithm to perform the following core functions.

- Clear and Price the Day-Ahead Market
- Clear the Real-Time Market
- Price the Real-Time Market

3.1 Types of SCED Algorithms

The DA-SCED algorithm is a SCED algorithm used to clear and price the Day-Ahead Energy and Operating Reserve Market and is executed sequentially for each individual hour in the Day-Ahead Energy and Operating Reserve Market subsequent to the execution of the DA-SCUC algorithm. The RT-SCED algorithm is a SCED algorithm used to clear the Real-Time Energy and Operating Reserve Market and generate real-time ex-ante prices and is executed every five minutes approximately ten minutes prior to the end of the Dispatch Interval. The Ex-Post Calculator is a SCED algorithm used to generate five minute real-time ex-post prices. The Ex-Post Calculator gets executed with the approval of the real-time dispatch.

3.2 Linear Programming ("LP") Solvers

All SCED algorithms use a Linear Programming ("LP") solver to determine cleared volumes and prices. With respect to the DA-SCED algorithm, both the cleared volumes and prices are binding. With respect to the RT-SCED algorithm, only the cleared volumes are binding. With respect to the Ex-Post Calculator, only the prices are binding.

An LP solver is a software program that solves a linear optimization problem. As a matter of fact, an LP solver actually solves two linear optimization problems, one of which is referred to as the primal problem and one of which is referred to as the dual problem. The primal problem is solved to determine cleared values for Energy, Regulating Reserve and Contingency Reserve whereas the dual problem is solved to determine Shadow Prices that are ultimately used to calculate energy Locational Marginal Prices ("LMPs") and Operating Reserve Market Clearing Prices ("MCPs"). A comparison of the results of the primal problem and dual problem provide valuable information as to the quality of the overall solution. Primal problems and dual problems will be discussed in more detail throughout this Attachment.

The standard mathematical representation of an LP solver problem is as follows:

Maximize $\mathbf{[c]}^T\mathbf{[x]}$ subject to $\mathbf{[A][x]} \leq \mathbf{[b]}$, $x \geq 0$ (A-54)

where

[c] = Vector of objective function coefficients
[x] = Vector of variables to be optimized
[A] = Matrix of constraint coefficients
[b] = Vector of constraint limits

A minimization LP problem may be converted into the standard form as follows:

Minimize $\mathbf{[c]}^T\mathbf{[x]}$ subject to $\mathbf{[A][x]} \geq \mathbf{[b]}$, $x \geq 0$ (A-55)

This problem is converted to standard form as follows:

Maximize $\mathbf{[c']}^T\mathbf{[x]}$ subject to $\mathbf{[A']][x]} \leq \mathbf{[b']}$, $x \geq 0$ (A-56)

where

[c'] = - **[c]**
[A'] = - **[A]**
[b'] = - **[b]**

The standard form of the LP problem is a maximization problem with "less than or equal to" (“LE”) constraints. A "greater than or equal to" (“GE”) constraint in the standard form can be converted to an LE constraint in the standard form by multiplying both the left-hand side (“LHS”) of the constraint and the right-hand side (“RHS”) of the constraint by -1. That is, $Ax \geq b$ is equivalent to $-Ax \leq -b$.

An equality constraint is modeled by using both a GE constraint and an LE constraint. Once in standard form, the GE constraint can be converted to an LE constraint. For example, the constraint $Ax = b$ is replaced with the constraints $Ax \leq b$ and $Ax \geq b$, or $Ax \leq b$ and $-Ax \leq -b$ after converting the GE constraint to an LE constraint.

In summary, an LP solver is a computer program that will find the set of values for **[x]** that maximize the objective function (or objective cost) $\mathbf{[c]}^T\mathbf{[x]}$ while satisfying the set of constraints **[A][x]** ≤ **[b]**.

LP Solver Example: *Consider the following objective function and constraints associated with a two-variable problem with two constraints:*

Objective Function: Maximize $\mathbf{[c]^T[x]}$ *= Maximize* $\{c_1x_1 + c_2x_2\}$

Constraint 1: $A_{11}x_1 + A_{12}x_2 \leq b_1$

Constraint 2: $A_{21}x_1 + A_{22}x_2 \leq b_2$

Assume the following values:

$c_1 = 4, c_2 = 2, A_{11} = 1, A_{12} = 0, A_{21} = 0, A_{22} = 1, b_1 = 1, b_2 = 3$

The problem is now stated as follows:

Maximize $\{4x_1 + 2x_2\}$

subject to:

$x_1 \leq 1, x_2 \leq 3$

The solution to this problem is:

$x_1 = 1, x2 = 3$, *Maximized Objective Function = 10*

The MISO SCED problems can include well over 100,000 primal variables and 100,000 constraints, thus the solution to the typical MISO SCED problem is not as trivial as the solution to the problem in this example.

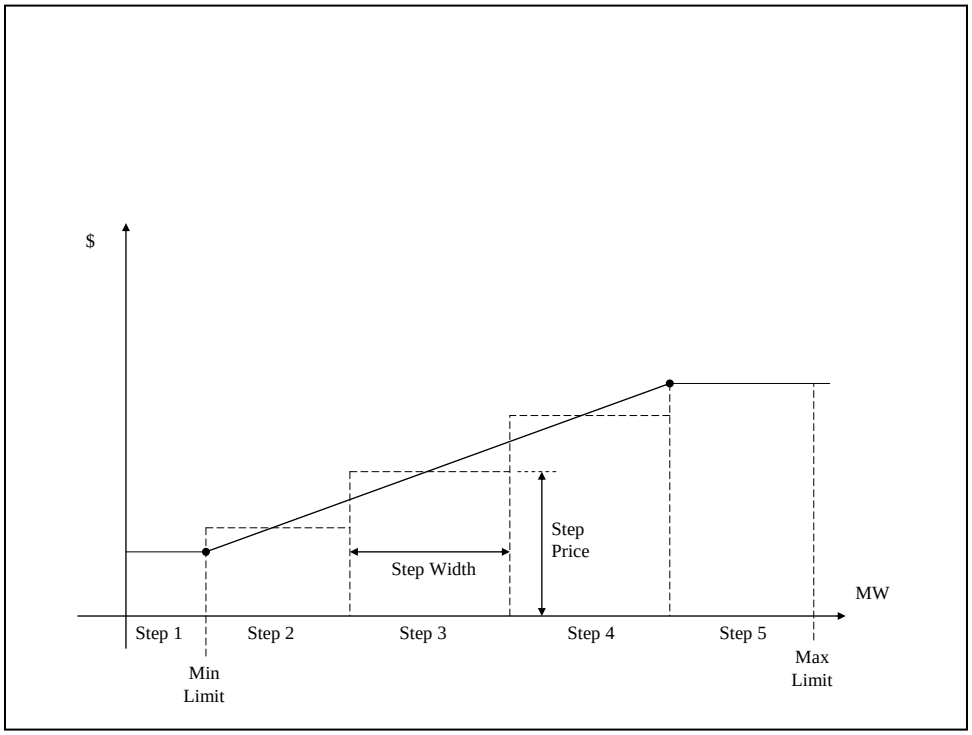
3.3 Implementing SCED with an LP Solver

The MISO SCED algorithms utilize LP solvers to determine cleared quantities for Energy, Regulating Reserve and Contingency Reserve. This is the primal problem. Therefore, the primal variables in the MISO SCED algorithms are cleared Energy, cleared Regulating Reserve and cleared Contingency Reserve for each resource.

The MISO SCED objective function is a cost minimization function that represents the total dispatch cost. Since commitment occurs prior to the execution of the SCED algorithm, commitment costs such as start-up costs, no-load costs, shut-down costs and hourly curtailment costs are not included in the SCED objective function. These costs are included in the SCUC objective function, but are considered sunk in the dispatch timeframe (i.e., commitment decisions have already been made).

The coefficients of the objective function (i.e., $\mathbf{[c]}^T$) represent the offers. For Operating Reserve products, these coefficients represent the single Operating Reserve offer value. For Energy, offers curves must be converted into stepped approximations to linearize the problem (if the offer curve is submitted as a stepped curve already, this process is unnecessary). Each step of the offer curve is dispatched independently, and is thus associated with a unique energy dispatch primal variable. The fixed offer value for the offer curve step represents the coefficient for the energy step dispatch primal variable. A maximum limit is applied to each offer curve step as appropriate based on the step width, and the sum of the energy cleared in each step, which represents the cleared energy on the resource, must be between the resource minimum and maximum limits. The use of a linearized step approximation for energy offer curves requires all offer curves to be monotonically increasing, otherwise the upper operating range of a resource could be dispatched without dispatching the lower operating range first, which is a physical impossibility. The linearization of an energy offer curve into several step energy dispatch primal variables is illustrated below in Exhibit 3-1:

Exhibit 3-1: Energy Offer Curve Primal Variable



There are many constraints in the SCED problem. The SCED problem includes equality constraints, "greater than or equal to" constraints (i.e., GE constraints) and "less than or equal to" constraints (i.e., LE constraints). Examples of equality constraints are global power balance constraints. Examples of GE constraints are Sub-Regional Power Balance Constraints, Operating Reserve requirement constraints, minimum limit constraints and ramp-down constraints. Examples of LE constraints are transmission constraints, ramp-up constraints and maximum limit constraints.

The coefficients of the constraints (i.e., **[A]**) vary depending on the type of constraint. For example, in a global power balance constraint, the coefficients for all cleared energy primal variables would be 1 whereas the coefficients for all cleared Operating Reserve primal variables would be zero. For a specific resource limit constraint, the coefficients for the cleared Energy, cleared Regulating Reserve and cleared Contingency Reserve primal variables for that resource would be 1 whereas the coefficients for the cleared Energy, cleared Regulating Reserve and cleared Contingency Reserve primal variables for all other resources would be zero. For a transmission constraint, the coefficients would be zero for Operating Reserve primal variables, but would be equal to the shift factor for each resource cleared energy primal variable.

The constraint limits ((i.e., **[b]**) will include the system Load Forecast plus interchange for a global power balance constraint, fixed Bus Load for a global power balance constraint, reserve requirements for a reserve constraint, Resource limits and ramp rates for Resource limit and ramping constraints respectively and transmission limits for transmission constraints.

3.4 LP Solution Characteristics

3.4.1 Infeasible LP Solutions

A feasible LP solution is one in which a set of values exists for the primal variables that satisfies all of the constraints. If there is no set of primal variable values that satisfy all constraints, the solution is said to be infeasible. Infeasible solutions generally result when there are two or more opposing constraints that cannot be simultaneously satisfied by a single solution. For example, consider a constraint that requires a specific primal variable to be greater than or equal to 100 and a second constraint that requires this same primal variable to be less than or equal to 50. There is no value that satisfies both of these constraints. Therefore, an LP problem that contains both of these constraints is said to be infeasible.

3.4.2 Unbounded LP Solutions

A bounded LP solution is one in which a finite maximum (or minimum if a minimization problem) solution exists. If a finite maximum (or minimum) does not exist, the problem is said to be unbounded. For example, consider a problem to maximize x subject to the constraint that x is greater than or equal to 1. There is no finite solution to this problem since there is no constraint that provides an upper bound on the value of x. If this problem were a minimization problem, it would not be unbounded and the solution would be 1.

3.4.3 Binding vs. Non-Binding Constraints

A constraint is said to be binding in an LP solution if the LP solution is feasible and the LHS of the constraint (i.e., ΣAx) would have been greater than the RHS of the constraint (i.e., b or constraint limit) in standard form had the constraint not been included in the LP problem formulation. Otherwise, the constraint is non-binding. Since a non-binding constraint does not impact the solution of an LP problem, a non-binding constraint has no impact on the objective function and the marginal cost of enforcing or satisfying a non-binding constraint is zero. Since a binding constraint does impact the solution of an LP problem, a binding constraint does impact the objective function and the marginal cost of enforcing or satisfying a binding constraint is non-zero.

3.4.4 Shadow Prices

A Shadow Price is associated with a specific constraint and basically represents the incremental cost of satisfying a constraint. One definition of a Shadow Price would be the incremental change in the objective function with respect to an incremental relaxation of a single constraint limit. For GE constraints, limits are relaxed by reducing the limit an incremental amount. For LE constraints, limits are relaxed by increasing the limit an incremental amount.

Shadow Price Example 1. *Assume the following two SCED solutions are executed:*

SCED Solution 1: Objective Cost = X
SCED Solution 1: Transmission Constraint k Limit = b
SCED Solution 2: Objective Cost = X'
SCED Solution 2: Transmission Constraint k Limit = b + db
where db is an incremental amount.
Assume SCED Problem 1 is otherwise formulated the same as SCED Problem 2. The Shadow Price of transmission constraint k would be expressed as follows:
[X' - X] / db
or

ΔObjective Cost / Incremental Limit Relaxation

If the constraint is non-binding, X' will equal X and the Shadow Price will be zero. If the constraint is binding, X' will be less than X and the Shadow Price will be negative. LE constraints such as transmission constraints have negative Shadow Prices, which basically means an incremental increase in the limit will result in a lower objective cost.

Shadow Price Example 2. *Assume the following two SCED solutions are executed:*

SCED Solution 1: Objective Cost = X
SCED Solution 1: Reserve Requirement = b
SCED Solution 2: Objective Cost = X'
SCED Solution 2: Reserve Requirement = b - db
where db is an incremental amount.

Assume SCED Problem 1 is otherwise formulated the same as SCED Problem 2. The Shadow Price of the reserve requirement constraint would be expressed as follows:

[X' - X] / -db
or
ΔObjective Cost / Incremental Limit Relaxation

If the constraint is non-binding, X' will equal X and the Shadow Price will be zero. If the constraint is binding, X' will be less than X and the Shadow Price will be positive. GE constraints such as reserve requirement constraints have positive Shadow Prices, which basically means an incremental decrease in the limit will result in a lower objective cost.

3.5 Dual Problem

While it is possible to determine Shadow Prices by executing a separate incremental LP solution for an incremental relaxation of each constraint, this method would be too time consuming and computationally intensive. Instead, Shadow Prices are determined by solving the dual problem. Every optimization problem, which is referred to as the primal problem, has a dual problem that is defined as follows based on the standard form of the primal problem:

Minimize $\mathbf{[b]}^T\mathbf{[\lambda]}$ subject to $\mathbf{[A]}^T\mathbf{[\lambda]} \geq \mathbf{[c]}$, $\lambda \geq 0$ (A-57)

where
$\mathbf{[\lambda]}$ = Vector of dual variables

Where a primal variable exists for each cleared quantity in the primal problem, a dual variable exists for each constraint in the dual problem. The dual variable associated with a specific constraint is equal to the Shadow Price of the constraint. Therefore, solving the primal problem yields the solution for the primal variables (i.e., cleared volumes) whereas solving the dual problem yields the solution for the dual variables (i.e., Shadow Prices). Shadow Prices are then used to calculate LMPs and MCPs.

3.5.1 Basic Duality Concepts

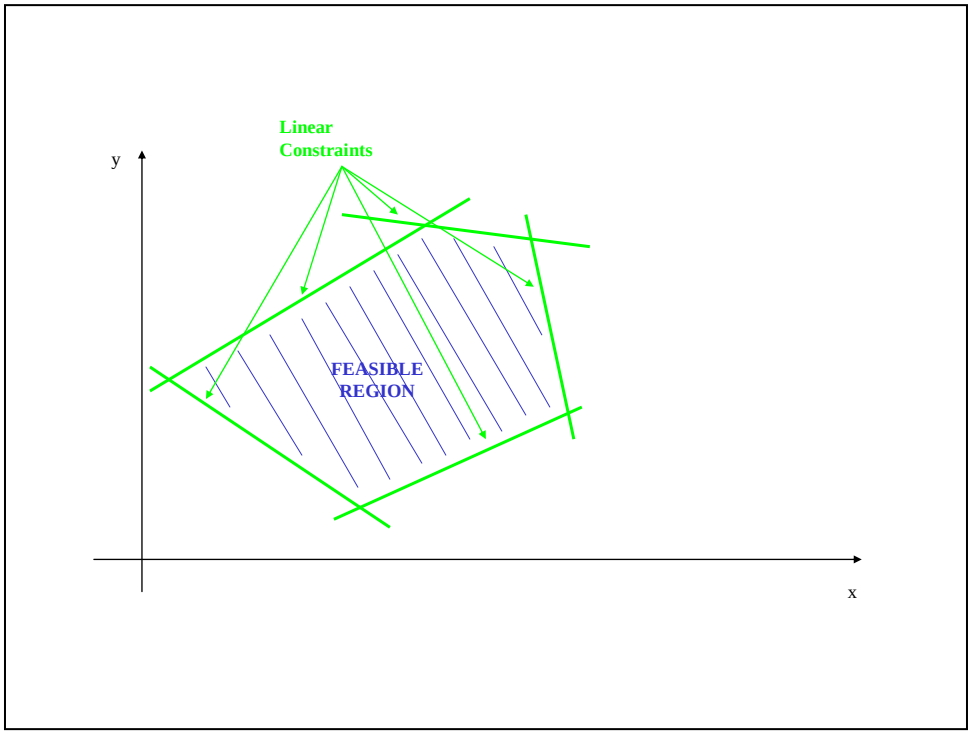
The following duality concepts provide insight into how the LP solver is used to implement the SCED algorithm:

- If a primal problem has an optimum solution (i.e., is not infeasible or unbounded), then the corresponding dual problem will also have an optimum solution.
- If the primal problem is infeasible, the dual problem will be either infeasible or unbounded.
- If the primal problem is unbounded, the dual problem will be infeasible.
- The objective cost of the optimal primal solution will be equal to the objective cost of the optimal dual solution.
- The duality gap is the difference between the objective cost of the primal solution and the objective cost of the dual solution. If the primal and dual solutions are both optimal, there is no duality gap.
- LP solvers typically solve both the primal and dual problem and then use the duality gap as a measure of how close the solutions are to optimal (the smaller the duality gap the better).

3.6 Qualitative Description of an LP Solver

To understand how an actual LP solver works, consider the feasible region defined by the linear constraints in the two variable optimization problem illustrated in Exhibit 3-2 below:

Exhibit 3-2: Two Variable Optimization Example

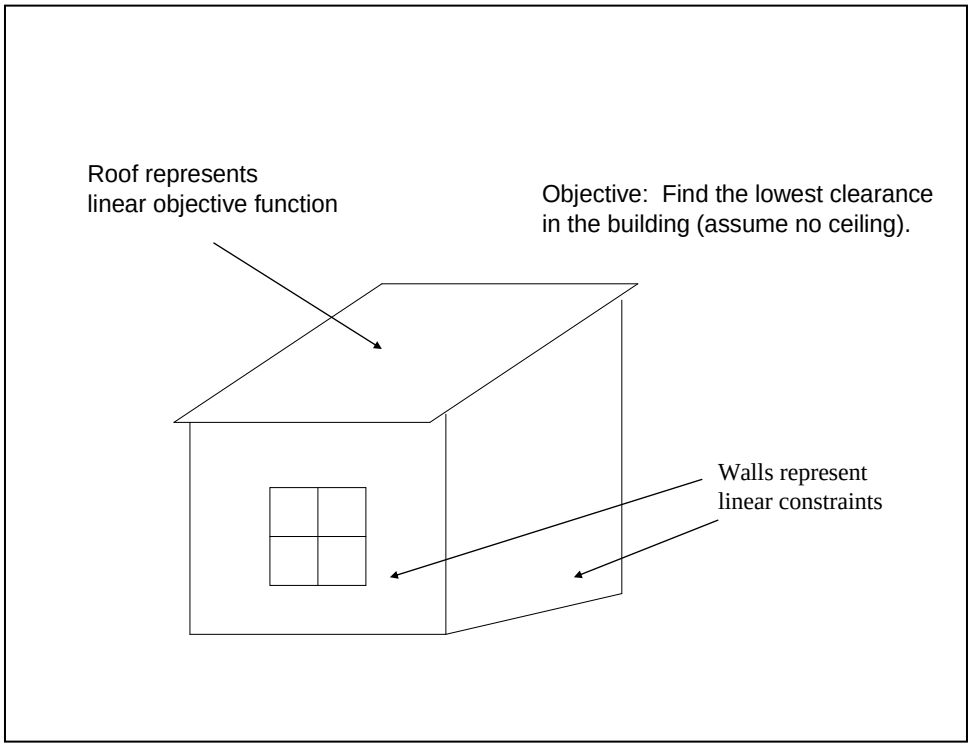


The following four statements can be made about Exhibit 3-2:

- **Statement 1:** The linear constraints are planes that define a convex feasible region, where a convex feasible region in two dimensions is one in which a line segment drawn from any point on the boundary to any other point on the boundary will lie totally within the feasible region (the boundary is considered part of the feasible region).
- **Statement 2:** The maximum and minimum solutions will lie on the boundary formed by the feasible region if the objective function is also linear and thus represented by a plane in three-dimensional space.
- **Statement 3:** If only a single maximum or minimum value exists for the optimization problem, it will lie at a vertex between two constraints. In the example in Exhibit 3-2, this would represent a solution where two adjacent constraints are binding.
- **Statement 4:** If there are multiple maximums or minimums, they will be equal to each other and will occur at two adjacent vertices and every point on the constraint that joins the two adjacent vertices. In the example in Exhibit 3-2, this would represent a solution where only one constraint is binding.

To better understand the four statements listed above, it is helpful to consider the example of a building with a shed roof (referred to as a SCED Shed) as depicted in Exhibit 3-3.

Exhibit 3-3: SCED Shed Example 1



The building in Exhibit 3-3 is a very good geometric representation of a two variable linear optimization problem. In Exhibit 3-3, the walls of the building, which are planes, could represent linear constraints. The feasible region could be represented by the inside of the building, and the roof, which is also a plane, could represent the linear objective function. Assume the objective is to find the low point of the roof inside the building. The walls, which represent constraints, eliminate the overhang as a feasible solution. It is obvious that the lowest clearance in the building is along the front wall that contains the window. In this particular LP problem, the minimum value is on the boundary between the two front corners (i.e., vertices) of the building. Thus Statements 2 and 4 apply. Keep in mind a sagging roof would be a non-linear objective function, as would a hip roof, gable roof or a roof with a dome, and Statements 1 through 4 would not necessarily apply.

Now consider a slight modification of the example above as depicted in Exhibit 3-4.

Exhibit 3-4: SCED Shed Example 2

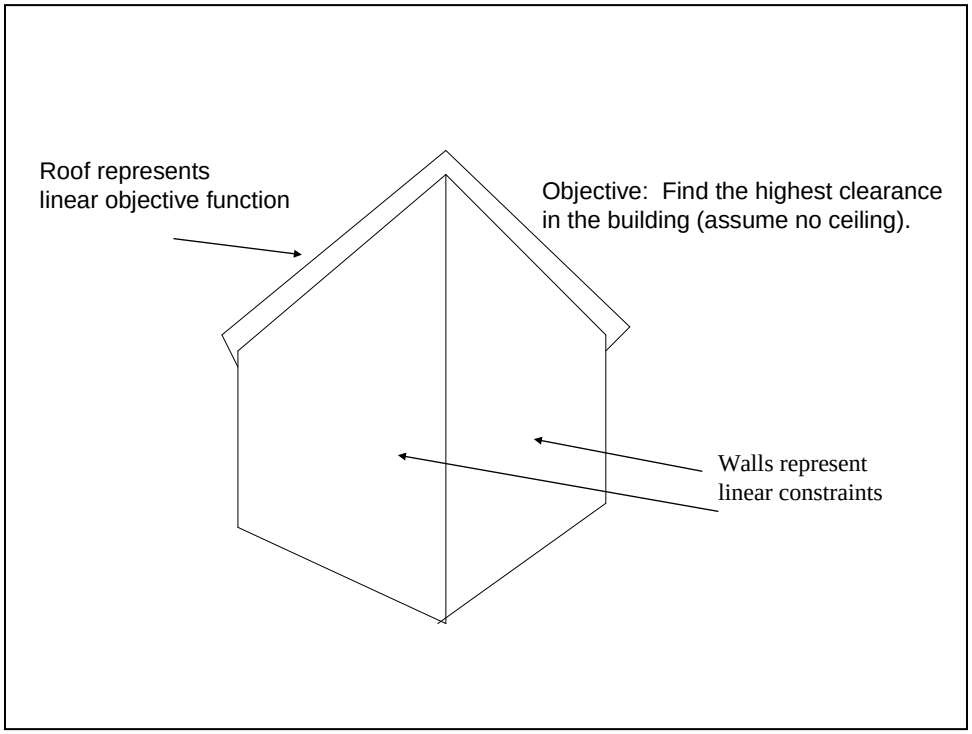
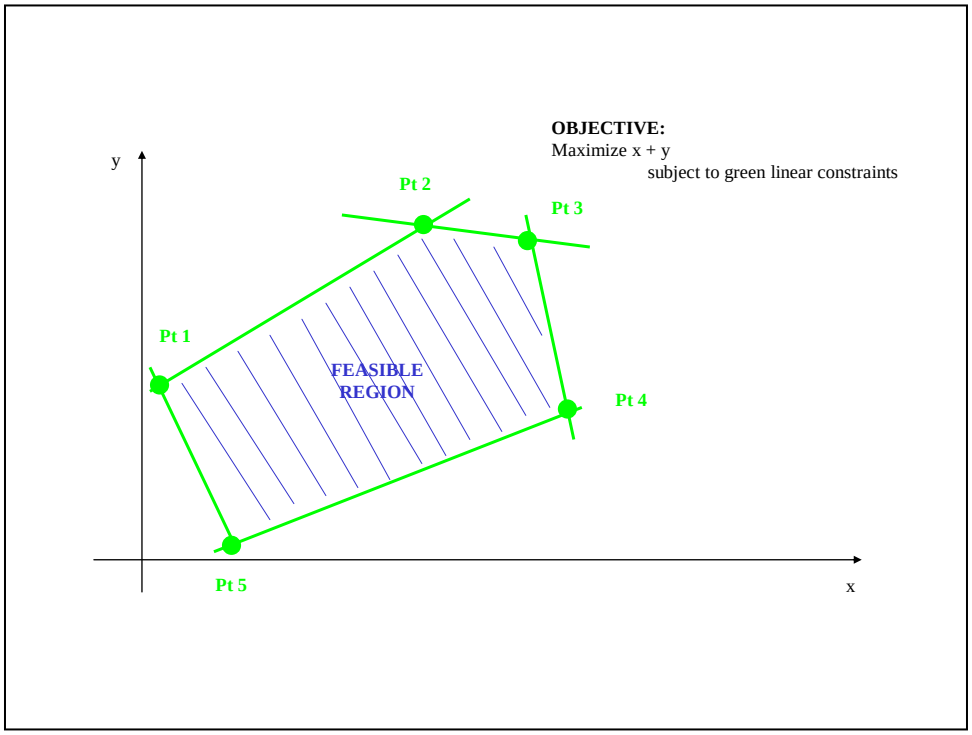


Exhibit 3-4 represents a building with a fancy shed roof that slopes from one corner of the building to the opposite corner rather than from one wall of the building to the opposite wall. In this particular example, Statement 2 still holds true in that the maximum and minimum points are on the boundary. However, in this example, there is only one maximum and minimum point, and they occur at the building corners (i.e., vertices). Therefore, in this particular example, Statements 2 and 3 apply.

LP solvers can take advantage of Statements 2, 3 and 4 which indicate that optimum solutions to linear optimization problems are always on the boundary and, if unique, will be located at a vertex or, if not unique, will include points on vertices. From a qualitative standpoint, all an LP solver basically has to do is start at a vertex, and then search adjacent vertices for a more optimum solution. The adjacent vertex with the most improvement is then considered and the process of searching adjacent vertices is repeated. When the LP solver reaches a vertex and the solutions at adjacent vertices are not more optimum, the LP solver has found the optimum solution. That is, if your goal is to get to the South Pole, you know you are there when the only direction you can travel in is North. Likewise, if your goal is to get to the North Pole, you know you are there when the only direction you can travel in is South.

The LP process is illustrated in Exhibit 3-5 below.

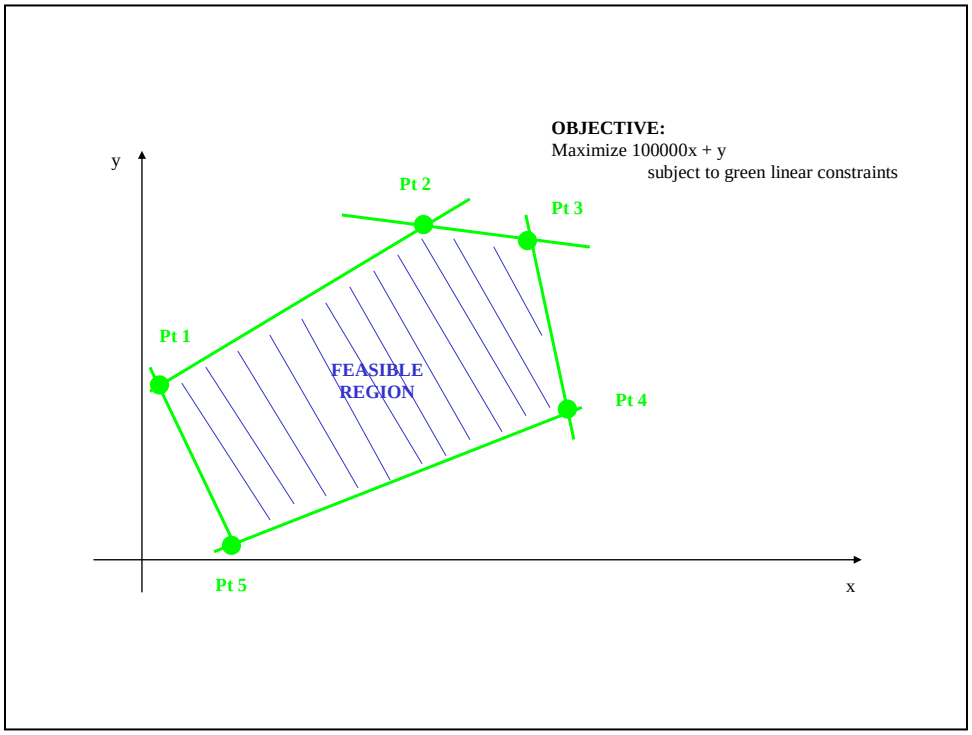
Exhibit 3-5: LP Process Example 1



Assume the LP solver initially selects Point 2 as a potential solution in Exhibit 3-5. The LP solver would then consider solutions at the adjacent vertices of Point 1 and Point 3. Point 1 represents a lower objective function value since both x and y are less at this point, whereas Point 3 represents a higher objective function value since x increases by a larger amount than y decreases. Since this is a maximization problem, the LP solver would choose Point 3 and then repeat the process by checking vertices adjacent to Point 3 (i.e., Point 2 and Point 4). Point 2 has already been ruled out as an optimum solution. The objective function value at Point 4 is lower than Point 3 since y decreases by a much larger amount than x increases. Since both Point 2 and Point 4 are less optimal than Point 3, the LP solver selects Point 3 as the optimal solution. In problems with many variables, the LP solver generally must traverse through only a handful of vertices to find the optimum solution. That is, if someone is located in Kansas and their objective is to get to Texas, that person only needs to travel through Oklahoma.

Now consider the same constraints and feasible region with a modified objective function as illustrated in Exhibit 3-6 below:

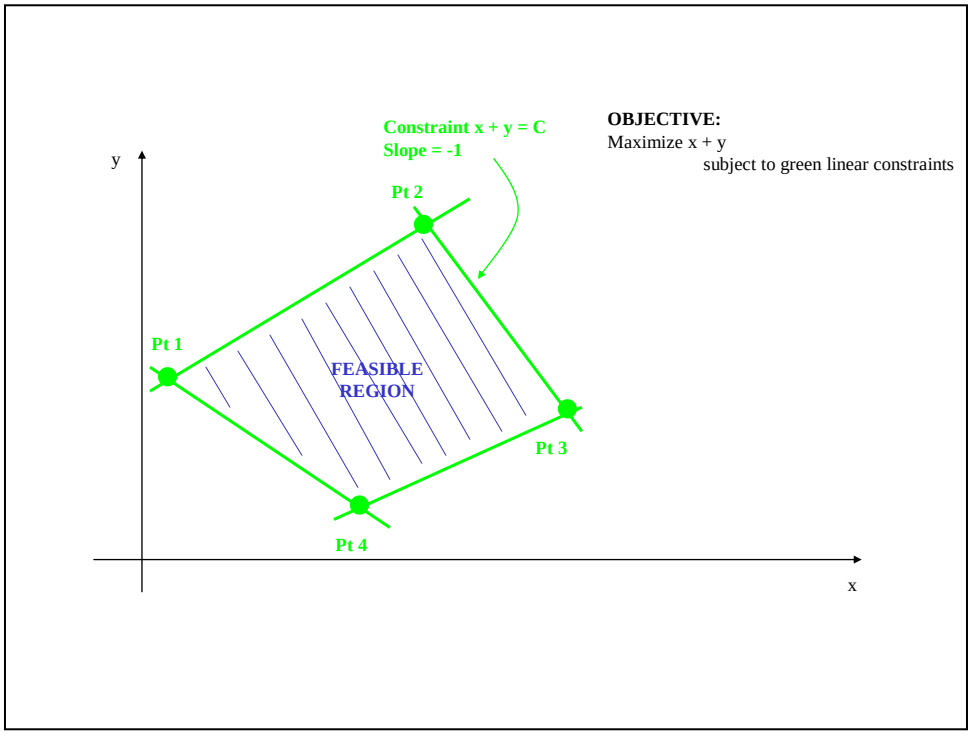
Exhibit 3-6: LP Process Example 2



Assume the LP solver selects Point 1 as the initial solution in Exhibit 3-6. The LP solver then checks the adjacent vertices of the Point 2 and Point 5. While both Point 2 and Point 5 represent more optimum solutions than Point 1, Point 2 represents the largest improvement because both x and y are larger at Point 2 than they are at Point 5. Therefore, Point 2 is chosen and the adjacent vertices of Point 3 and Point 1 are evaluated. Point 1 has already been ruled out as an optimum solution. Point 3 is more optimum than point 2 because x increases more than y decreases and because x is weighted much heavier than y in the objective function. The LP solver chooses Point 3 and then adjacent vertices of Point 4 and Point 2 are considered. Point 2 has already been ruled out as an optimum solution. Point 4 corresponds to a large decrease in y and a small increase in x. However, since x is weighted 100,000 times as much as y in the objective function, Point 4 actually represents a higher objective function value, and is chosen. Next the adjacent vertices of Point 5 and Point 3 are considered, but both have already been ruled out, so Point 4 is the optimum solution selected by the LP solver in this case.

One more example is presented in Exhibit 3-7 below:

Exhibit 3-7: LP Process Example 3



Assume the LP solver chooses Point 1 as the initial solution. The LP solver then evaluates solutions at Point 2 and Point 4. Since this is a maximization problem and Point 2 represents an increase in both x and y, Point 2 is more optimum than Point 1. Point 4 is also more optimum than Point 1 because x increases more than y decreases. However, both x and y at Point 2 are larger than they are at Point 4, so Point 2 is chosen as the next point. The LP solver once again checks vertices adjacent to Point 2. Point 1 has already been ruled out. Point 3 involves an increase in x and decrease in y. The magnitude of the increase in x is equal to the magnitude of the decrease in y since the slope of the constraint is -1. Therefore, the solution at Point 3 is just as optimum as the solution at Point 2. As a matter of fact, a solution lying anywhere on the boundary between Point 2 and Point 3 would be equally optimum. In this case, there are an infinite number of optimum solutions, but the LP solver always chooses a solution at a vertex, so the LP solver would choose either Point 2 or Point 3 in this case.

3.7 Addressing Infeasible Solutions

Real-world optimization problems generally have enough constraints to avoid situations where the solution is unbounded. However, there are often times when infeasible solutions could result

due to opposing constraints, and the LP solver must have a method to deal with these situations or a solution cannot be obtained.

There are two basic methods that are used to avoid infeasible solutions. The first method involves input validation. For example, if a participant enters a maximum limit for a generator of 100 MW and a minimum limit for that same generator of 200 MW, it is not possible for the SCED algorithm to find a dispatch value that is both less than 100 MW and greater than 200 MW. This situation is driven by an obvious error in the input data, and the validation programs can be utilized to reject this input.

Unfortunately, there can also be opposing constraints that cannot be detected via input validation. For example, consider a radial transmission line with a limit of 100 MW that serves a Bus with a fixed Load of 500 MW and a Generator with a maximum limit of 200 MW. This situation results in three constraints in the SCED algorithm, one of which fixes the Load at 500 MW (equality constraint), one of which does not allow the Generator to produce more than 200 MW (LE constraint) and one of which does not allow the transmission line to be loaded above 100 MW (LE constraint). Unfortunately, all three of these constraints cannot be enforced. To avoid an infeasible solution, the LP solver is able to prioritize constraints and then allow lower priority constraints to be violated when necessary to obtain a feasible solution.

To understand how the LP solver prioritizes constraints, it is important to understand the difference between a hard constraint and a penalized constraint. A hard constraint has one of the following forms and must be satisfied or the LP solver will fail.

- Hard Equality Constraint:

  $$\sum_{i} A(i) * x(i) = b \qquad \text{(A-58)}$$

- Hard GE Constraint:

  $$\sum_{i} A(i) * x(i) \geq b \qquad \text{(A-59)}$$

- Hard LE Constraint:

  $$\sum_{I} A(i) * x(i) \leq b \qquad \text{(A-60)}$$

A penalized constraint has one of the following forms.

- Penalized Equality Constraint:

  $$\sum_{i} A(i) * x(i) = b + \text{Surplus Violation} - \text{Deficit Violation} \quad \text{(A-61)}$$

- Penalized GE Constraint:

  $$\sum_{i} A(i) * x(i) \geq b - \text{Deficit Violation} \quad \text{(A-62)}$$

- Penalized LE Constraint:

  $$\sum_{I} A(i) * x(i) \leq b + \text{Surplus Violation} \quad \text{(A-63)}$$

The additional terms Surplus Violation and Deficit Violation are additional, non-negative, primal variables known as violation variables. The idea is that these violation variables should normally have a value of zero. However, should it not be possible to satisfy a certain GE Constraint, the Deficit Violation variable for that constraint would take on a positive non-zero value, effectively lowering the limit. Likewise, should it not be possible to satisfy a certain LE Constraint, the Surplus Violation variable would take on a positive non-zero value, effectively raising the limit. Since an equality constraint is modeled as a GE and LE constraint, it would have both a Deficit Violation variable and a Surplus Violation variable.

In order to ensure violation variables are zero whenever possible and in order to prioritize which constraints violate first, special penalty pricing terms of the following form are added to the objective function for each penalized constraint:

- Penalty Price * Surplus Violation (for each LE and equality constraint)
- Penalty Price * Deficit Violation (for each GE and equality constraint)

By setting each penalty price to a very high value (i.e., penalty prices are positive in minimization problems), the cost minimization objective of the SCUC and SCED algorithms will attempt to set the violation variables to zero unless there are opposing constraints. In the case of opposing constraints, the constraint with the lower penalty price is violated first, so the magnitude of the penalty price sets the priority of the constraint.

At the MISO, constraints are classified into three broad categories: physical constraints, reliability constraints and good utility practice constraints. Physical constraints include such constraints as limit constraints and ramp rate constraints and have the highest priority. These constraints are either modeled as hard constraints or with very high penalty prices. Reliability constraints include

such constraints as reserve requirement constraints, power balance constraints and transmission constraints. These constraints have a lower priority than physical constraints. Good utility practice constraints have the lowest priority. Good utility practice constraints include items such as self-schedules and the constraints that limit the amount of regulating reserve or contingency reserve that can clear on a specific resource to a pre-determined percentage of the reserve requirement.

It is important to note that if a constraint violates, it is generally not appropriate to allow the arbitrary penalty price to set LMPs and MCPs unless the penalty price is specified in the tariff. Demand curves are examples of penalty prices that are specified in the tariff, and in the case of operating reserve demand curves, the penalty prices are multi-valued. For example, the zonal operating reserve demand curve specifies a scarcity price of $1,100 for cleared zonal operating reserve between 10% of the zonal operating reserve requirement and 100% of the zonal operating reserve requirement and a scarcity price of $2,500 for cleared zonal operating reserve between 0% of the zonal operating reserve requirement and 10 % of the zonal operating reserve requirement. To model this in the SCED algorithm, there would be two violation variables for the zonal operating reserve constraint, one of which had a penalty price of $1,100 and one of which had a penalty price of $2,500. A maximum constraint of 90% of the zonal operating reserve requirement would be applied to the violation variable with a penalty price of $1,100 to force the scarcity price to $2,500 when the cleared zonal operating reserve reduces below 10% of the zonal operating reserve requirement.

In the case of other types of constraints such as transmission constraints, penalty prices or demand curves are not specified in the tariff. Therefore, if these constraints must be violated to avoid violation of a higher priority constraint, it is necessary to relax the constraint limit by the violation amount and re-execute the SCED algorithm. The idea is that the second SCED iteration will provide the same primal solution as the initial SCED algorithm, but with the relaxed limit, the violation variable will be zero during the second iteration and prices will be based on the maximum supply price rather than an arbitrary penalty price not specified in the tariff.

3.8 Addressing Multiple Optimum Solutions

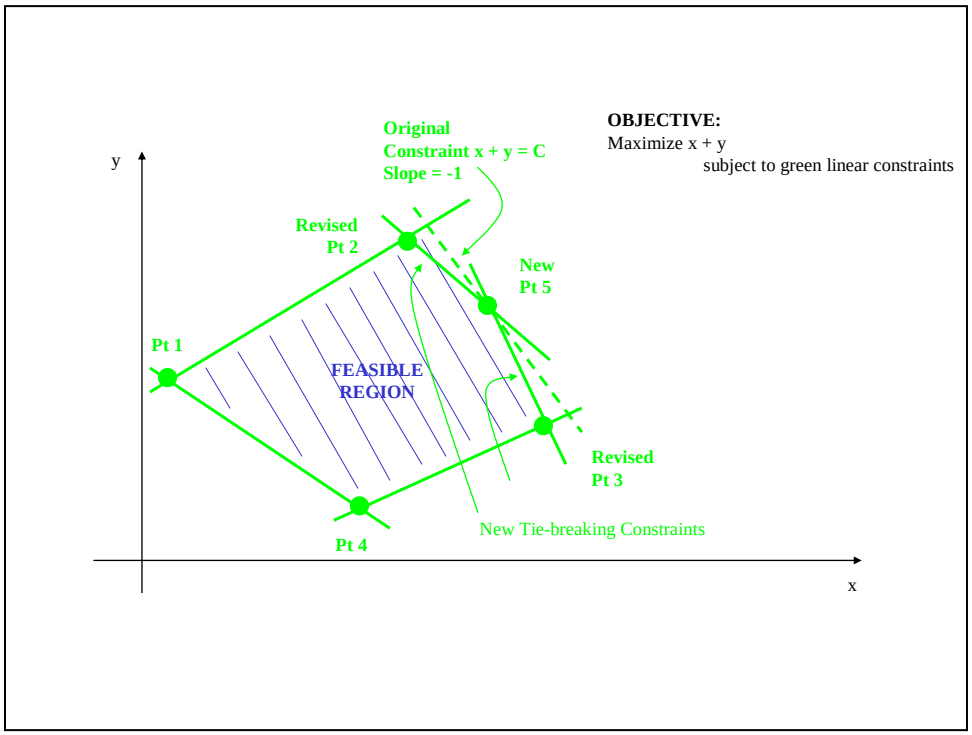
As stated above, an LP solver always selects a solution at a vertex. In the MISO markets, it is possible to have situations where there are multiple optimum solutions. For example, assume two generators, designated as Generator A and Generator B are located at the same Bus, have the same offers, and represent the marginal units for energy. Assume each generator has a minimum limit of 100 MW and a maximum limit of 500 MW, and there is 600 MW of energy to be dispatched between the units. Because the offers are the same, dispatching 100 MW on Generator A and

500 MW on Generator B would be just as optimal as dispatching 500 MW on Generator A and 100 MW on Generator B, which in turn would be just as optimal as dispatching 300 MW on each generator. The scenario of dispatching 300 MW on each generator is the fair solution, but would never be chosen because it actually represents a point on the boundary (a point where the limit constraints do not bind). Instead, the LP solver would choose to dispatch 100 MW on one generator and 500 MW on the other generator, and the choice is somewhat arbitrary. This results in an unfair dispatch since one market participant is arbitrarily favored over the other.

To address this situation, tie-breaking constraints are introduced. A tie breaking constraint is a constraint that forces equal dispatch between two units, but only when everything else is equal. That is, tie-breaking constraints have violation variables with very low penalty prices (about $1 x 10^{-6}). Given two or more offers are tied, this constraint forces equal dispatch between the two. If the offers are not tied based on economics and/or other constraints, the low penalty cost of violating the tie-breaking constraint will cause the constraint to be overridden by the selection of a more economical solution and/or the enforcement of one or more constraints with higher penalty prices.

Exhibit 3-8 below, which is an extension of Exhibit 3-7, illustrates how a tie-breaking constraint would force a single, fair optimum solution at the new Point 5 which is on the original boundary between original Points 2 and 3:

Exhibit 3-8: LP Process Tie Breaking Constraints



4. Security Constrained Unit Commitment ("SCUC") Algorithm Overview

The MISO uses a simultaneously co-optimized Security Constrained Unit Commitment ("SCUC") algorithm to perform the following core functions.

- Commit Resource in the Day-Ahead Market
- Commit Resources in the Reliability Assessment Commitment ("RAC") Process
- Schedule Resources to regulate in the Day-Ahead Market
- Schedule Resources to regulate in the RAC Process
- Release Emergency operating ranges on Resources in the Day-Ahead Market
- Release Emergency operating ranges on Resources in the RAC process

4.1 The Ideal SCUC Algorithm

The ideal SCUC algorithm would consist of a SCED algorithm that would be independently executed for each possible commitment scenario. The algorithm would then discard all of the infeasible scenarios, and choose the feasible SCED solution with the lowest cost. The commitment flag inputs associated with the low cost SCED solution would then become commitment results. Such a SCUC algorithm would produce highly accurate commitment results, about the same accuracy level as typical SCED results.

Unfortunately, there is a problem with the ideal SCUC algorithm. For an Energy-only market with only four Resources and a time horizon of twenty-four hours, there are 96 commitment decisions to make (one per resource per hour). The number of SCED scenarios that would need to be executed in the ideal SCUC algorithm can be expressed as follows:

$$\text{Number of SCED Executions} = 2^{96} \approx 8 * 10^{28} \pm \text{a few trillion} \qquad \text{(A-64)}$$

Assuming a very fast SCED solver that takes only 1 second per scenario, the ideal SCUC algorithm in an Energy-only market with only four Resources cold generate a commitment solution in as little as 2.5×10^{19} centuries. If there was a way to eliminate 99.9999% of the scenarios ahead of time, the commitment execution time would significantly improve to 2.5×10^{13} centuries

Unfortunately, the MISO market is not an energy-only market and has well over 4 resources, and even if it did not, an execution time of 2.5×10^{13} centuries will not meet the market timing requirements outlined in the Tariff.

Fortunately, there are other ways to solve the problem. The MISO uses a Mixed Integer Programming ("MIP") solver to implement a SCUC algorithm that generates highly accurate commitment solutions within a reasonable time frame (well under 2.5×10^{13} centuries).

4.2 Mixed Integer Programming ("MIP") Solvers

All SCUC algorithms use a Mixed Integer Programming ("MIP") solver. A MIP solver is similar to an LP solver, but allows primal variables to either be continuous (a requirement of an LP solver) or integer. The SCUC problems formulated with MIP solvers in the MISO market are special types of MIP solvers where the integer variable are restricted to Boolean variables.

In general, LP solvers cannot handle integer or other types of non-continuous variables. However, a MIP solver operates by first converting a non-continuous problem (such as the commitment problem) into a continuous problem by allowing the Boolean variables to be continuous from 0 to 1 inclusive. That is, the Boolean variable are modeled as continuous variables (referred to as relaxed Boolean variables) with a constraint that these continuous variables, or relaxed Boolean Variable, be less than or equal to 1 (by definition, LP solver primal variables cannot be negative). A standard LP solver is then used to solve a continuous linear problem (referred to as the relaxed problem since the Boolean variables have been relaxed to continuous variables). That is, the relaxed problem allows decision variables such as commitment flags or regulation flags to take on non-Boolean values between 0 and 1. The resulting solution for the objective function will actually be more optimal than the true optimum solution since the relaxed LP problem is less restrictive (e.g., actually allows a unit commitment flag to be 0.3, etc.).

Once the initial relaxed problem is solved, special tools can be used to modify the problem into a more restrictive LP problem and then re-solve. These tools are called cutting planes. Cutting planes are basically additional linear constraint that reduce the size of the relaxed problem's feasible region, cutting off the relaxed solution, and forcing a subsequent LP solution to move toward the optimum non-relaxed solution (where a non-relaxed solution is a solution where the relaxed Boolean variables take on Boolean values). The idea is that the problem is always formulated as a continuous linear problem and solved by an LP solver, but eventually, the cutting planes will drive the LP solver toward a solution where the relaxed Boolean variables take on values of either 0 or 1 (i.e., a non-relaxed solution), even though they have been modeled as continuous variables that may take on any value between 0 and 1. Should the cutting planes not be able to drive the LP solver all the way to a non-relaxed solution, branch and bound algorithms can be used as a last resort to obtain a non-relaxed solution. Therefore, a MIP solver is basically

an LP solver, a cutting plane generation algorithm and a branch and bound algorithm that iterate together to drive a relaxed linear problem toward a non-relaxed solution.

After several iterations between the LP solver and the cutting plane generation algorithm, the LP solution approaches a non-relaxed solution. At this point, the MIP solver would compare the objective cost of the initial (fully relaxed) LP solution with that of the final (non-relaxed) LP solution. The difference in these objective costs is then divided by the initial solution's objective cost to calculate what is known as the MIP Gap as illustrated below:

MIP Gap = (Final LP Objective Cost - Initial LP Objective Cost)
/ Initial LP Objective cost (A-65)

The MIP gap basically indicates how close the final non-relaxed solution is to the optimum non-relaxed solution. That is, the initial solution is actually better than optimum since it is relaxed (allows Boolean variables to take on continuous values). Therefore, if the final LP objective cost is close to the initial LP objective cost, the final LP objective cost must be close to the optimum solution. For example, if the initial objective cost is $80 Million and the final objective cost is at $80.1 million, then at the very worst, the final solution is within 0.125% of the optimum solution and could be even closer to the optimum since the $80 million solution is most likely better than optimum.

The MIP gap tolerance for the Day-Ahead Market SCUC algorithm is typically set at 0.1%, which means the SCUC algorithm will continue looking for better solutions until the MIP gap is at or below 0.1%. The actual MIP gap achieved in most Day-Ahead Market SCUC executions is less than 0.1%.

With regard to the RAC process, the MIP gaps tolerance is typically set at 0.5%, but the average MIP gap of a RAC process is generally below this number as well.

A very simple example of how a MIP solver works is illustrated in Exhibits 4-1 through 4-7.

Exhibit 4-1: MIP Algorithm Example

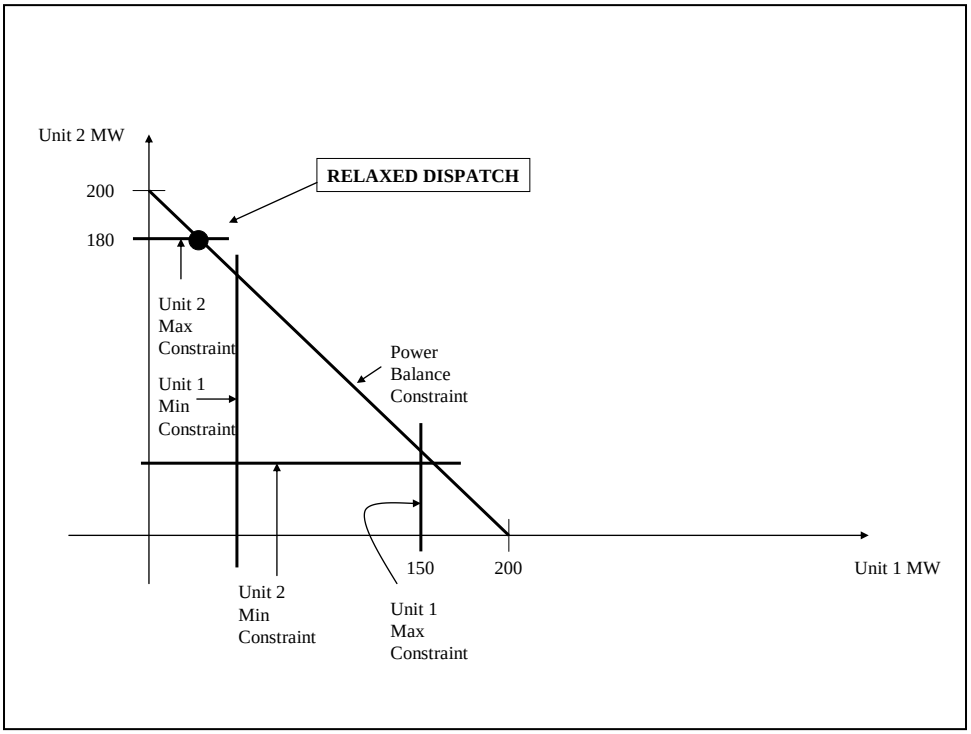


Exhibit 4-1 outlines a very simple SCUC problem consisting of two units and a SCUC period of 1 hour. The only constraints modeled are the maximum and minimum limit constraints for each unit and a power balance constraint. The feasible region for the non-relaxed problem is represented by the inner right triangle. This problem is shown in two dimensions but actually has four variables (a commitment flag (Boolean) and Energy dispatch (continuous) variable for each of the two units).

Since unit 2 is cheaper, the initial relaxed LP solution dispatches unit 2 to its maximum limit and unit 1 below its minimum limit (it can do this because the commitment flags are allowed to take on values between 0 and 1 initially). Therefore, the feasible region for the relaxed problems is the outer right triangle that allows operation of the units below the minimum limits, with the corners cut off by the maximum limit constraints (i.e., the commitment flags cannot take on values greater than 1). The initial LP solution is at a vertex of the relaxed feasible region.

Exhibit 4-2 below illustrates the application of the first cutting plane, which eliminates the initial relaxed LP solution, but does not cut into the non-relaxed feasible region.

Exhibit 4-2: MIP Algorithm Example – First Cutting Plane

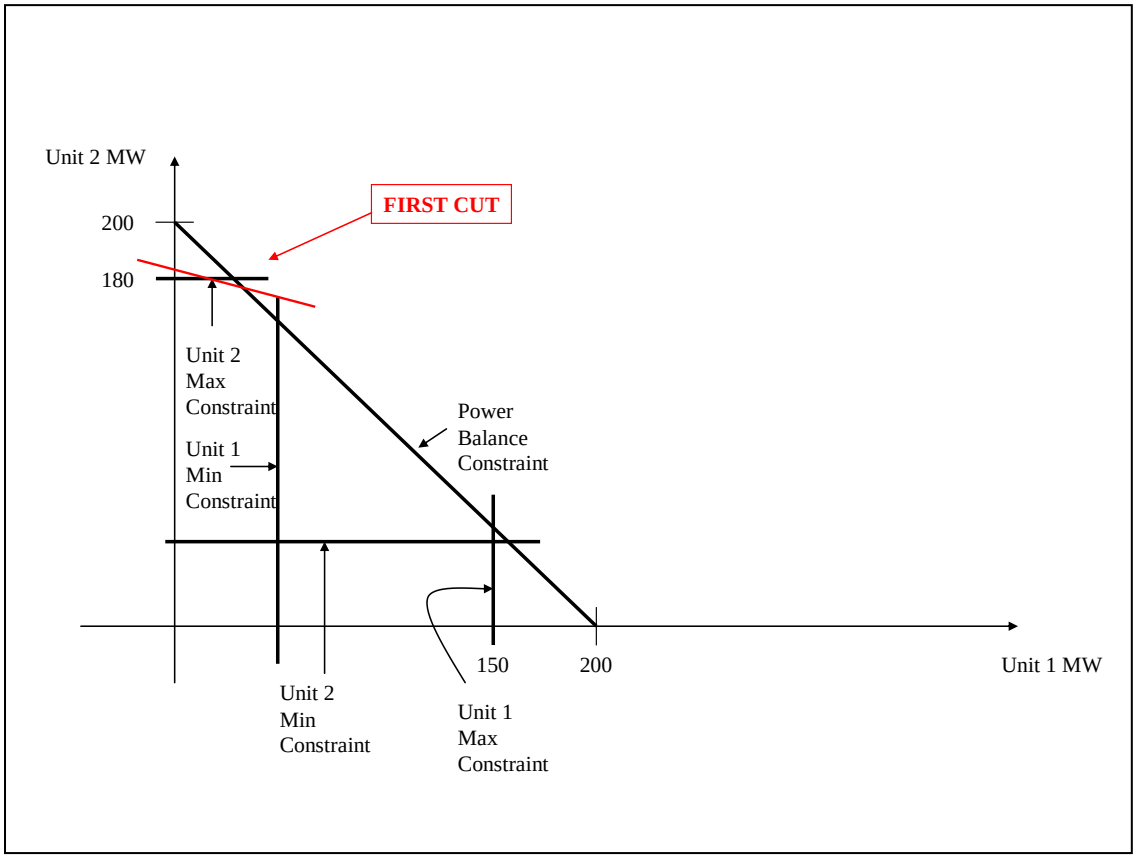
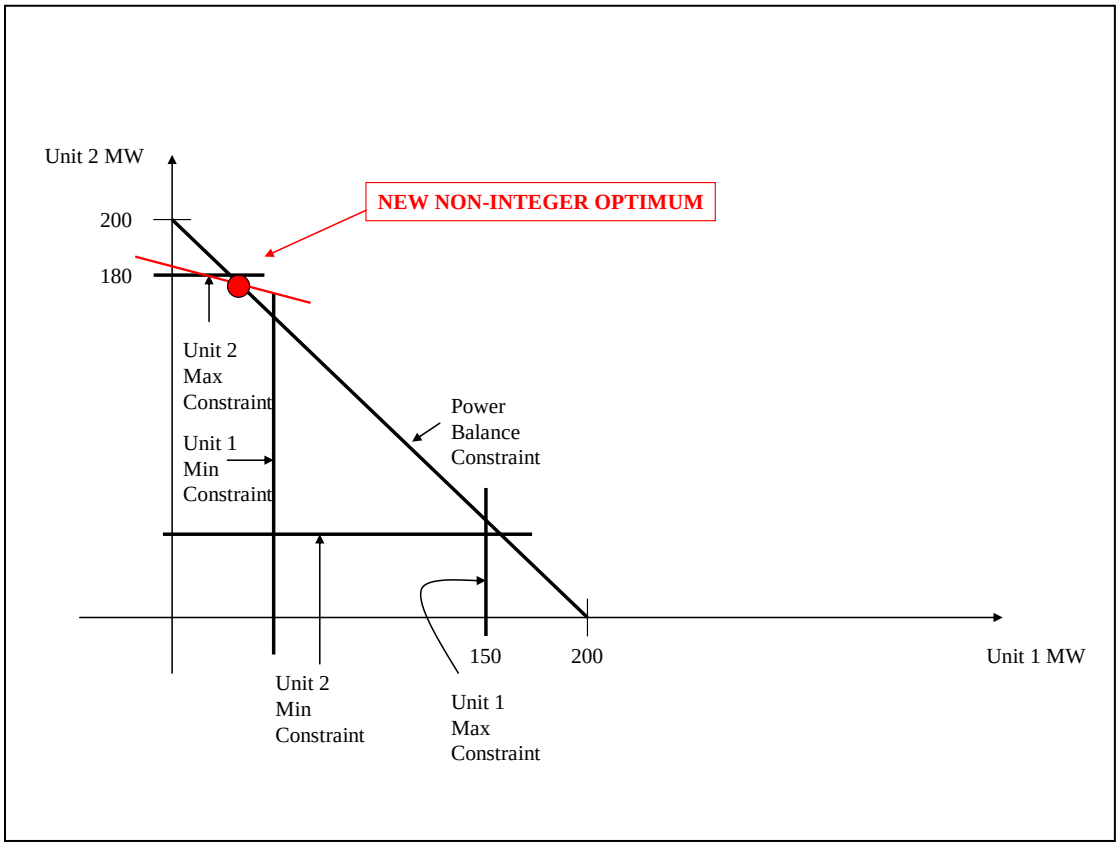


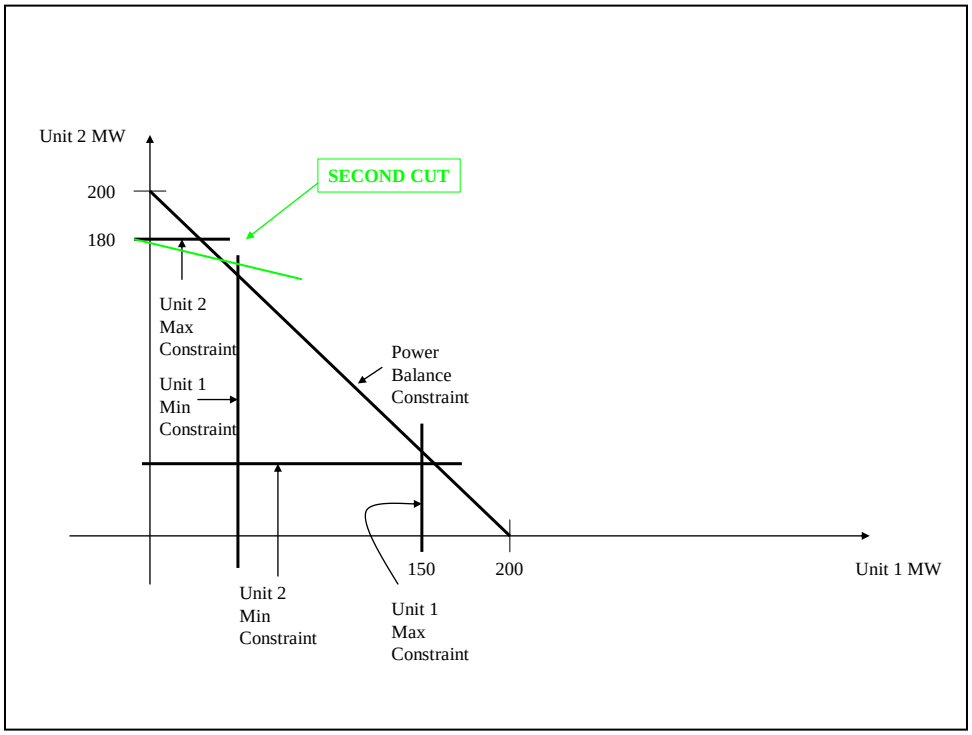
Exhibit 4-3 below illustrates the second iteration of the LP solver, subject to the cutting plane constraint.

Exhibit 4-3: MIP Algorithm Example – Second LP Solution



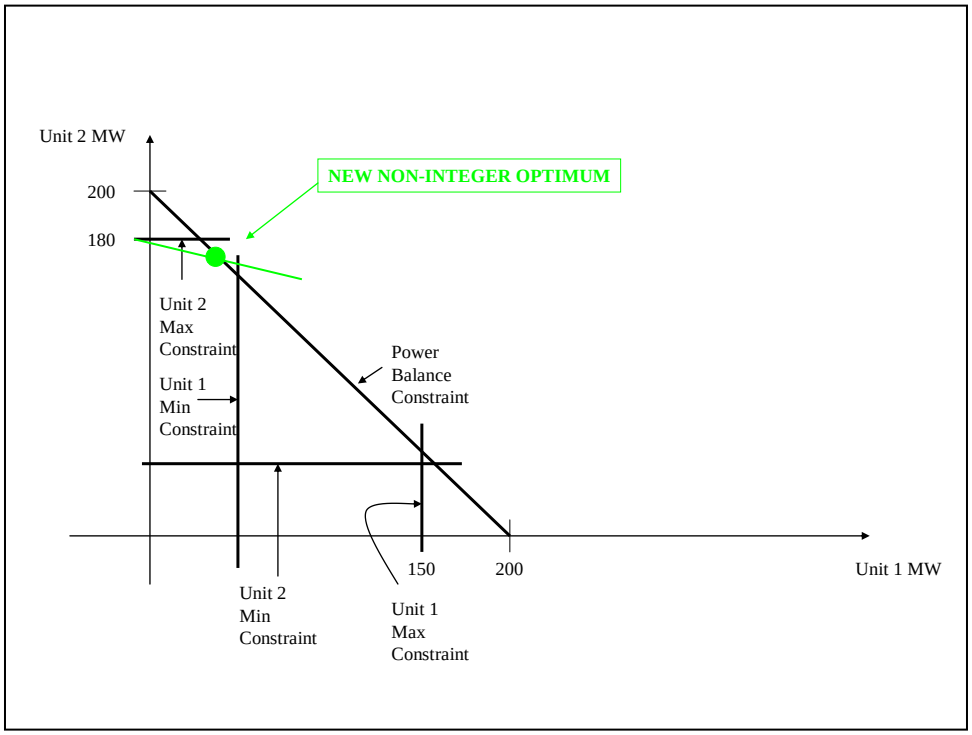
While the second LP solution is closer to a non-relaxed solution, another cutting plane is required, and is illustrated below in Exhibit 4-4:

Exhibit 4-4: MIP Algorithm Example – Second Cutting Plan



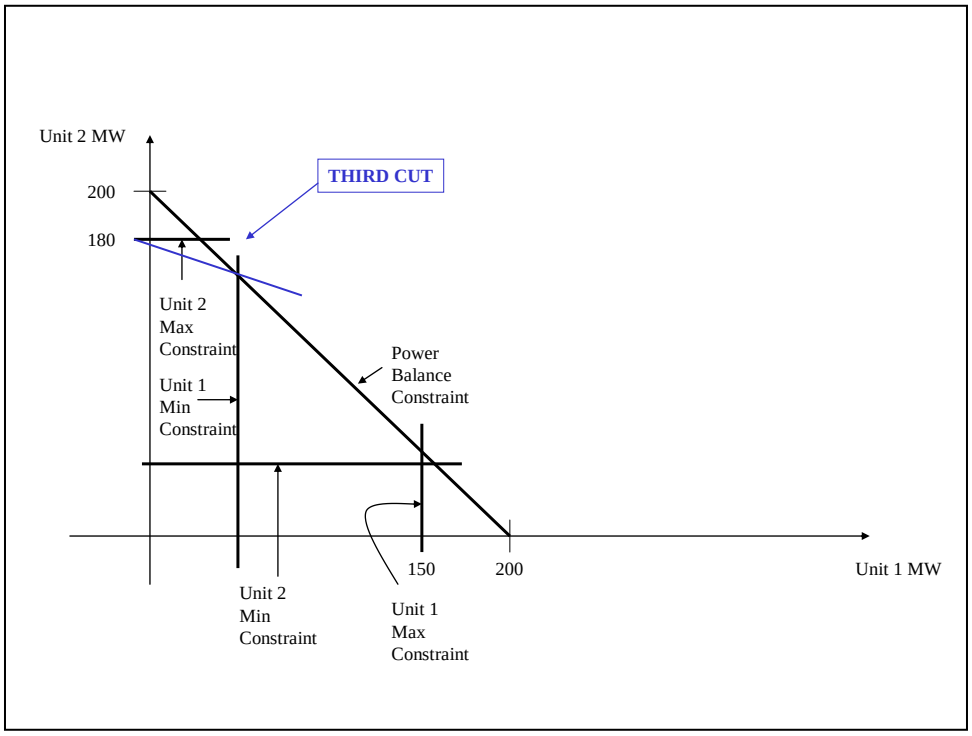
With the second cutting plane constraint in place, the LP solver executes a third time, as illustrated in Exhibit 4-5 below:

Exhibit 4-5: MIP Algorithm Example – Third LP Solution



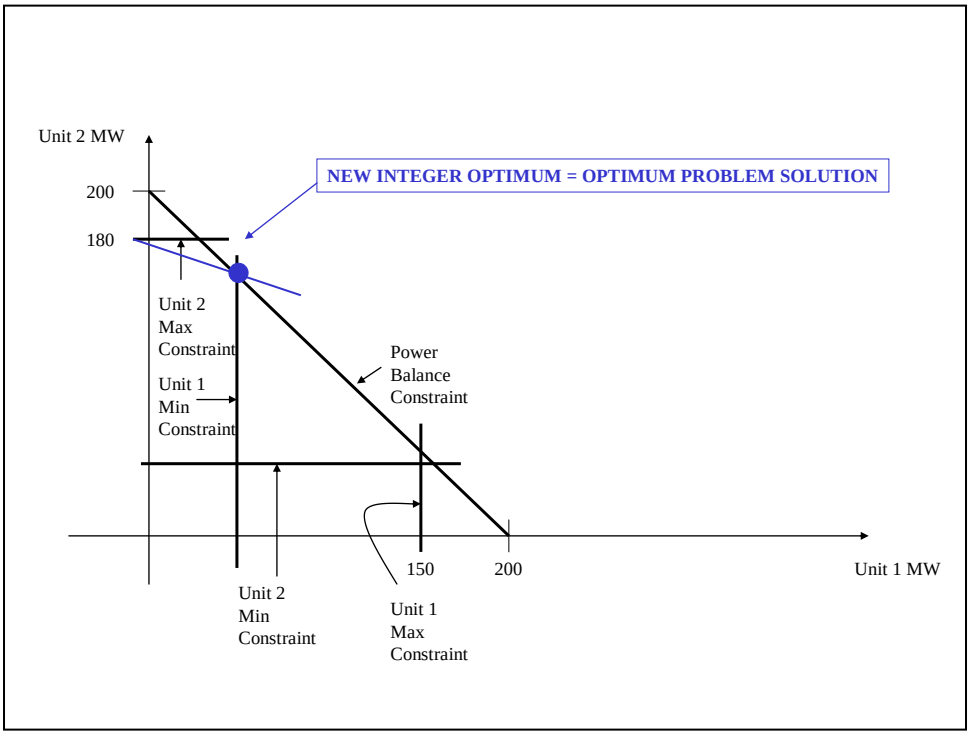
While the third LP solution is getting very close to a non-relaxed solution, at least one more cutting plane is required to drive the LP solver closer to a non-relaxed solution. This is illustrated in Exhibit 4-6 below:

Exhibit 4-6: MIP Algorithm Example – Third Cutting Plane



With the third cutting plane in place, the LP solver executes a fourth time, as illustrated in Exhibit 4-7 below:

Exhibit 4-7: MIP Algorithm Example – Fourth LP Solution



The fourth and final LP solution has achieved the optimum non-relaxed solution using three cutting planes.

In summary, while MIP solvers are not quite as accurate as LP solvers and require more memory and time to execute, recent technological advances allow today's MIP solvers and cutting planes to perform the SCUC function with very acceptable accuracy and performance levels (as evidenced by actual MIP gaps and solve times), even in a market as large as the MISO market.

4.3 Refinement to MIP Solver

On occasion, cases can solve with poor MIP gaps or poor performance. Three r algorithms to refine the MIP solver have been created to account for these issues. A case may be re-executed with a refinement algorithm based on the strengths and weakness of each algorithm compared to the issues observed in the initial MIP solver execution.

4.3.1 Decomposition

Decomposition is a general approach to solve a problem by breaking it up into smaller ones and solving each of the smaller ones separately, either in parallel or sequentially. The decomposition algorithm itself is working properly in RSC.

4.3.1.1 Algorithm

Step 1: If there is a source case RSC specified for polishing as indicated in section 3, then start with source case commitment. Otherwise use MIP to solve unit commitment problem (i.e. master problem) with no transmission constraints.

Step 2: Freeze the commitment variables and solve the original model using linear programming ("LP").

Step 3: Pick all violated and binding transmission constraints from the LP run and feed them back into the original problem and then re-optimize the MIP with additional transmission constraints.

Step 4: Compare the objectives between Step 2 and Step 3. Stop the iteration if they are within the gap requirement. If not, go to Step 3. This algorithm may take 2~3 iterations. Set the iteration number limit to be an RSC option. To improve performance, the multi-thread approach may be used.

4.3.2 MISO Lagrangian Relaxation Polishing

In constrained optimization problems, Lagrangian Relaxation techniques can be used to find the optimal value of a function that is subjected to constraints. With the introduction of additional variables into the problem, the lagrangian multiplier method converts the constrained problem into an unconstrained problem.

4.3.2.1 Algorithm

Step 1: If there is a source case RSC specified for polishing as indicated in section 3, then start with source case commitment. Otherwise use MIP to solve unit commitment problem (i.e. master problem) with no transmission constraints.

Step 2: Freeze the commitment variables and solve the original model using Linear programming ("LP").

Step 3: Based on the prices, solve profit maximization for each resources

Step 4: Calculate the profit of step 3, select the top number of (for example, 20, make this a RSC option value) resources out of the money for commitment adjustment. Freeze commitment variables of all other resources. Solve MIP for the top resources. Stop the iteration if they are within the gap requirement.

If not, go to Step 3.

4.3.3 IBM Lagrangian Relaxation

IBM Lagrangian Relaxation is an approach from IBM with polishing from MISO. It does not start from other RSC commitment solutions. If a source case RSC solution is specified and this method is selected, then the source case RSC solution is just ignored.