The University of Danang
University of Science and Technology

MATHEMATICS FOR COMPUTER SCIENCE

Chap 4. Optimizations

Faculty of Information Technology
PHAM Cong Thang, PhD

References

1. M. P. Deisenroth, A. A. Faisal, and C. S. Ong, Mathematics for Machine Learning, Cambridge University Press; 1 edition (April 23, 2020), 398 pages.
2. E. Lehman, F. T. Leighton, A. R. Meyer, 2017, Mathematics for Computer Science, Eric Lehman Google Inc, 998 pages
3. A. Laaksonen, Competitive Programmer's Handbook, 2018, 286 pages.
4. W. H. Press, S. A. Teukolsky, W.T. Vetterling, B. P. Flannery Numerical Recipes: The Art of Scientific Computing, Third Edition, Cambridge University Press, 1262 pages.
5. Other online/offline learning resources

Optimizations

- **Introduction**
- Linear Programming (simplex method, M-Method)
- Simulated Annealing Methods
- Programming nonlinear

Optimizations - Introduction

- Since machine learning algorithms are implemented on a computer,
  - The mathematical formulations are expressed as numerical optimization methods.
  - An optimal solution is a set of values of the variables that are contained in the feasible region and also provide the best value of the objective function

Optimizations - Introduction

- Since machine learning algorithms are implemented on a computer
  - Optimization Using Gradient Descent
  - Constrained Optimization and Lagrange Multipliers
  - Convex Optimization

Optimization Using Gradient Descent

- We now consider the problem of solving for the minimum of a real-valued function

$$\min_x f(x)$$

where $f : \mathbb{R}^d \longrightarrow \mathbb{R}$ is an objective function that captures the machine learning problem at hand. We assume that our function f is differentiable, and we are unable to analytically find a solution in closed form.

- Gradient descent is a first-order optimization algorithm. To find a local minimum of a function using gradient descent, one takes steps proportional to the negative of the gradient of the function at the current point.
- Another useful intuition is to consider the set of lines where the function is at a certain value ($f(x) = c$ for some value $c \in \mathbb{R}$), which are known as the contour lines. The gradient points in a direction that is orthogonal to the contour lines of the function we wish to optimize.

Optimization Using Gradient Descent

- Let us consider multivariate functions.
  - Imagine a surface (described by the function $f(x)$) with a ball starting at a particular location x_0.
  - When the ball is released, it will move downhill in the direction of steepest descent.
  - Gradient descent exploits the fact that $f(x_0)$ decreases fastest if one moves from x0 in the direction of the negative gradient $-\left((\nabla f)(x_0)\right)^T$ of f at $x0$

Optimization Using Gradient Descent

- Let us consider multivariate functions. We assume that the functions $f(x)$ are differentiable, then if

$$x_1 = x_0 - \gamma \left((\nabla f)(x_0)\right)^T$$

for a small *step-size* $\gamma > 0$, then $f(x_1) \leq f(x_0)$

  - Note that we use the transpose for the gradient since otherwise the dimensions will not work out.

Optimization Using Gradient Descent

- This observation allows us to define a simple gradient descent algorithm:
  - If we want to find a local optimum $f(x^*))$ of a function

$$f : \mathbb{R}^n \to \mathbb{R}, x \to f(x)$$

  - We start with an initial guess x0 of the parameters we wish to optimize and then iterate according to

$$x_{i+1} = x_i - \gamma_i \left((\nabla f)(x_i)\right)^T$$

  - For suitable step-size γ_i, the sequence $f(x_0) \geq f(x_1) \geq \cdots$ converges to a local minimum.

Optimization Using Gradient Descent

- Example: Consider a quadratic function in two dimensions

$$f\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right) = \frac{1}{2}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}^\top \begin{bmatrix} 2 & 1 \\ 1 & 20 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} - \begin{bmatrix} 5 \\ 3 \end{bmatrix}^\top \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

with gradient

$$\nabla f\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right) = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}^\top \begin{bmatrix} 2 & 1 \\ 1 & 20 \end{bmatrix} - \begin{bmatrix} 5 \\ 3 \end{bmatrix}^\top$$

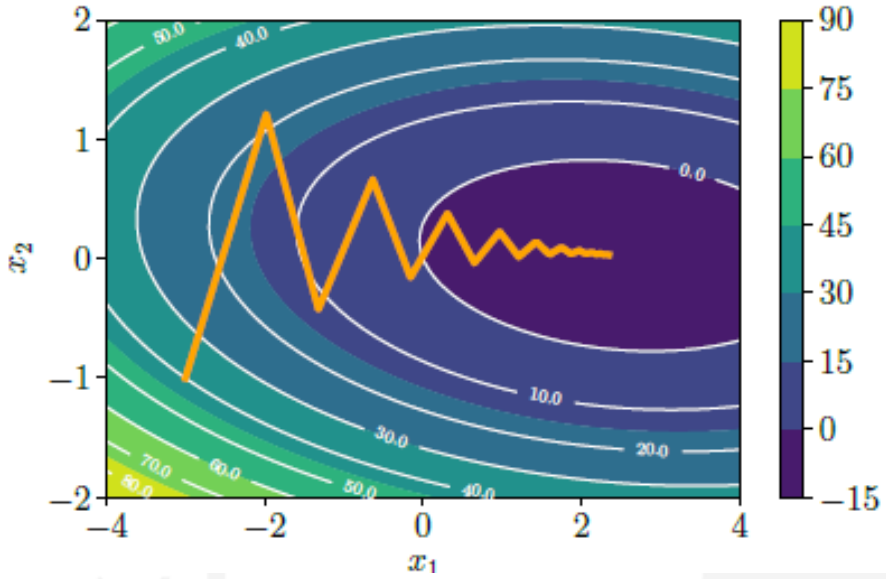
- Starting at the initial location $x_0 = [-3; -1]^T$, we iteratively apply gradient descent to obtain a sequence of estimates that converge to the minimum value

Optimization Using Gradient Descent

- Example:
  - We can see (both from the figure and by plugging x_0 into gradient

$$\nabla f\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right) = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}^\top \begin{bmatrix} 2 & 1 \\ 1 & 20 \end{bmatrix} - \begin{bmatrix} 5 \\ 3 \end{bmatrix}^\top$$

with $\gamma = 0{,}085$) that the negative gradient at x_0 points north and east, leading to $\boldsymbol{x_1 = [1.98, 1.21]^T}$. epeating that argument gives us $\boldsymbol{x_2 = [-1.32, -0.42]^T}$, and so on.



Optimization Using Gradient Descent

- Step-size (also called the learning rate)
  - Choosing a good step-size is important in gradient descent.
    - If the step-size is too small, gradient descent can be slow.
    - Step-size is chosen too large, gradient descent can overshoot, fail to converge, or even diverge.
  - Adaptive gradient methods rescale the step-size at each iteration, depending on local properties of the function:
    - When the function value increases after a gradient step, the step-size was too large. Undo the step and decrease the step-size.
    - When the function value decreases the step could have been larger. Try to increase the step-size

Gradient Descent With Momentum

- The convergence of gradient descent may be very slow if the curvature of the optimization surface is such that
  - There are regions that are poorly scaled.
  - The curvature is such that the gradient descent steps hops between the walls of the valley and approaches the optimum in small steps
- Gradient descent with momentum is a method that introduces an additional term to remember what happened in the previous iteration.
  - This memory dampens oscillations and smoothes out the gradient updates.
  - Continuing the ball analogy, the momentum term emulates the phenomenon of a heavy ball that is reluctant to change directions

Gradient Descent With Momentum

- The idea is to have a gradient update with memory to implement a moving average.
  - The momentum-based method remembers the update Δx_i at each iteration i and determines the next update as a linear combination of the current and previous gradients

$$\boldsymbol{x}_{i+1} = \boldsymbol{x}_i - \gamma_i((\nabla f)(\boldsymbol{x}_i))^\top + \alpha\Delta\boldsymbol{x}_i$$
$$\Delta\boldsymbol{x}_i = \boldsymbol{x}_i - \boldsymbol{x}_{i-1} = \alpha\Delta\boldsymbol{x}_{i-1} - \gamma_{i-1}((\nabla f)(\boldsymbol{x}_{i-1}))^\top$$

where $\alpha \in [0,1]$

  - Sometimes we will only know the gradient approximately
    - The momentum term is useful since it averages out different noisy estimates of the gradient
    - One particularly useful way to obtain an approximate gradient is by using a stochastic approximation

Stochastic Gradient Descent

- Computing the gradient can be very time consuming
  - However, often it is possible to find a “cheap” approximation of the gradient.
  - Approximating the gradient is still useful as long as it points in roughly the same direction as the true gradient
  - **Stochastic gradient descent** (often shortened as SGD) is a stochastic approximation of the gradient descent method for minimizing an objective function that is written as a sum of differentiable functions.
    - The word stochastic here refers to the fact that we acknowledge that we do not know the gradient precisely, but instead only know a noisy approximation to it.
    - By constraining the probability distribution of the approximate gradients, we can still theoretically guarantee that SGD will converge.

Stochastic Gradient Descent

- Given $n = 1, \dots, N$ data points, we often consider objective functions that are the sum of the losses L_n incurred by each example n. In mathematical notation, we have the form:

$$L(\boldsymbol{\theta}) = \sum_{n=1}^{N} L_n(\boldsymbol{\theta})$$

where is the vector of parameters of interest, i.e., we want to find that minimizes L.

  - An example from regression is the negative loglikelihood, which is expressed as a sum over log-likelihoods of individual examples so that

$$L(\boldsymbol{\theta}) = -\sum_{n=1}^{N} \log p(y_n | \boldsymbol{x}_n, \boldsymbol{\theta})$$

where $x_n \in \mathbb{R}^D$ are the training inputs, y_n are the training targets, and θ are the parameters of the regression model.

Stochastic Gradient Descent

- Standard gradient descent is a "batch" optimization method, i.e., optimization is performed using the full training set by updating the vector of parameters according to

$$\theta_{i+1} = \theta_i - \gamma_i (\nabla L(\theta_i))^\top = \theta_i - \gamma_i \sum_{n=1}^{N} (\nabla L_n(\theta_i))^\top$$

for a suitable step-size parameter γ_i.

- Evaluating the sum gradient may require expensive evaluations of the gradients from all individual functions L_n
  - When the training set is enormous and/or no simple formulas exist, evaluating the sums of gradients becomes very expensive

Stochastic Gradient Descent

- Consider the term $\Sigma_{n=1}^{N}(\nabla L_n(\theta_i))$ we can reduce the amount of computation by taking a sum over a smaller set of L_n
  - In contrast to batch gradient descent, which uses all L_n for $n = 1, \ldots, N$, we randomly choose a subset of L_n for mini-batch gradient descent.
  - In the extreme case, we randomly select only a single L_n to estimate the gradient. The key insight about why taking a subset of data is sensible is to realize that for gradient descent to converge
    - We only require that the gradient is an unbiased estimate of the true gradient.
    - The term $\Sigma_{n=1}^{N}(\nabla L_n(\theta_i))$ is an empirical estimate of the expected value of the gradient.
    - Any other unbiased empirical estimate of the expected value, for example using any subsample of the data, would suffice for convergence of gradient descent.

Stochastic Gradient Descent

- When the learning rate decreases at an appropriate rate, and subject to relatively mild assumptions, stochastic gradient descent converges almost surely to local minimum.
- Why should one consider using an approximate gradient?
  - A major reason is practical implementation constraints, such as the size of central processing unit (CPU)/graphics processing unit (GPU) memory or limits on computational time.
    - We can think of the size of the subset used to estimate the gradient in the same way that we thought of the size of a sample when estimating empirical means

Stochastic Gradient Descent

- Why should one consider using an approximate gradient?
  - Large mini-batch sizes will provide accurate estimates of the gradient, reducing the variance in the parameter update.
  - Furthermore, large mini-batches take advantage of highly optimized matrix operations in vectorized implementations of the cost and gradient.
  - The reduction in variance leads to more stable convergence, but each gradient calculation will be more expensive.

Stochastic Gradient Descent

- Why should one consider using an approximate gradient?
  - In contrast, small mini-batches are quick to estimate.
    - If we keep the mini-batch size small, the noise in our gradient estimate will allow us to get out of some bad local optima, which we may otherwise get stuck in.
  - In machine learning, optimization methods are used for training by minimizing an objective function on the training data, but the overall goal is to improve generalization performance
  - Since the goal in machine learning does not necessarily need a precise estimate of the minimum of the objective function, approximate gradients using mini-batch approaches have been widely used.
  - Stochastic gradient descent is very effective in large-scale machine learning problems

Constrained Optimization and Lagrange Multipliers

- We considered the problem of solving for the minimum of a function

$$\min_x f(x)$$

where $f : \mathbb{R}^d \longrightarrow \mathbb{R}$

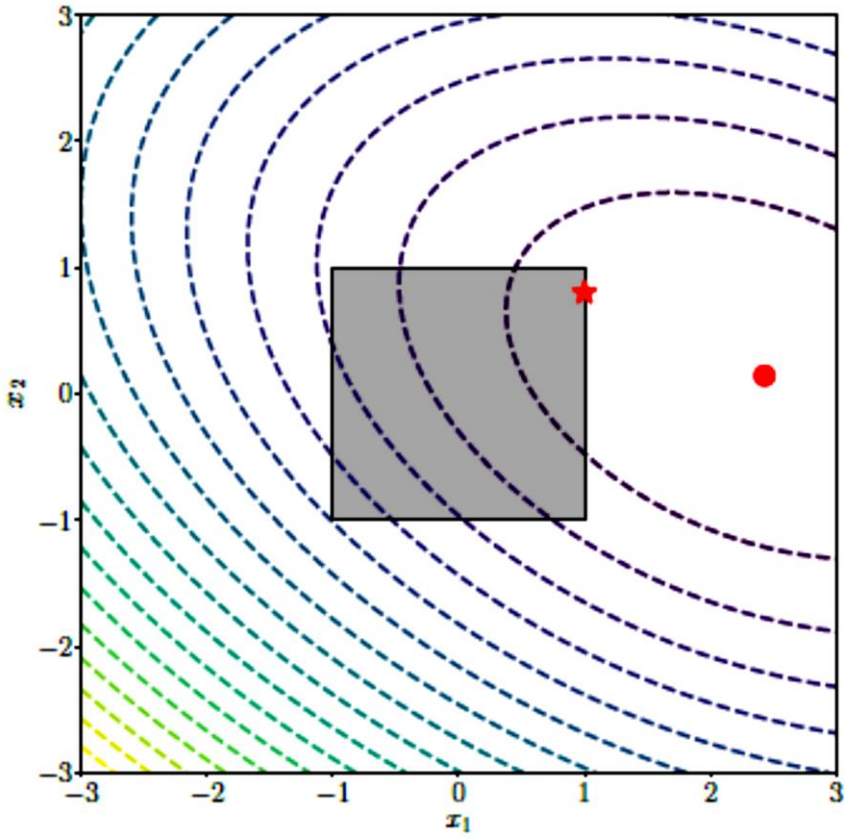
- We have additional constraints. That is, for real-valued functions $g_i: \mathbb{R}^d \longrightarrow \mathbb{R}$ for $i = 1, \ldots, m$, we consider the constrained

$$\min_x f(x)$$

subject to $g_i \leq 0$ for all $i = 1, \ldots, m$

Constrained Optimization and Lagrange Multipliers

- Illustration of constrained optimization
  - The unconstrained problem (indicated by the contour lines) has a minimum on the right side (indicated by the circle).
  - The box constraints ($-1 \leq x \leq 1$ and $-1 \leq y \leq 1$) require that the optimal solution is within the box, resulting in an optimal value indicated by the star.



Constrained Optimization and Lagrange Multipliers

- Way of converting the constrained problem

$$\min_x f(x)$$

subject to $g_i \leq 0$ for all $i = 1, \ldots, m$

into an unconstrained problem using Lagrange multipliers

$$\begin{aligned}\mathfrak{L}(\boldsymbol{x}, \boldsymbol{\lambda}) &= f(\boldsymbol{x}) + \sum_{i=1}^{m} \lambda_i g_i(\boldsymbol{x}) \\ &= f(\boldsymbol{x}) + \boldsymbol{\lambda}^{\mathsf{T}} \boldsymbol{g}(\boldsymbol{x}),\end{aligned}$$

- We have concatenated all constraints $g_i(x)$ into a vector $\boldsymbol{g}(x)$
- All the Lagrange multipliers into a vector $\lambda \in \mathbb{R}^m$.

Constrained Optimization and Lagrange Multipliers

- Lagrangian duality
  - In general, duality in optimization is the idea of converting an optimization problem in one set of variables x (called the primal variables), into another optimization problem in a different set of variables λ (called the dual variables).
  - Problem $\min_x f(x)$, subject to $g_i \leq 0$ for all $i = 1, \ldots, m$, is known as the *primal problem*, corresponding to the primal variables x.
    - The associated *Lagrangian dual problem* is given by

$$\begin{aligned} \max_{\boldsymbol{\lambda} \in \mathbf{R}^m} \quad & \mathfrak{D}(\boldsymbol{\lambda}) \\ \text{subject to} \quad & \boldsymbol{\lambda} \geqslant \mathbf{0}, \end{aligned}$$

where λ are the dual variables and $\mathfrak{D} = \boldsymbol{min}_{\boldsymbol{x} \in \mathbb{R}^d} \boldsymbol{\mathcal{L}}(\boldsymbol{x}, \boldsymbol{\lambda})$

Constrained Optimization and Lagrange Multipliers

- Equality Constraints. Consider the problem $\min_x f(x)$ subject to $g_i \leq 0$ for all $i = 1, \ldots, m$, with additional equality constraints:

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & f(\boldsymbol{x}) \\ \text{subject to} \quad & g_i(\boldsymbol{x}) \leqslant 0 \quad \text{for all} \quad i = 1, \ldots, m \\ & h_j(\boldsymbol{x}) = 0 \quad \text{for all} \quad j = 1, \ldots, n. \end{aligned}$$

  - We can model equality constraints by replacing them with two inequality constraints.
    - That is for each equality constraint $h_j(x) = 0$ we equivalently replace it by two constraints $h_j(x) \leq 0$ and $h_j(x) \geq 0$. It turns out that the resulting Lagrange multipliers are then unconstrained
      - We constrain the Lagrange multipliers corresponding to the inequality constraints to be non-negative,
      - Leave the Lagrange multipliers corresponding to the equality constraints unconstrained.

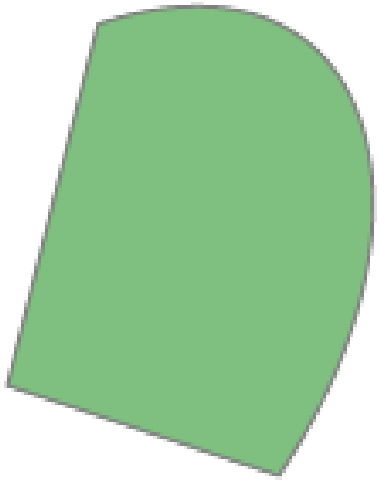
Convex Optimization

- We focus our attention of a particularly useful class of optimization problems, where we can guarantee global optimality.
  - When $f(x)$is a convex function, and when the constraints involving $g(\cdot)$ and $h(\cdot)$ are convex sets, this is called a convex optimization problem
  - A set C is a convex set if for any $x, y \in C$ and for any scalar θ with $0 \leq \theta \leq 1$, we have

$$\theta x + (1-\theta)y \in C$$

Convex Optimization

- Convex sets are sets such that a straight line connecting any two elements of the set lie inside the set



Convex set

Non-convex set

Convex Optimization

- Convex functions are functions such that a straight line between any two points of the function lie above the function
  - Let function $f : \mathbb{R}^D \longrightarrow \mathbb{R}$ be a function whose domain is a convex set.
  - The function f is a *convex function* if for all x, y in the domain convex function of f, and for any scalar with $0 \leq \theta \leq 1$, we have

$$f(\theta x + (1-\theta)y) \leqslant \theta f(x) + (1-\theta) f(y)$$

  - **A *concave function* is the negative of a convex function.**
  - The inequality is sometimes called *Jensen's inequality*
    - In fact, a whole class of inequalities for taking nonnegative weighted sums of convex functions are all called Jensen's inequality.

Convex Optimization

- In summary, a constrained optimization problem is called a convex optimization problem if

$$\begin{aligned} &\min_{\boldsymbol{x}} f(\boldsymbol{x}) \\ &\text{subject to } g_i(\boldsymbol{x}) \leqslant 0 \quad \text{for all} \quad i = 1, \ldots, m \\ &\qquad\qquad h_j(\boldsymbol{x}) = 0 \quad \text{for all} \quad j = 1, \ldots, n\,, \end{aligned}$$

where all functions $f(x)$ and $g_i(x)$ are convex functions, and all $h_j(x)$ =0 are convex sets

Optimizations

- Introduction
- **Linear Programming (simplex method, M-Method)**
- Simulated Annealing Methods
- Programming nonlinear

Linear Programming

- Consider the special case when all the preceding functions are linear, i.e.,

$$\min_{\boldsymbol{x} \in \mathbf{R}^d} \quad \boldsymbol{c}^\top \boldsymbol{x}$$

$$\text{subject to} \quad \boldsymbol{A}\boldsymbol{x} \leqslant \boldsymbol{b}$$

where $A \in \mathbb{R}^{m \times d}$ and $b \in \mathbb{R}^m$. It has d variables and m linear constraints

Linear Programming

- The Lagrangian is given by

$$\mathfrak{L}(\boldsymbol{x}, \boldsymbol{\lambda}) = \boldsymbol{c}^\top \boldsymbol{x} + \boldsymbol{\lambda}^\top (\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b})$$

where $\lambda \in \mathbb{R}^m$ is the vector of non-negative Lagrange multipliers.

- Rearranging the terms corresponding to x yields

$$\mathfrak{L}(\boldsymbol{x}, \boldsymbol{\lambda}) = (\boldsymbol{c} + \boldsymbol{A}^\top \boldsymbol{\lambda})^\top \boldsymbol{x} - \boldsymbol{\lambda}^\top \boldsymbol{b}$$

- Taking the derivative of $\mathcal{L}(\mathrm{x}, \lambda)$ with respect to x and setting it to zero gives us

$$\boldsymbol{c} + \boldsymbol{A}^\top \boldsymbol{\lambda} = \boldsymbol{0}$$

Linear Programming-Graphical Solution

- Determine or draw the feasible set F (The feasible set F is the set of all the points satisfying the constraint inequations/ equations
  - If $F = \emptyset$: the problem has no optimal solution: it is said to be infeasible
- The problem is called unbounded and has no optimal solution provided that

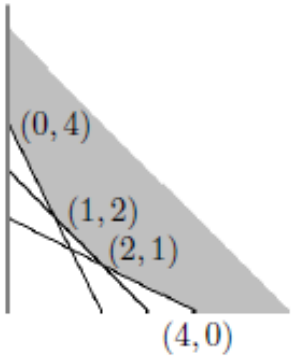
$F \neq \emptyset$ and the objective function on F has

  - Arbitrarily large positive values for a maximization problem, or
  - Arbitrarily large negative values for a minimization problem.
- A problem is called bounded provided that it is neither infeasible nor unbounded In this case, an optimal solution exists.
  - Determine all the vertices of F and values of the objective function at the vertices.
  - Choose the vertex of F producing the maximal or minimal value of the objective function.

Linear Programming-Graphical Solution

- The feasible set that is shaded in Figure is unbounded but $f(x) = 0$ is bounded below, so $f(x)$ has a minimum.
  - The vertices are $(4;\ 0), (2;\ 1), (1;\ 2)$, and $(0;\ 4)$, with values $f(4,0) = 12$, $f(2,1) = 8, f(1,2) = 7$and $f(0,4) = 12$. Therefore, the minimum value is 7 which is attained at $(1,2)$

Minimize: $3x_1 + 2x_2$,
Subject to: $2x_1 + x_2 \geq 4$,
$x_1 + x_2 \geq 3$,
$x_1 + 2x_2 \geq 4$,
$x_1 \geq 0$, and $x_2 \geq 0$.



Linear Programming - Simplex Method

- The graphical solution method is not a practical algorithm for most problems
  - The number of vertices for a linear program grows very fast as the number of variables and constraints increase.
  - The first step is to add more variables into the standard maximization linear programming problem to make all the inequalities of the form $x_i \geq 0$ for some variables x_i

Linear Programming - Simplex Method

- Simplex:
  - A linear-programming algorithm that can solve problems having more than two decision variables
  - The simplex technique involves generating a series of solutions in tabular form, called tableaus.
    - By inspecting the bottom row of each tableau, one can immediately tell if it represents the optimal solution.
    - Each tableau corresponds to a corner point of the feasible solution space. The first tableau corresponds to the origin.
    - Subsequent tableaus are developed by shifting to an adjacent corner point in the direction that yields the highest (smallest) rate of profit (cost). This process continues as long as a positive (negative) rate of profit (cost) exists

Linear Programming - Simplex Method

- The simplex method in tabular form
  - Steps
    - Initialization
    - Test for optimality
    - Iteration

Linear Programming - Simplex Method

- A basic solution is an augmented corner point solution.
- A basic solution has the following properties:
  - Each variable is designated as either a nonbasic variable or a basic variable.
  - The number of basic variables equals the number of functional constraints. Therefore, the number of nonbasic variables equals the total number of variables minus the number of functional constraints.
  - The nonbasic variables are set equal to zero.
  - The values of the basic variables are obtained as simultaneous solution of the system of equations (functional constraints in augmented form). The set of basic variables are called “basis”
  - If the basic variables satisfy the nonnegativity constraints, the basic solution is a Basic Feasible (BF) solution.

Linear Programming - Simplex Method

- Steps
  - Initialization
    - Transform all the constraints to equality by introducing slack, surplus, and artificial variables

Constraint type	Variable to be added
≤	+ slack (s)
≥	- Surplus (s) + artificial (A)
=	+ Artificial (A)

Linear Programming - Simplex Method

- Steps
  - Initialization
    - Transform all the constraints to equality by introducing slack, surplus, and artificial variables

Original Form

Maximize $Z = 3x_1 + 5x_2$

subject to

$x_1 \leq 4$

$2x_2 \leq 12$

$3x_1 + 2x_2 \leq 18$

$x_1, x_2 \geq 0$

***Augmented** Form*

Maximize $Z = 3x_1 + 5x_2$

subject to

$x_1 + \mathbf{s_1} = 4$

$2x_2 + \mathbf{s_2} = 12$

$3x_1 + 2x_2 + \mathbf{s_3} = 18$

$x_1, x_2 \geq 0$

Linear Programming - Simplex Method

- Steps
  - Initialization
    - Construct the initial simplex tableau

Basic variable	X_1	...	X_n	S_1	.....	S_n	A_1	....	A_n	RHS
S	Coefficient of the constraints									b_1
…										…
A										b_m
Z	Objective function coefficient In different signs									Z value

Linear Programming - Simplex Method

- Steps
  - Test for optimality:
    - Case 1: Maximization problem the current Basic feasible (BF) solution is optimal if every coefficient in the objective function row is **nonnegative**
    - Case 2: Minimization problem the current BF solution is optimal if every coefficient in the objective function row is **nonpositive**

Linear Programming - Simplex Method

- Steps
  - Iteration
    - **Step 1**: determine the entering basic variable by selecting the variable (automatically a nonbasic variable)
      - $\boldsymbol{with\ the\ most\ negative\ value\ (in\ case\ of\ maximization)}$ or $\boldsymbol{with\ the\ most\ positive\ (in\ case\ of\ minimization)}$ $in\ the\ last\ row\ (Z\ row)$.
      - Put a box around the column below this variable, and call it the "**pivot column**"

Linear Programming - Simplex Method

- Steps
  - Iteration
    - **Step 2:** Determine the leaving basic variable by the minimum ratio test as following:
      - Pick out each coefficient in the pivot column that is strictly positive (>0)
      - Divide each of these coefficients into the right hand side entry for the same row
      - Identify the row that has the smallest of these ratios
      - The basic variable for that row is the leaving variable, so replace that variable by the entering variable in the basic variable column of the next simplex tableau. Put a box around this row and call it the "**pivot row**"

Linear Programming - Simplex Method

- Steps
  - Iteration
    - **Step 3**:
      - Solve for the new BF solution by using elementary row operations (multiply or divide a row by a nonzero constant; add or subtract a multiple of one row to another row) to construct a new simplex tableau, and then return to the optimality test. The specific elementary row operations are:
      - Divide the pivot row by the “pivot number” (the number in the intersection of the pivot row and pivot column)
      - For each other row that has a negative coefficient in the pivot column, add to this row the product of the absolute value of this coefficient and the new pivot row.
      - For each other row that has a positive coefficient in the pivot column, subtract from this row the product of the absolute value of this coefficient and the new pivot row.

Linear Programming - Simplex Method

- Example:
  - Maximize $Z = 3x_1 + 5x_2$ *Subject to*

$$x1 \leq 4$$

$$2\,x_2 \leq 12$$

$$3x_1 + 2x_2 \leq 18$$

$$x_1, x_2 \geq 0$$

Linear Programming - Simplex Method

- Example:
  - Standard form (Sometimes it is called the augmented form of the problem because the original form has been augmented by some supplementary variables needed to apply the simplex method)

$$Z - 3x_1 - 5x_2 = 0 \quad Subject\ to$$
$$x1 + s_1 = 4$$
$$2\,x_2 + s_2 = 12$$
$$3x_1 + 2x_2 + s_3 = 18$$
$$x_1, x_2, s_1, s_2, s_3 \geq 0$$

Linear Programming - Simplex Method

- Example: Initial tableau

Entering variable

Basic variable	x_1	x_2	S_1	S_2	S_3	RHS
S_1	1	0	1	0	0	**4**
S_2	0	2	0	1	0	**12**
S_3	3	2	0	0	1	**18**
Z	**-3**	**-5**	**0**	**0**	**0**	**0**

Leaving variable

Pivot column

Pivot number

Pivot row

Linear Programming - Simplex Method

- The basic feasible solution at the initial tableau is $(0, 0, 4, 12, 18)$: $x_1 = 0, x_2 = 0, S_1 = 4, S_2 = 12, S_3 = 18$, and $Z = 0$

where S_1, S_2, and S_3 are basic variables; x_1 and x_2 are nonbasic variables

  - The solution at the initial tableau is associated to the origin point at which all the decision variables are zero.

- Optimality test: By investigating the last row of the initial tableau, we find that there are some negative numbers. Therefore, the current solution is not optimal

Linear Programming - Simplex Method

- Iteration:
  - Step 1: Determine the entering variable by selecting the variable with the most negative in the last row.
  - From the initial tableau, in the last row (Z row), the coefficient of x_1 is -3 and the coefficient of x_2 is -5; therefore, the most negative is -5. consequently, x_2 is the entering variable.
  - x_2 is surrounded by a box and it is called the pivot column

Linear Programming - Simplex Method

- Iteration:
  - Step 2: Determining the leaving variable by using the minimum ratio test as following:

Basic variable	x_1	x_2	S_1	S_2	S_3	RHS	Ratio
S_1	1	0	1	0	0	**4**	∞
S_2	0	2	0	1	0	**12**	6
S_3	3	2	0	0	1	**18**	**9**
Z	**-3**	**-5**	**0**	**0**	**0**	**0**	

Smallest ratio

Pivot row

Linear Programming - Simplex Method

- Iteration:
  - Step 2: Determining the leaving variable by using the minimum ratio test as following:

Basic variable	x_1	x_2	S_1	S_2	S_3	RHS	Ratio
S_1	1	0	1	0	0	**4**	∞
S_2	0	2	0	1	0	**12**	6
S_3	3	2	0	0	1	**18**	**9**
Z	**-3**	**-5**	**0**	**0**	**0**	**0**	

Smallest ratio

Pivot row

Linear Programming - Simplex Method

- Step 3: solving for the new BF solution by using the eliminatory row operations as following
  - $New\ pivot\ row\ =\ old\ pivot\ row\ \div\ pivot\ number$

Basic variable	x_1	x_2	S_1	S_2	S_3	RHS
S_1						
x_2	0	1	0	1/2	0	**6**
S_3						
Z						

x_2 becomes in the basic variables list instead of S_2

Linear Programming - Simplex Method

- Step 3: For the other row apply this rule:
  - *New row = old row – the coefficient of this row in the pivot column (new pivot row).*

Basic variable	x_1	x_2	S_1	S_2	S_3	RHS
S_1	1	0	1	0	0	**4**
x_2	0	1	0	1/2	0	**6**
S_3	3	2	0	0	1	**18**
Z	**-3**	**-5**	**0**	**0**	**0**	**0**

Basic variable	x_1	x_2	S_1	S_2	S_3	RHS
S_1	1	0	1	0	0	**4**
x_2	0	1	0	1/2	0	**6**
S_3	3	0	0	-1	1	**6**
Z	**-3**	**0**	**0**	**5/2**	**0**	**30**

Linear Programming - Simplex Method

- Iteration: This solution is not optimal, since there is a negative numbers in the last row

Basic variable	x_1	x_2	S_1	S_2	S_3	RHS	Ratio
S_1	1	0	1	0	0	**4**	**4**
x_2	0	1	0	1/2	0	**6**	∞
x_1	3	0	0	-1	1	**6**	**2**
Z	**-3**	**0**	**0**	**5/2**	**0**	**30**	

Linear Programming - Simplex Method

- Iteration: Apply the same rules we will obtain this solution:

Basic variable	x_1	x_2	S_1	S_2	S_3	RHS
S_1	1	0	1	0	0	**4**
x_2	0	1	0	1/2	0	**6**
X_1	3	0	0	-1	1	**6**
Z	**-3**	**0**	**0**	**5/2**	**0**	**30**

Basic variable	x_1	x_2	S_1	S_2	S_3	RHS
S_1	0	0	1	1/3	-1/3	**2**
x_2	0	1	0	1/2	0	**6**
X_1	1	0	0	-1/3	1/3	**2**
Z	**0**	**0**	**0**	**3/2**	**0**	**36**

- This solution is optimal; **since there is no negative solution in the last row**: basic variables are $x_1 = 2, x_2 = 6$ and $S_1 = 2$; the nonbasic variables are $S_2 = S_3 = 0, Z = 36$

Linear Programming - Simplex Method

- In any Simplex tableau, the intersection of any basic variable with itself is always one and the rest of the column is zeroes
- In any simplex tableau, the objective function row (Z row) is always in terms of the nonbasic variables. This means that under any basic variable (in any tableau) there is a zero in the Z row. For the non basic there is no condition (it can take any value in this row)
- If there is a zero under one or more nonbasic variables in the last tableau (optimal solution tableau), then there is a multiple optimal solution
- When determining the leaving variable of any tableau, if there is no positive ratio (all the entries in the pivot column are negative and zeroes), then the solution is unbounded.

Linear Programming - Simplex Method

- If there is a tie (more than one variables have the same most negative or positive) in determining the entering variable, choose any variable to be the entering one
- If there is a tie in determining the leaving variable, choose any one to be the leaving variable. In this case a zero will appear in RHS column; therefore, a "cycle" will occur, this means that the value of the objective function will be the same for several iterations
- A Solution that has a basic variable with zero value is called a "degenerate solution"
- If there is no Artificial variables in the problem, there is no room for "infeasible solution"

Linear Programming - Big M method

- Simplex method incase of Artificial variables
  - In order to use the simplex method, a bfs is needed.
  - To remedy the predicament, **artificial variables** are created.
  - In the optimal solution, all artificial variables must be set equal to zero. To accomplish this, in a min Linear Programming , a term MA_i is added to the objective function for each artificial variable A_i .

Linear Programming - Big M method

- M, a very large number, is used to ensure that the values of A_1 and $A_2, \dots, A_n$ will be zero in the final (optimal) tableau as follows:
  - If the objective function is Minimization, then $A_1, A_2, \dots, A_n$ must be added to the RHS of the objective function multiplied by a very large number (M).
  - Example: if the objective function is $Min\ Z = x_1 + x_2$, then the obj. function should be

$$Min\ Z = x_1 + x_2 + MA_1 + MA_2 + \dots + MA_n$$

OR

$$Z - x_1 - x_2 - MA_1 - MA_2 - \dots - MA_n = 0$$

Linear Programming - Big M method

- M, a very large number, is used to ensure that the values of A_1 and $A_2, \dots, A_n$ will be zero in the final (optimal) tableau as follows:
  - If the objective function is Maximization, then $A_1, A_2, \dots, A_n$ must be subtracted from the RHS of the objective function multiplied by a very large number (M).
  - Example: if the objective function is $Min\ Z = x_1 + x_2$, then the obj. function should be

$$Max\ Z = x_1 + x_2 + MA_1 + MA_2 + \dots + MA_n$$

or

$$Z - x_1 - x_2 + MA_1 + MA_2 + \dots + MA_n = 0$$

Linear Programming - Big M method

- Modify the constraints so that the rhs of each constraint is nonnegative. Identify each constraint that is now an = or ≥ constraint.
- Convert each inequality constraint to standard form (add a slack variable for ≤ constraints, add an excess variable for ≥ constraints).
- For each ≥ or = constraint, add artificial variables. Add sign restriction $a_i \geq 0$
- Let M denote a very large positive number.
  - Add (for each artificial variable) Ma_i to min problem objective functions or -Ma_i to max problem objective functions.

Linear Programming - Big M method

- Since each artificial variable will be in the starting basis, all artificial variables must be eliminated from row 0 before beginning the simplex. Remembering M represents a very large number, solve the transformed problem by the simplex.
- If all artificial variables in the optimal solution equal zero, the solution is optimal. If any artificial variables are positive in the optimal solution, the problem is infeasible

Linear Programming - Big M method

- Example

$$Min\ Z\ -2x_1 - 3x_2 = 0 \quad Subject\ to$$

$$\frac{1}{2}x1\ + \frac{1}{4}x_2 \leq 4$$

$$x_1 + 3x_2 \geq 20$$

$$x_1\ + x_2 =\ 10$$

$$x_1\ , x_2, \geq\ 0$$

Linear Programming - Big M method

- Example:
  - Converting:

$$Min\ Z,$$

$$Z - 2x_1 - 3x_2 - MA_1 - MA_2 = 0 \quad Subject\ to$$

$$\frac{1}{2}x1 + \frac{1}{4}x_2 + S_1 = 4$$

$$x_1 + 3x_2 - S_2 + A_1 = 20$$

$$x_1 + x_2 + A_2 = 10$$

$$x_1, x_2, S_1, S_2, A_1, A_2 \geq 0$$

Linear Programming - Big M method

- Initial tableau

Basic variables	x_1	x_2	S_1	S_2	A_1	A_2	RHS
S_1	½	¼	1	0	0	0	4
A_1	1	3	0	-1	1	0	20
A_2	1	1	0	0	0	1	10
Z	-2	-3	0	0	-M	-M	0

- One of the simplex rules is violated, which is the basic variables A_1, and A_2 have a non zero value in the z row; therefore, this violation must be corrected before proceeding in the simplex algorithm as follows.

Linear Programming - Big M method

- To correct this violation before starting the simplex algorithm, the elementary row operations are used as follows:

$$New\ (Z\ row)\ =\ old\ (z\ row)\ \pm\ M\ (A1\ row)\ \pm\ M\ (A2\ row)$$

	x_1	x_2	S_1	S_2	A_1	A_2	RHS
Z (old)	-2	-3	0	0	-M	-M	0
$M(A_1$ old)	M	3M	0	-M	M	0	20M
$M(A_2$ old)	M	M	0	0	0	M	10M
Z	-2+2M	-3+4M	0	-M	0	0	30M

Linear Programming - Big M method

- Initial tableau

Basic variables	x_1	x_2	S_1	S_2	A_1	A_2	RHS
S_1	1/2	1/4	1	0	0	0	4
A_1	1	3	0	-1	1	0	20
A_2	1	1	0	0	0	1	10
Z	2M-2	4M-3	0	-M	0	0	30M

- Since there is a positive value in the last row, this solution is not optimal
  - The entering variable is X2 (it has the most positive value in the last row)
  - The leaving variable is A1 (it has the smallest ratio)

Linear Programming - Big M method

- Iteration

Basic variables	x_1	x_2	S_1	S_2	A_1	A_2	RHS
S_1	1/2	1/4	1	0	0	0	4
A_1	1	3	0	-1	1	0	20
A_2	1	1	0	0	0	1	10
Z	2M-2	4M-3	0	-M	0	0	30M

Basic variables	x_1	x_2	S_1	S_2	A_1	A_2	RHS
S_1	1/2	1/4	1	0	0	0	4
x_2	**1/3**	**1**	**0**	**-1/3**	**1/3**	**0**	**20/3**
A_2	1	1	0	0	0	1	10
Z	2M-2	4M-3	0	-M	0	0	30M

Linear Programming - Big M method

- Iteration

Basic variables	x_1	x_2	S_1	S_2	A_1	A_2	RHS
S_1	1/2	1/4	1	0	0	0	4
x_2	1/3	1	0	-1/3	1/3	0	20/3
A_2	1	1	0	0	0	1	10
Z	2M-2	4M-3	0	-M	0	0	30M

Basic variables	x_1	x_2	S_1	S_2	A_1	A_2	RHS
S_1	5/12	0	1	1/12	-1/12	0	7/3
x_2	**1/3**	**1**	**0**	**-1/3**	**1/3**	**0**	**20/3**
A_2	2/3	0	0	1/3	-1/3	1	10/3
Z	2/3M-1	0	0	1/3M-1	1-4/3M	0	20+10/3M

Linear Programming - Big M method

- Iteration

Basic variables	x_1	x_2	S_1	S_2	A_1	A_2	RHS
S_1	5/12	0	1	1/12	-1/12	0	7/3
x_2	**1/3**	**1**	**0**	**-1/3**	**1/3**	**0**	**20/3**
A_2	2/3	0	0	1/3	-1/3	1	10/3
Z	**2/3M-1**	0	0	1/3M-1	1-4/3M	0	20+10/3M

- Since there is a positive value in the last row, this solution is not optimal
  - The entering variable is X_1 (it has the most positive value in the last row)
  - The leaving variable is A_2 (it has the smallest ratio)

Linear Programming - Big M method

- Iteration

Basic variables	x_1	x_2	S_1	S_2	A_1	A_2	RHS
S_1	5/12	0	1	1/12	-1/12	0	7/3
x_2	**1/3**	**1**	**0**	**-1/3**	**1/3**	**0**	**20/3**
A_2	2/3	0	0	1/3	-1/3	1	10/3
Z	**2/3M-1**	0	0	1/3M-1	1-4/3M	0	20+10/3M

Basic variables	x_1	x_2	S_1	S_2	A_1	A_2	RHS
S_1							
x_2							
A_2	1	0	0	1/2	-1/2	3/2	5
Z							

Linear Programming - Big M method

- Iteration

Basic variables	x_1	x_2	S_1	S_2	A_1	A_2	RHS
S_1	5/12	0	1	1/12	-1/12	0	7/3
x_2	1/3	1	0	-1/3	1/3	0	20/3
A_2	1	0	0	1/2	-1/2	3/2	5
Z	2/3M-1	0	0	1/3M-1	1-4/3M	0	20+10/3M

Basic variables	x_1	x_2	S_1	S_2	A_1	A_2	RHS
S_1	0	0	1	-1/8	1/8	-5/8	1/4
x_2	0	1	0	-1/2	1/2	-1/2	5
A_2	1	0	0	1/2	-1/2	3/2	5
Z	0	0	0	-1/2	½-M	3/2-M	25

Linear Programming - Big M method

- Iteration
  - This solution is optimal, since there is no positive value in the last row. The optimal solution is:
    - $X_1 = 5, X_2 = 5, S_1 = ¼$
    - $A_1 = A_2 = 0 \ and \ Z = 25$

Basic variables	x_1	x_2	S_1	S_2	A_1	A_2	RHS
S_1	0	0	1	-1/8	1/8	-5/8	1/4
$\boldsymbol{x_2}$	0	1	0	-1/2	1/2	-1/2	5
x_1	1	0	0	1/2	-1/2	3/2	5
Z	0	0	0	-1/2	½-M	3/2-M	25

Linear Programming - Big M method

- In the final tableau,
  - If one or more artificial variables $(A_1, A_2, \dots)$ still basic and has a nonzero value, then the problem has an infeasible solution.
  - If there is a zero under one or more nonbasic variables in the last tableau (optimal solution tableau), then there is a multiple optimal solution.
  - When determining the leaving variable of any tableau, if there is no positive ratio (all the entries in the pivot column are negative and zeroes), then the solution is unbounded.

Optimizations

- Introduction
- Linear Programming (simplex method, M-Method)
- **Simulated Annealing Methods**
- Programming nonlinear

Simulated Annealing Methods

- Simulated Annealing is a stochastic optimization method that derives its name from the annealing process used to re-crystallize metals
- The method of simulated annealing is a technique that has attracted significant attention as suitable for optimization problems of large scale, especially ones where a desired global extremum is hidden among many, poorer, local extrema.
- Comes under the category of evolutionary techniques of optimization

Simulated Annealing Methods

- Annealing
  - A heat process whereby a metal is heated to a specific temperature and then allowed to cool slowly. This softens the metal which means it can be cut and shaped more easily
  - Initially when the metal is heated to high temperatures, the atoms have lots of space to move about
  - Slowly when the temperature is reduced the movement of free atoms are slowly reduced and finally the metals crystallize themselves

Simulated Annealing Methods

- Relation between annealing and simulated annealing
  - Simulated annealing is analogous to this annealing process
  - Initially the search area is more, there input parameters are searched in more random space and slow with each iteration this space reduces.
  - This helps in achieving global optimized value, although it takes much more time for optimizing

Simulated Annealing Methods

- Annealing
  - Energy in thermodynamic system
  - High-mobility atoms are trying to orient themselves with other nonlocal atoms and the energy state can occasionally go up.
  - Low-mobility atoms can only orient themselves with local atoms and the energy state is not likely to go up again.
- Simulated Annealing
  - Value of objective function
  - At high temperatures, SA allows f evaluations at faraway points and it is likely to accept a new point.
  - At low temperatures, SA evaluates the objective function only at local points and the likelihood of it accepting a new point with higher energy is much lower.

Simulated Annealing Methods

- Motivated by the physical annealing process
  - an analogy to the statistical mechanics of annealing in solids
- Material is heated and slowly cooled into a uniform structure
  - Heat the solid state metal to a high temperature and cool it down very slowly according to a specific schedule
- Simulated annealing mimics this process

Simulated Annealing Methods

- The first SA algorithm was developed in 1953 (Metropolis)
  - It is a technique for combinatorial optimization problems, such as minimizing functions of very many variables.
  - Because many real world design problems can be cast in the form of such optimization problems, here is intense interest in general techniques for their solution.
- SA techniques use an analogous set of controlled cooling operations for nonphysical optimization problems, transforming unordered solution into a highly optimized, desired solution

Simulated Annealing Methods

Physical System		Optimization Problem
State (configuration)	→	Solution
Energy	→	Cost function
Ground State	→	Optimal solution
Rapid Quenching	→	Iteration improvement
Careful Annealing	→	Simulated annealing

- Metal ←→ Problem
- Energy State ←→ Cost Function
- Temperature ←→ Control Parameter
- A completely ordered crystalline structure ←→ the optimal solution for the problem

Simulated Annealing Methods

- To apply simulated annealing with optimization purposes we require the following:
  - A successor function that returns a “close” neighboring solution given the actual one. This will work as the “disturbance” for the particles of the system.
  - A target function to optimize that depends on the current state of the system. This function will work as the energy of the system.
- The search is started with a randomized state
  - In a polling loop we will move to neighboring states always accepting the moves that decrease the energy while only accepting bad moves accordingly to a probability distribution dependent on the “temperature” of the system

Simulated Annealing Methods

- Decrease the temperature slowly, accepting less bad moves at each temperature level until at very low temperatures the algorithm becomes a greedy hill-climbing algorithm
- The distribution used to decide if we accept a bad movement is know as Boltzman distribution

$$P(E) = \frac{1}{Z(T)} \exp(-\frac{E}{kT})$$

  - This distribution expresses the idea that a system in thermal equilibrium at temperature T has its energy probabilistically distributed among all different energy states E: ***k*** **is a constant known as Boltzmann's constant, Z(T) Normalization factor- temperature dependan (not important)**

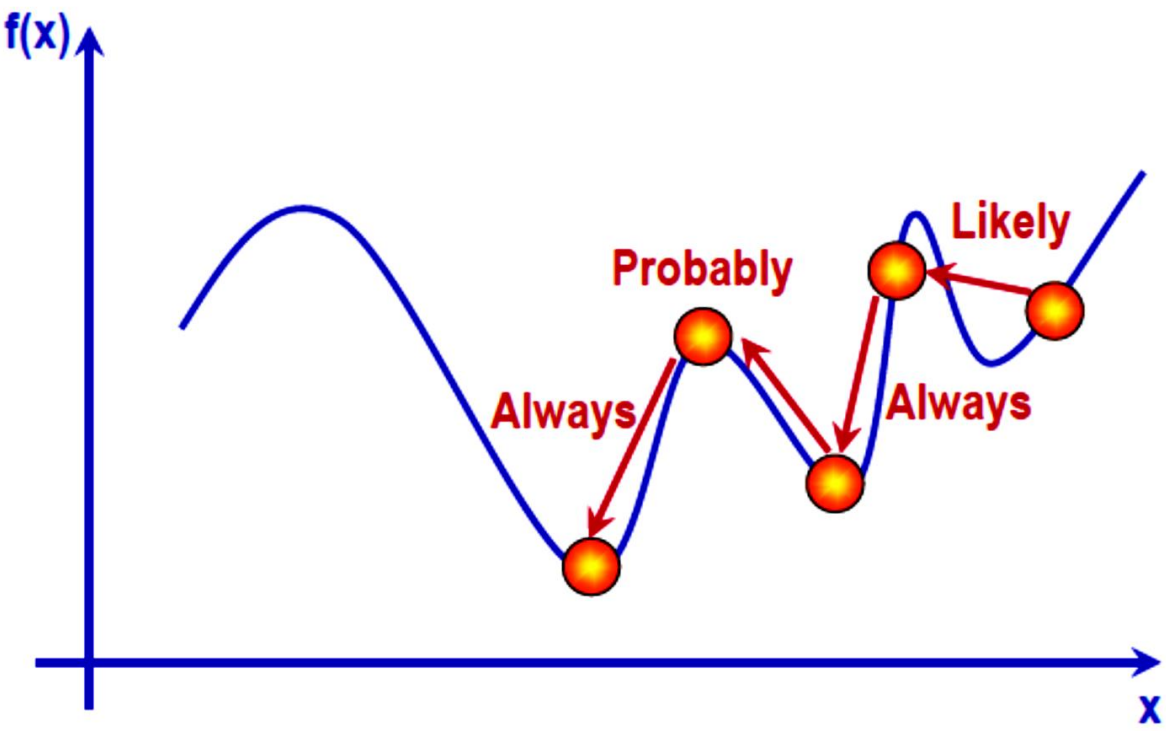
Simulated Annealing Methods

- For combinatorial optimization problems
  - Objective function: $E = f(x)$, where each x is viewed as a point in an input space
  - The task of SA is to sample the input space effectively to find an x that minimizes E.

Simulated Annealing Methods

- A simulated thermodynamic system was assumed to change its configuration from energy E_1 to energy E_2 with probability $p = \exp[-(E_2 - E_1)/kT]$
  - Notice that if $E_2 < E_1$, this probability is greater than unity; in such cases the change is arbitrarily assigned a probability $p = 1$, i.e., the system *always* took such an option.
  - This general scheme, of always taking a downhill step while *sometimes* taking an uphill step, has come to be known as the Metropolis algorithm

Simulated Annealing Methods

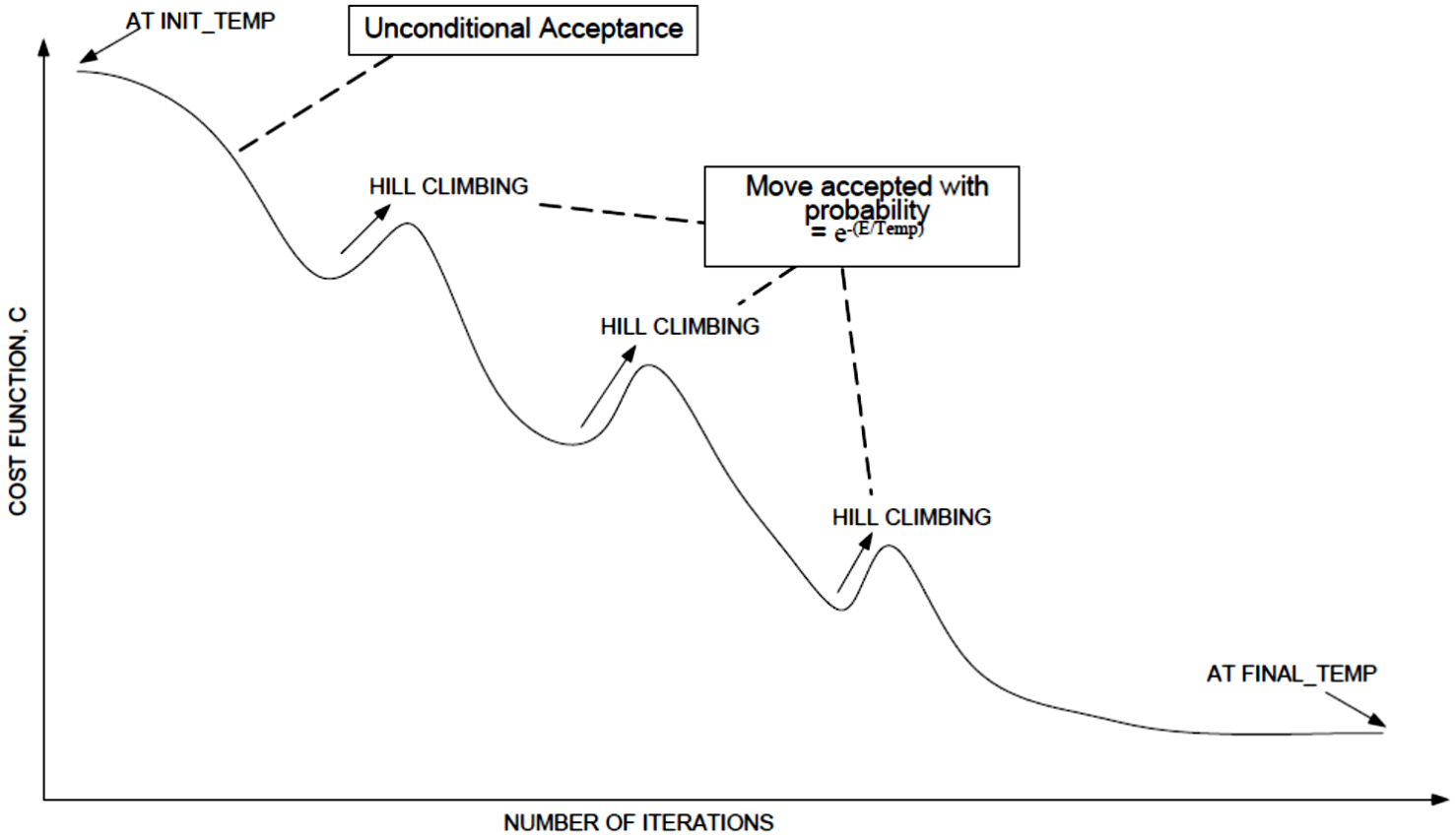


Simulated annealing accept/reject new solution candidate based on probability

Simulated Annealing Methods

- **Step 1:** Initialize – Start with a random initial placement. Initialize a very high "temperature"
- **Step 2:** Move – Perturb the placement through a defined move
- **Step 3:** Calculate score – calculate the change in the score due to the move made
- **Step 4:** Choose – Depending on the change in score, accept or reject the move. The problem of acceptance depends on the current "temperature"
- **Step 5:** Update and repeat– Update the temperature value by lowering the temperature. Go back to Step 2.

Simulated Annealing Methods



Simulated Annealing Methods

```
Algorithm SIMULATED-ANNEALING
Begin
   temp = INIT-TEMP;
   place = INIT-PLACEMENT;
   while (temp > FINAL-TEMP) do
         while (inner_loop_criterion = FALSE) do
                  new_place = PERTURB(place);
                  ΔC = COST(new_place)  - COST(place);
                  if (ΔC < 0) then
                           place = new_place;
                  else if (RANDOM(0,1) > e^-(ΔC/temp)) then
                           place = new_place;
         temp = SCHEDULE(temp);
End
```

Simulated Annealing Methods

- Step 1: start from an initial point $X = X_0$ & $K = 0$
- Step 2: evaluate cost function $E = f(X_K)$
- Step 3: randomly move from X_K to a new solution X_{K+1}
- Step 4: if $f(X_{K+1}) < E$, then
  - Accept new solution
  - $X = X_{K+1}$ & $E = f(X_{K+1})$
- End if

Similar to local optimization

- Step 5: if $f(X_{K+1}) \geq E$, then
  - Accept new solution with certain probability
  - $X = X_{K+1}$ & $E = f(X_{K+1})$ iff rand(1) < ε
- End if

Help to get out of local minimum

- Step 6: $K = K + 1$ & go to Step 2

Simulated Annealing Methods

- **Accept/reject new solution with the probability ε**
  - If $f(X_{K+1}) \geq E$, then
    - Accept new solution with certain probability
    - $X = X_{K+1}$ & $E = f(X_{K+1})$ iff rand(1) < ε
  - End if
- **Option 1**
  - Constant probability, i.e., ε = 0.1
- **Option 2 (better than Option 1)**
  - Dynamically varying probability, i.e., decreasing over time

Simulated Annealing Methods

- **Accept/reject new solution with the probability ε**
  - If $f(X_{K+1}) \geq E$, then
    - Accept new solution with certain probability
    - $X = X_{K+1}$ & $E = f(X_{K+1})$ iff rand(1) < ε
  - End if

- **Use Boltzmann distribution to determine the probability ε**

$$\varepsilon = \exp\left[-\frac{f(X_{K+1}) - E}{T_{K+1}}\right]$$

  - T_{K+1} is a "temperature" parameter that gradually decreases
  - E.g., $T_{K+1} = \alpha \cdot T_K$ where $\alpha < 1$

Simulated Annealing Methods

- Simulated annealing does not guarantee global optimum
  - However, it tries to avoid a large number of local minima
  - Therefore, it often yields a better solution than local optimization
- Simulated annealing is not deterministic
  - Whether accept or reject a new solution is random
  - We can get different answers from multiple runs
- Simulated annealing is more expensive than local optimization
  - It is the price we must pay to achieve a better optimal solution

Simulated Annealing Methods

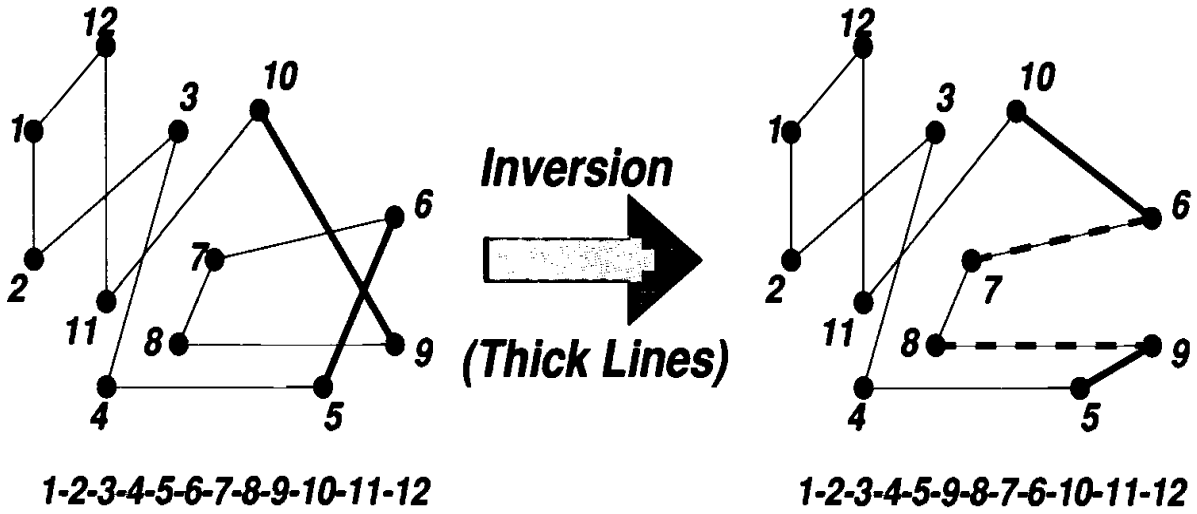
- Inefficient global search: In order to assure a final convergence effectively, the moving methods with relatively small changes should be used, so the global search within a short runtime is quite limited.
- Informational waste: It does not use the information of past experience, including past solutions and past moves.
- Simulated annealing has been used to solve many practical engineering problems

Simulated Annealing Methods

- Travelling Salesman Problem
  - In a typical TSP problem there are 'n' cities, and the distance (or cost) between all pairs of these cities is an $n\ x\ n$ distance (or cost) matrix D, where the element d_{ij} represents the distance (cost) of traveling from city i to city j
  - The problem is to find a closed tour in which each city, except for starting one, is visited exactly once, such that the total length (cost) is minimized.
  - combinatorial optimization; it belongs to a class of problems known as NP-complete

Simulated Annealing Methods

- Travelling Salesman Problem
  - Inversion: Remove two edges from the tour and replace them to make it another legal tour.

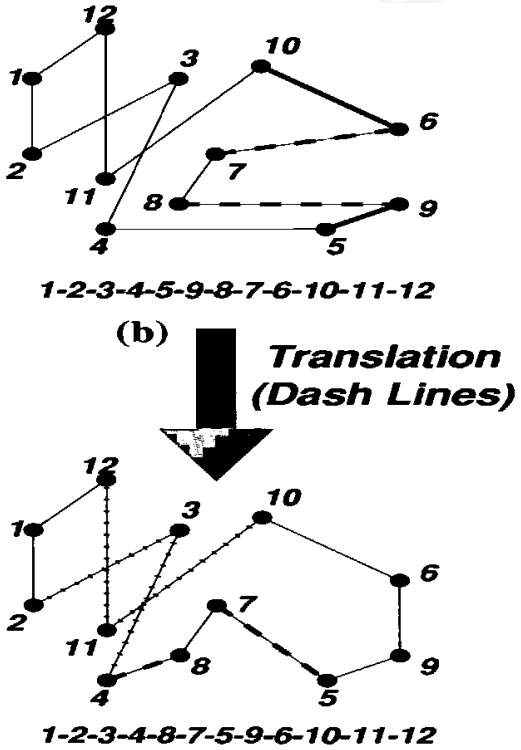


1-2-3-4-5-6-7-8-9-10-11-12

1-2-3-4-5-9-8-7-6-10-11-12

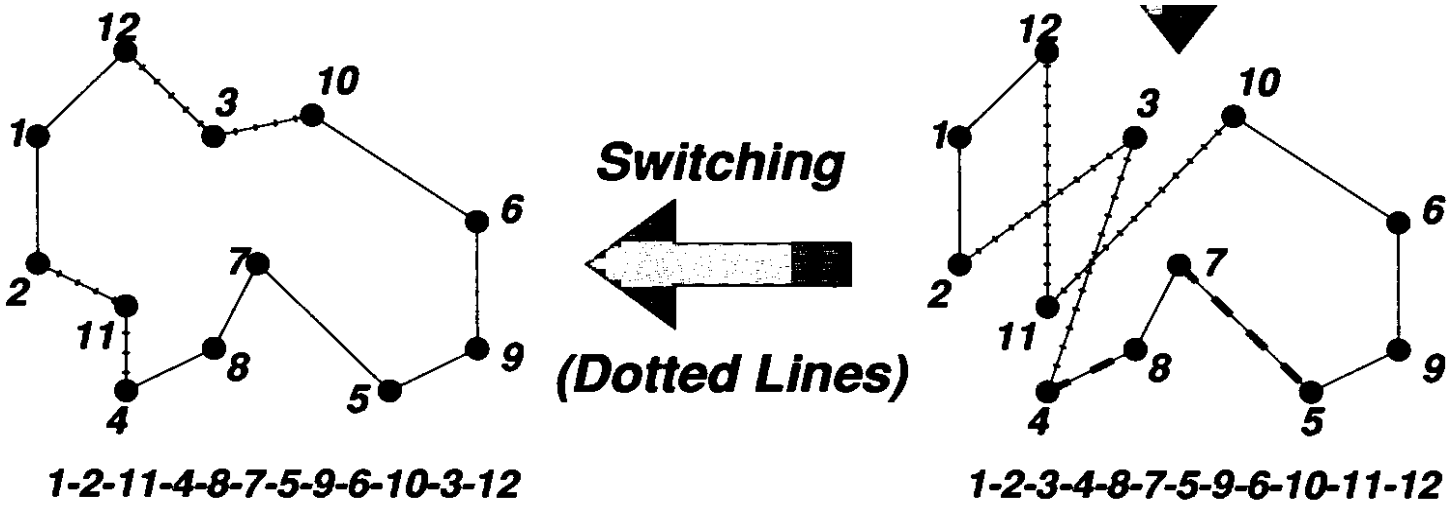
Simulated Annealing Methods

- Travelling Salesman Problem
  - Translation Remove a section (8-7) of the tour and then replace it in between two randomly selected consecutive cities 4 and 5)



Simulated Annealing Methods

- Travelling Salesman Problem
  - Switching: Randomly select two cities and switch them in the tour

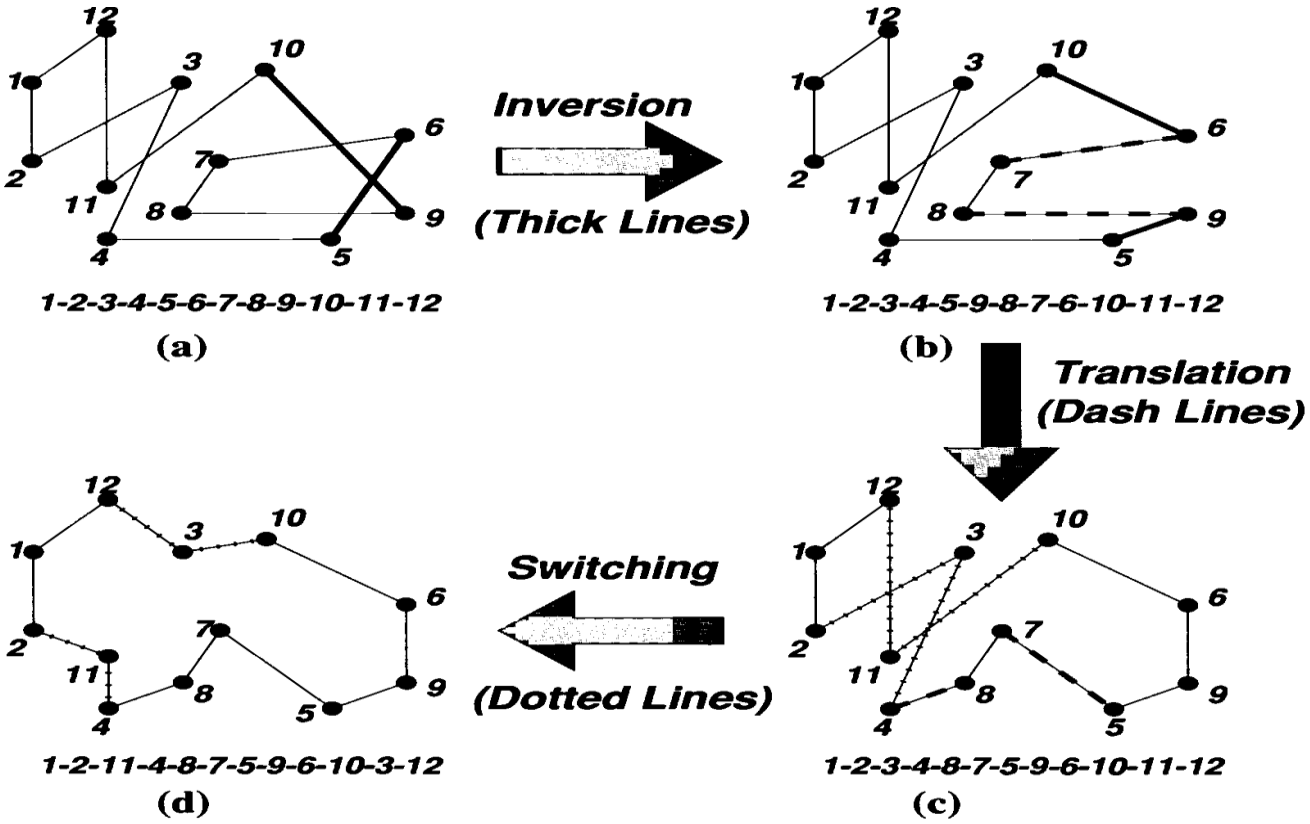


1-2-11-4-8-7-5-9-6-10-3-12

1-2-3-4-8-7-5-9-6-10-11-12

Simulated Annealing Methods

- Travelling Salesman Problem

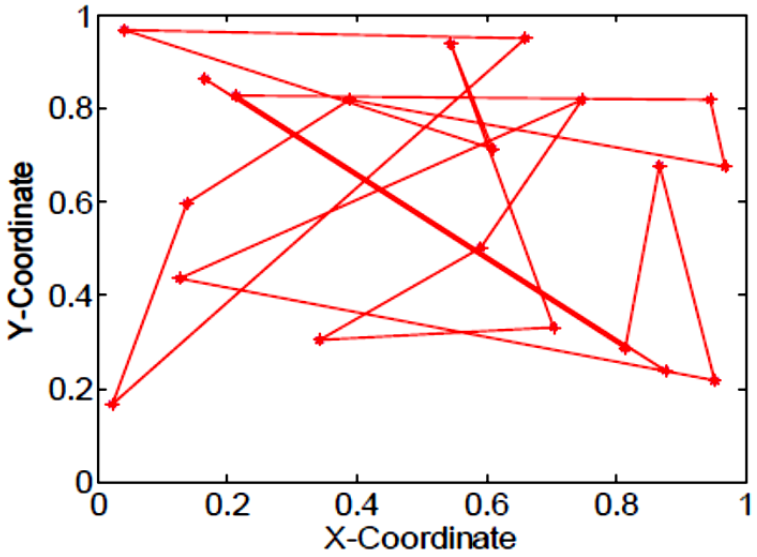


Simulated Annealing Methods

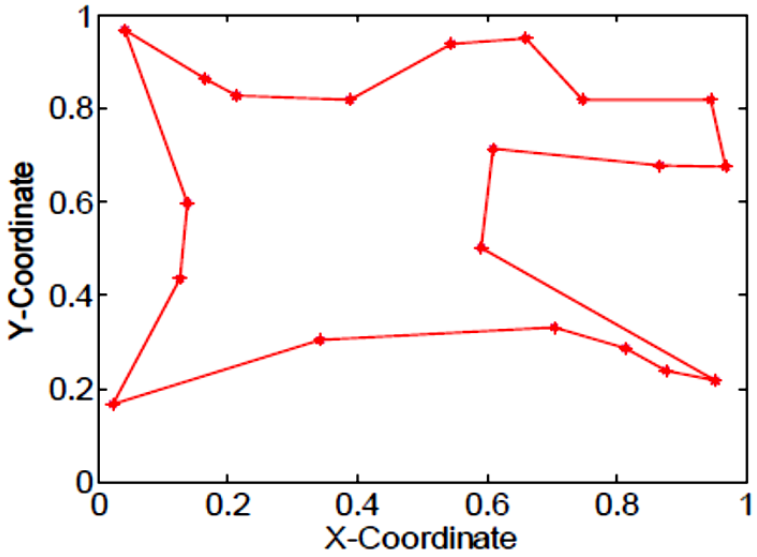
- Travelling Salesman Problem
  - **Step 1: start from random route R, initial temperature T & K = 1**
  - **Step 2: evaluate cost function E = f(R)**
  - **Step 3: define new route R_K by randomly swapping two cities**
  - **Step 4: if $f(R_K) < E$, then**
    - Accept new route
    - $R = R_K$ & $E = f(R_K)$
  - **End if**
  - **Step 5: if $f(R_K) \geq E$, then**
    - Accept new solution with certain probability
    - $R = R_K$ & $E = f(R_K)$ iff $rand(1) < \exp\{[E - f(R_K)]/T\}$
  - **End if**
  - **Step 6: $T = \alpha T$ ($\alpha < 1$), K = K + 1, and go to Step 3**

Simulated Annealing Methods

- Travelling Salesman Problem



Initial route



Optimized route

Optimizations

- **Introduction**
- Linear Programming (simplex method, M-Method)
- Simulated Annealing Methods
- **Nonlinear Programming**

Nonlinear Programming

- Problems that fit the general linear programming format but contain nonlinear functions are termed nonlinear programming (NLP) problems
- Solution methods are more complex than linear programming methods
- Determining an optimal solution is often difficult, if not impossible
- Solution techniques generally involve searching a solution surface for high or low points requiring the use of advanced mathematics.

Nonlinear Programming

- A nonlinear problem containing one or more constraints becomes a constrained optimization model or a nonlinear programming (NLP) model.
- A nonlinear programming model has the same general form as the linear programming model except that the objective function and/or the constraint(s) are nonlinear.
- Solution procedures are much more complex and no guaranteed procedure exists for all NLP models.

Nonlinear Programming

- Unlike linear programming, solution is often not on the boundary of the feasible solution space.
- Cannot simply look at points on the solution space boundary but must consider other points on the surface of the objective function.
- This greatly complicates solution approaches.
- Solution techniques can be very complex.

Nonlinear Programming

- A general nonlinear programming problem (NLP) can be expressed as follows:

$$\begin{aligned} \max\ (or\ \min)\ z &= f(x_1, x_2, \dots, x_n) \\ s.t. \quad & g_1(x_1, x_2, \dots, x_n)\ (\leq, =, or\ \geq) b_1 \\ s.t. \quad & g_2(x_1, x_2, \dots, x_n)\ (\leq, =, or\ \geq) b_2 \end{aligned}$$

$$\dots$$

$$g_m(x_1, x_2, \dots, x_n)\ (\leq, =, or\ \geq) bm$$

  - Find the values of decision variables $x_1, x_2, \dots x_n$ that

Nonlinear Programming

- Function $f(x_1, x_2, \dots, x_n)$ is the NLP's objective function, and
  - $g_1(x_1, x_2, \dots, x_n)(\leq, =, or\ \geq)\ b1, \dots g_m(x_1, x_2, \dots, x_n)(\leq, =, or\ \geq)\ b_m$ are the NLP's constraints.
  - An NLP with no constraints is an unconstrained NLP.
  - The feasible region for NLP above is the set of points $(x_1, x_2, \dots, x_n)$ that satisfy the m constraints in the NLP. A point in the feasible region is a feasible point, and a point that is not in the feasible region is an infeasible point

Nonlinear Programming

- If the NLP is a maximization problem then any point $x*$ in the feasible region for which $f(x*) \geq f(x)$ holds true for all points x in the feasible region is an optimal solution to the NLP.
- Even if the feasible region for an NLP is a convex set, the optimal solution need not be a extreme point of the NLP's feasible region.
- For any NLP (maximization), a feasible point $x = (x_1, x_2, \ldots, x_n)$ is a **local maximum**
  - if for sufficiently small $\in$, any feasible point $x' = (x'_1, x'_2, \ldots, x'_n)$ having $|x_1 - x'_1| < \in (i = 1,2,\ldots,n)$ satisfies $f(x) \geq f(x')$

Nonlinear Programming

- Suppose the feasible region S for NLP is a convex set.
  - If $f(x)$ is concave on S, then any local maximum (minimum) for the NLP is an optimal solution to the NLP.
  - Suppose $f(x_1, x_2, \dots, x_n)$ has continuous second-order partial derivatives for each point $x = (x_1, x_2, \dots, x_n) \in S$
    - Then $f(x_1, x_2, \dots, x_n)$ is a convex function on S if and only if for each $x \in S$, all principal minors of H are non-negative
      - The Hessian of $f(x_1, x_2, \dots, x_n)$ is the $n \times n$ matrix whose $ijth$ entry is

$$\frac{\partial^2 f}{\partial x_i \partial x_j}$$

Nonlinear Programming

- Saddle point
  - In the case of a function of two variables f (x,y), the Hessian matrix may be neither positive nor negative definite at a point (x*,y*) at which

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial y}$$

In such a case, the point $(x*, y*)$ is called a saddle point

  - The characteristic of a saddle point is that it corresponds to a relative minimum or maximum of $f(x, y)$ wrt one variable, say, x (the other variable being fixed at $y = y*$) and a relative maximum or minimum of $f(x, y)$ wrt the second variable y (the other variable being fixed at $x*$)

Nonlinear Programming

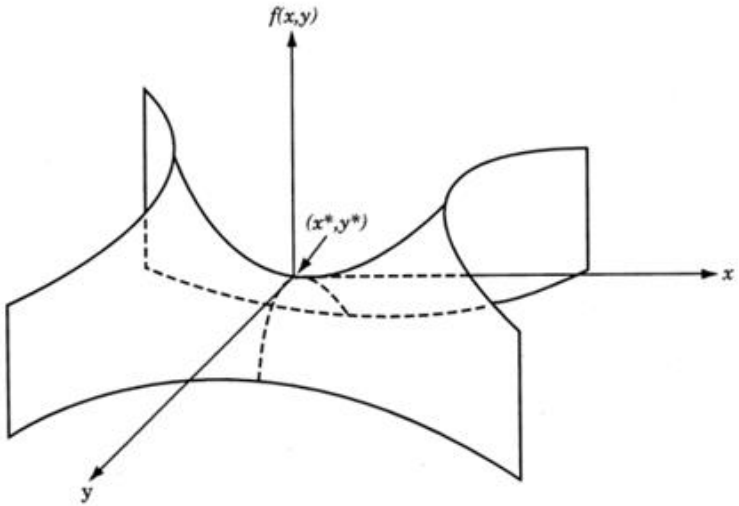
- Saddle point
  - Consider the function $f(x,y) = x^2 - y^2$
  - For this function:

$$\frac{\partial f}{\partial x} = 2x; \frac{\partial f}{\partial y} = -2y$$

    - These first derivatives are zero at $x^* = 0$ and $y^* = 0$
    - The Hessian matrix of f at (x^*, y^*) is given by

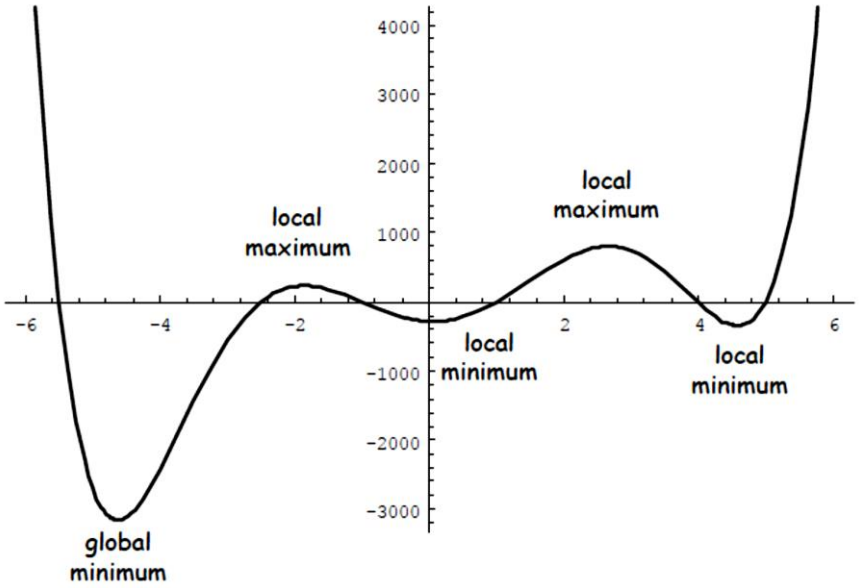
$$J = \begin{bmatrix} 2 & 0 \\ 0 & -2 \end{bmatrix}$$

    - Since this matrix is neither positive definite nor negative definite, the point $x^* = 0$ and $y^* = 0$ is a saddle point.



Nonlinear Programming

- Global minimum: a function $f(x)$ defined on a set S attains its global minimum at a point $x^* \in S$ if and only if $f(x^*) \leq f(x)\ \forall x \in S$
- Local minimum: a function $f(x)$ defined on a set S attains its global minimum (relative minimum) at a point $x^* \in S$ if and only if there exist $\epsilon > 0$ such that $f(x^*) \leq f(x)\ \forall x \in S$ *satisfying* $|x - x^*|$



Nonlinear Programming

- **Monotonic function**: a function $f(x)$ is monotonic (either increasing or decreasing) if for two point $x_1 \leq x_2 \in S$, it follows that
  - $f(x_1) \leq f(x_2)$: monotonically increasing
  - $f(x_1) \geq f(x_2)$: monotonically decreasing
- **Unimodal function**: a function $f(x)$ is unimodal function on the interval $a \leq x \leq b$ if and only if it is monotonic on either side of the single optimal point x^* in the interval. In other words, if x^* is the single minimum point of $f(x)$ in the range $a \leq x \leq b$, then $f(x)$ is unimodal on the interval if and only if for any two point x_1 and x_2
  - $x^* \leq x_1 \leq x_2$: implies that $f(x^*) \leq f(x_1) \leq f(x_2)$
  - $x^* \geq x_1 \geq x_2$: implies that $f(x^*) \leq f(x_1) \leq f(x_2)$

Nonlinear Programming

- Solving NLPs with One Variable

$$\max \ (or \ \min) \ \ f(x_1, x_2, \dots, x_n)$$
$$s.t. \ x \in [a, b]$$

  - There are three types of points for which the NLP can have a local maximum or minimum (these points are often called extremum candidates)
  - Points where $a < x < b, f'(x) = 0$: called a stationary point of $f(x)$
  - Points where $f'(x)$ does not exist Endpoints a and b of the interval [a,b]

Nonlinear Programming

- Solving NLPs with One Variable

$$\max \ (or \ \min) \ \ f(x)$$
$$s.t. \ x \in [a, b]$$

  - To find the optimal solution for the NLP find all the local maxima (or minima)
  - A point that is a local maximum or a local minimum for the NLP is called a local extremum
  - The optimal solution is the local maximum (or minimum) having the largest (or smallest) value of $f(x)$

Nonlinear Programming

- Unconstrained Maximization and Minimization with Several Variables

$$\max \ (or \ \min) \ \ f(x_1, x_2, \dots, x_n)$$
$$s.t. (x_1, x_2, \dots, x_n) \in [a, b]$$

  - These theorems provide the basics of unconstrained NLP's that may have two or more decision variables.

  - If $\bar{x}$ $(\bar{x}_1, \bar{x}_2, \dots, \bar{x}_n)$ is a local extremum for the NLP then $\frac{\partial f(\bar{x})}{\partial x_i} = 0$

    - A point $\bar{x}$ having $\frac{\partial f(\bar{x})}{\partial x_i} = 0$ for $i = 1, 2, \dots, n$ is called a **stationary point** of f.

    - The stationary points include local and global minima and maxima, but may also include the points that are none of the above such as **inflection points**

Nonlinear Programming

- Unconstrained Maximization and Minimization with Several Variables

$$\max \ (or \ \min) \ \ f(x_1, x_2, \dots, x_n)$$
$$s.t. \ (x_1, x_2, \dots, x_n) \in [a, b]$$

  - Find the minima and maxima in this case
    - We find all the solutions $\bar{x}$ of the equation $\nabla f(x) = 0$ and then we rely on the second order information (**the Hessian of the function f:** $\nabla^2 f(\bar{x})$)

Nonlinear Programming

- Unconstrained Maximization and Minimization with Several Variables

$$\max \ (or \ \min) \ \ f(x_1, x_2, \dots, x_n)$$
$$s.t. \ (x_1, x_2, \dots, x_n) \in [a, b]$$

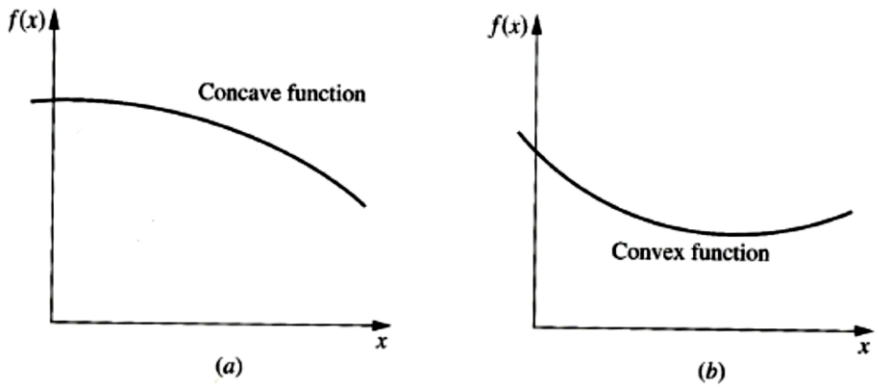
  - Find the minima and maxima in this case:
    - We check the Hessian $\nabla^2 f(\bar{x})$at each of the points $\bar{x}$:
      - If the matrix $\nabla^2 f(\bar{x}) < 0$ then $\bar{x}$ is a local maximum
      - If the matrix $\nabla^2 f(\bar{x}) > 0$ then $\bar{x}$ is a local miniimum
      - If the matrix $\nabla^2 f(\bar{x}) = 0$ then we do not know what is happening

Nonlinear Programming

- Unconstrained Maximization and Minimization with Several Variables

$$\max \ (or \ \min) \ \ f(x_1, x_2, \dots, x_n)$$
$$s.t. \ (x_1, x_2, \dots, x_n) \in [a, b]$$

  - What helps us here in some situations is that the function is "nice" for the given optimization problem:
    - f is convex and we need to minimize f
    - f is concave and we need to maximize f



Nonlinear Programming

- Golden Section Search
  - The Golden Section Method can be used if the function is a unimodal function
  - A function $f(x)$ is unimodel on $[a, b]$ if for some point x^* on $[a, b]$, $f(x)$ is strictly increasing on $[a,\ x^*]$ and strictly decreasing on $[x^*, b]$
  - The optimal solution of the NLP is some point on the interval $[a, b]$
    - By evaluating f(x) at two points x_1 and x_2 on $[a, b]$, we may reduce the size of the interval in which the solution to the NLP must lie

Nonlinear Programming

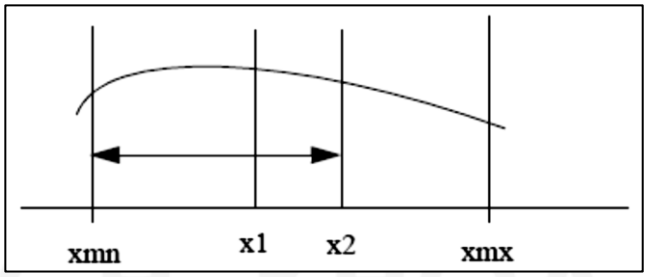
- Golden Section Search
  - The Golden section search is a technique for finding the extremum (minimum or maximum) of a strictly unimodal function by successively narrowing the range of values inside which the extremum is known to exist.
  - -The technique derives its name from the fact that the algorithm maintains the function values for triples of points whose distances form a golden ratio.
  - -Fibonacci search and Golden section search were discovered by Kiefer (1953)

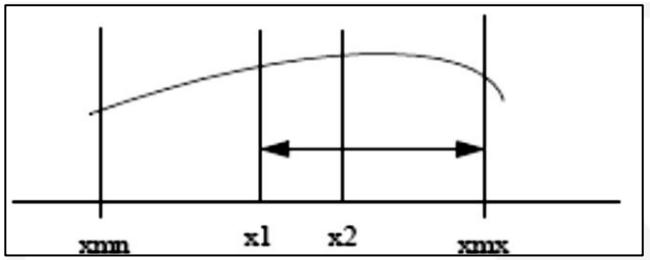
Nonlinear Programming

- Golden Section Search
  - A unimodal function contains only one minimum or maximum on the interval [a,b])
    - Assume that we are trying to find the maximum of a function
      - Define an interval with a single answer (unique maximum) inside the range sign of the curvature does not change in the given range
      - Divide interval into 3 sections by adding two internal points between ends

Nonlinear Programming

- Golden Section Search
  - Evaluate the function at the two internal points $x1\ and\ x2$
    - $if\ f(x1) > f(x2)$
      - the maximum is between xmn and $x2$
      - redefine range $xmn = xmn, xmx = x2$
    - $if\ f(x1) < f(x2)$
      - the maximum is between $x1$ and xmx
      - redefine range $xmn = x1, xmx = xmx$





Nonlinear Programming

- Gradient methods
  - Gradient Descent

$$x_{i+1} = x_i - \gamma_i \left((\nabla f)(x_i)\right)^T$$

  - Gradient Descent With Momentum

$$\boldsymbol{x}_{i+1} = \boldsymbol{x}_i - \gamma_i((\nabla f)(\boldsymbol{x}_i))^\top + \alpha\Delta\boldsymbol{x}_i$$
$$\Delta\boldsymbol{x}_i = \boldsymbol{x}_i - \boldsymbol{x}_{i-1} = \alpha\Delta\boldsymbol{x}_{i-1} - \gamma_{i-1}((\nabla f)(\boldsymbol{x}_{i-1}))^\top$$

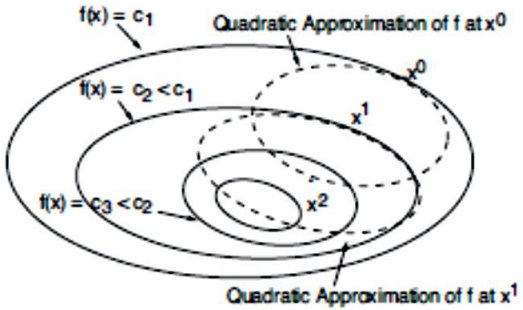
  - Newton's method

$$x^{i+1} = x^i - \alpha^i \left(\nabla^2 f(x^i)\right)^{-1} \nabla f(x^i), \qquad i = 0, 1, \ldots$$

Nonlinear Programming

- Gradient methods

Slow convergence of steepest descent



Fast convergence of Newton's method w/ $\alpha^k = 1$.

Given x^k, the method obtains x^{k+1} as the minimum of a quadratic approximation of f based on a second order Taylor expansion around x^k.

Nonlinear Programming

- Nonlinear conjugate gradient
  - minimize $f(x)$, f convex and differentiable

$x_{i+1} = x_i + \alpha_i d_i$, where α_i is obtained by line minimization

$d_i = -\nabla f(x_i) + \beta_i d_{i-1}, \; i = 1,2,\dots,n-1$

$$\beta_i = \frac{(\nabla f(x_i))^{\mathrm{T}} \nabla f(x_i)}{(\nabla f(x_{i-1}))^{\mathrm{T}} \nabla f(x_{i-1})} \quad \text{or} \quad \beta_i = \frac{(\nabla f(x_i) - \nabla f(x_{-1}))^{\mathrm{T}} \nabla f(x_i)}{(\nabla f(x_{i-1}))^{\mathrm{T}} \nabla f(x_{i-1})}$$

the method terminates with an optimal solution after at most n steps

Nonlinear Programming

- Quasi – Newton method
  - minimize $f(x)$, f convex and differentiable

$$\Delta x_i = -\alpha_i B_i^{-1} \nabla f(x_i)$$

$$x_{i+1} = x_i + \Delta x_i,$$

  - **B_i is an inverse Hessian approximation**
  - **Reasonable requirement for B_i**

$$\nabla f(x_i) = \nabla f(x_{i-1}) + B_i(x_i - x_{i-1})$$

Nonlinear Programming

- Conditional gradient method
  - minimize $f(x)$, f convex and differentiable

$$x_{i+1} = x_i + \alpha_i(\bar{x}_i - x_i)\ ,$$

$$\bar{x}_i = \text{argmin}_{x \in X}(\nabla f(x_i)(x - x_i))$$

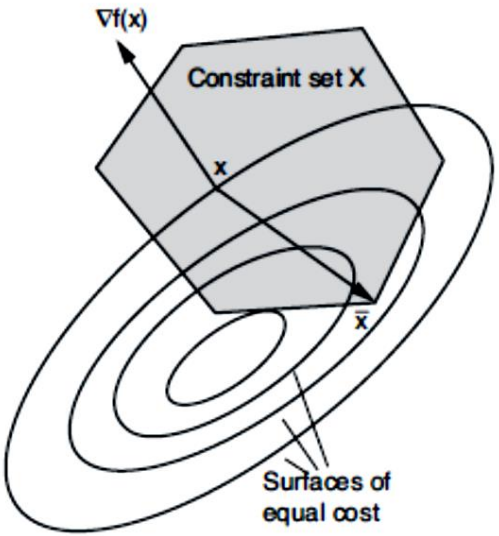


Illustration of the direction of the conditional gradient method.

Nonlinear Programming

- Gradient Search Procedure:
  - is to keep moving in the direction of the gradient from the current trial solution, not stopping until $f(x)$ stops increasing. This stopping point would be the next trial solution, so the gradient then would be recalculated to determine the new direction in which to move.
  - With this approach, each iteration involves changing the current trial solution x' as follows

$$\text{Reset} \qquad \mathbf{x}' = \mathbf{x}' + t^* \nabla f(\mathbf{x}'),$$

where t^* is the positive value of t that *maximizes* $f(\mathbf{x}' + t\,\nabla f(\mathbf{x}'))$; that is,

$$f(\mathbf{x}' + t^* \nabla f(\mathbf{x}')) = \max_{t \geq 0} f(\mathbf{x}' + t\,\nabla f(\mathbf{x}')).$$

Note that $f(\mathbf{x}' + t\,\nabla f(\mathbf{x}'))$ is simply $f(\mathbf{x})$ where

$$x_j = x_j' + t\left(\frac{\partial f}{\partial x_j}\right)_{\mathbf{x}=\mathbf{x}'}, \qquad \text{for } j = 1, 2, \ldots, n,$$

Nonlinear Programming

- Gradient Search Procedure:

and that these expressions for the x_j involve only constants and t, so $f(\mathbf{x})$ becomes a function of just the single variable t. The iterations of this gradient search procedure continue until $\nabla f(\mathbf{x}) = 0$ within a small tolerance ϵ, that is, until

$$\left| \frac{\partial f}{\partial x_j} \right| \leq \epsilon \qquad \text{for } j = 1, 2, \ldots, n.$$

Nonlinear Programming

- Gradient Search Procedure:

Initialization: Select ϵ and any initial trial solution x'. Go first to the stopping rule.

Iteration:

1. Express $f(\mathbf{x}' + t\,\nabla f(\mathbf{x}'))$ as a function of t by setting

$$x_j = x_j' + t\left(\frac{\partial f}{\partial x_j}\right)_{\mathbf{x}=\mathbf{x}'}, \qquad \text{for } j = 1, 2, \ldots, n,$$

and then substituting these expressions into $f(\mathbf{x})$.

2. Use the one-dimensional search procedure (or calculus) to find $t = t^*$ that maximizes $f(\mathbf{x}' + t\,\nabla f(\mathbf{x}'))$ over $t \geq 0$.
3. Reset $\mathbf{x}' = \mathbf{x}' + t^*\,\nabla f(\mathbf{x}')$. Then go to the stopping rule.

Stopping rule: Evaluate $\nabla f(\mathbf{x}')$ at $\mathbf{x} = \mathbf{x}'$. Check if

$$\left|\frac{\partial f}{\partial x_j}\right| \leq \epsilon \qquad \text{for all } j = 1, 2, \ldots, n.$$

If so, stop with the current $\mathbf{x}'$ as the desired approximation of an optimal solution $\mathbf{x}^*$. Otherwise, perform another iteration.

Nonlinear Programming

- Gradient Search Procedure:

Consider the following two-variable problem:

Maximize $f(\mathbf{x}) = 2x_1x_2 + 2x_2 - x_1^2 - 2x_2^2.$

$$\frac{\partial f}{\partial x_1} = 2x_2 - 2x_1,$$

$$\frac{\partial f}{\partial x_2} = 2x_1 + 2 - 4x_2.$$

It can be verified that $f(\mathbf{x})$ is concave by the convexity test.

To begin the gradient search procedure, $\mathbf{x} = (0,0)$ is selected as the initial trial solution. The gradient at $\mathbf{x} = (0,0)$ is $\nabla f(0, 0) = (0, 2).$

To begin the first iteration, set $x_1 = 0 + t(0) = 0,$

$$x_2 = 0 + t(2) = 2t,$$

and then substitute these expressions into $f(\mathbf{x})$ to obtain

$$\begin{aligned} f(\mathbf{x}' + t\,\nabla f(\mathbf{x}')) &= f(0, 2t) \\ &= 2(0)(2t) + 2(2t) - 0^2 - 2(2t)^2 \\ &= 4t - 8t^2. \end{aligned}$$

Nonlinear Programming

- Gradient Search Procedure:

Because $f(0, 2t^*) = \max_{t \geq 0} f(0, 2t) = \max_{t \geq 0} \{4t - 8t^2\}$

and $\frac{d}{dt}(4t - 8t^2) = 4 - 16t = 0,$

it follows that $t^* = \frac{1}{4},$

so reset $\mathbf{x}' = (0, 0) + \frac{1}{4}(0, 2) = (0, \frac{1}{2}).$

For this new trial solution, the gradient is

$$\nabla f(0, \tfrac{1}{2}) = (1, 0).$$

Maximize $f(\mathbf{x}) = 2x_1x_2 + 2x_2 - x_1^2 - 2x_2^2.$

Thus for the second iteration, set

$$\mathbf{x} = \mathbf{x}' + t\,\nabla f(\mathbf{x}') = (0, \tfrac{1}{2}) + t(1, 0) = (t, \tfrac{1}{2}),$$

so

$$f(\mathbf{x}' + t\,\nabla f(\mathbf{x}')) = f(t, \tfrac{1}{2})$$
$$= (2t)(\tfrac{1}{2}) + 2(\tfrac{1}{2}) - t^2 - 2(\tfrac{1}{2})^2$$
$$= t - t^2 + \tfrac{1}{2}.$$

Because $f(t^*, \frac{1}{2}) = \max_{t \geq 0} f(t, \frac{1}{2}) = \max_{t \geq 0} \{t - t^2 + \frac{1}{2}\}$

and $\frac{d}{dt}(t - t^2 + \frac{1}{2}) = 1 - 2t = 0,$

then $t^* = \frac{1}{2},$

so reset $\mathbf{x}' = \mathbf{x}' + t\,\nabla f(\mathbf{x}') = (0, \frac{1}{2}) + \frac{1}{2}(1, 0) = (\frac{1}{2}, \frac{1}{2}).$

Nonlinear Programming

- Lagrange Multipliers
  - Lagrange multipliers can be used to solve NLPs in which all the constraints are equality constraints
  - Consider NLPs of the following type:

$$\max\ (or\ \min)\ z = f(x_1, x_2, \dots, x_n)$$
$$s.t. \quad g_1(x_1, x_2, \dots, x_n) = b_1$$
$$s.t. \quad g_2(x_1, x_2, \dots, x_n) = b_2$$
$$\dots$$
$$g_m(x_1, x_2, \dots, x_n) = b_m$$

Nonlinear Programming

- Lagrange Multipliers
  - To solve the NLP, associate a **multiplier** λ_i with the *i*th constraint in the NLP and form the **Lagrangian**

$$L(x_1, x_2, \dots, x_n) = f(x_1, x_2, \dots, x_n) + \sum_{i=1}^{m} \lambda_i (b_i - g_i(x_1, \dots, x_m)$$

  - for which the points $(x_1, x_2, \dots, x_n, \lambda_1, \lambda_2, \dots, \lambda_m)$

$$\frac{\partial L}{\partial x_1} = \frac{\partial L}{\partial x_2} = \cdots = \frac{\partial L}{\partial x_n} = 0$$

$$\frac{\partial L}{\partial \lambda_1} = \frac{\partial L}{\partial \lambda_2} = \cdots = \frac{\partial L}{\partial \lambda_m} = 0$$

$$g_j(x) = \frac{\partial L}{\partial \lambda_j}$$

  - The Lagrange multipliers λ_1 can be used in sensitivity analysis.

- Lagrange Multipliers
  - A sufficient condition for $f(x_1, x_2, \ldots, x_n)$ to have a constrained relative **minimum** at $x^*$is that the quadratic Q defined by:

$$Q = \sum_{i=1}^{n} \sum_{j=1}^{n} \frac{\partial^2 L}{\partial x_i \partial x_j} dx_i dx_j$$

evaluated at $x = x^*$ must be **positive** definite for all values of dx for which the constraints are satisfied

  - if

$$Q = \sum_{i=1}^{n} \sum_{j=1}^{n} \frac{\partial^2 L}{\partial x_i \partial x_j}(\mathbf{X^*}, \lambda^*) dx_i dx_j$$

is **negative** for all choices of the admissable variations dx_i, x^* will be a constrained **maximum** of $f(x)$

- Lagrange Multipliers
  - A necessary condition for the quadratic form Q to be positive (negative) definite for all admissable variations dx is that each root of the polynomial z_i, defined by the following determinantal equation, be positive (negative):

$$\begin{vmatrix} L_{11}-z & L_{12} & L_{13} & \cdots & L_{1n} & g_{11} & g_{21} & \cdots & g_{m1} \\ L_{21} & L_{22}-z & L_{23} & \cdots & L_{2n} & g_{12} & g_{22} & \cdots & g_{m2} \\ \vdots & & & & & & & & \\ L_{n1} & L_{n2} & L_{n3} & \cdots & L_{nn}-z & g_{1n} & g_{2n} & \cdots & g_{mn} \\ g_{11} & g_{12} & g_{13} & \cdots & g_{1n} & 0 & 0 & \cdots & 0 \\ g_{21} & g_{22} & g_{23} & \cdots & g_{2n} & 0 & 0 & \cdots & 0 \\ \vdots & & & & & & & & \\ g_{m1} & g_{m2} & g_{m3} & \cdots & g_{mn} & 0 & 0 & \cdots & 0 \end{vmatrix} = 0$$

$$L_{ij} = \frac{\partial^2 L}{\partial x_i \, \partial x_j}(\mathbf{X}^*, \boldsymbol{\lambda}^*)$$

$$g_{ij} = \frac{\partial g_i}{\partial x_j}(\mathbf{X}^*)$$

Nonlinear Programming

- Lagrange Multipliers
  - Find the maximum of the function

$$f(x_1, x_2) = 2x_1 + x_2 + 10$$

s. t. $g(x_1, x_2) = {x_1}^2 + 2{x_2}^2 = 3$

using the Lagrange multiplier method.

Also find the effect of changing the right-hand side of the constraint on the optimum value of *f*.

Nonlinear Programming

- Lagrange Multipliers
  - The Lagrange function is given by:

$$L(x_1, x_2, \lambda) = 2x_1 + x_2 + 10 + \lambda(3 - x_1 - 2x_2^2)$$

  - The necessary conditions for the solution of the problem are:

$$\frac{\partial L}{\partial x_1} = 2 - \lambda = 0$$

$$\frac{\partial L}{\partial x_2} = 1 - 4\lambda x_2 = 0$$

$$\frac{\partial L}{\partial \lambda} = g(x_1, x_2) = 3 - x_1 - 2x_2^2 = 0$$

$$\mathbf{X}^* = \begin{Bmatrix} x_1^* \\ x_2^* \end{Bmatrix} = \begin{Bmatrix} 2.97 \\ 0.13 \end{Bmatrix}$$

$$\lambda^* = 2$$

Nonlinear Programming

- Lagrange Multipliers
  - The application of the sufficiency condition yields:

$$\frac{\partial L}{\partial x_1} = 2 - \lambda;\ \mathrm{L}_{11} = \frac{\partial L}{\partial x_1^2} = 0;\ \mathrm{L}_{12}\frac{\partial L}{\partial x_1 \partial x_2} = 0;\ \frac{\partial L}{\partial x_2} = 1 - 4\lambda x_2;\ \mathrm{L}_{22}\frac{\partial L}{\partial x_2^2} = -4\lambda;\ L_{21} = \frac{\partial L}{\partial x_2 \partial x_1} = 0;$$

$$g(x_1, x_2) = 3 - x_1 - 2x_2^2;\ \frac{\partial g}{\partial x_1} = -1; \frac{\partial g}{\partial x_2} = -4x_2;$$

$$g_{11} = \frac{\partial g}{\partial x_1} = -1; g_{12} = \ -4x_2$$

$$\mathbf{X^*} = \begin{Bmatrix} x_1^* \\ x_2^* \end{Bmatrix} = \begin{Bmatrix} 2.97 \\ 0.13 \end{Bmatrix}$$

$$\lambda^* = 2$$

$$\begin{vmatrix} L_{11} - z & L_{12} & g_{11} \\ L_{21} & L_{22} - z & g_{12} \\ g_{11} & g_{12} & 0 \end{vmatrix} = 0$$

$$0.2704z + 8 + z = 0$$

$$z = -6.2972$$

$$\begin{vmatrix} -z & 0 & -1 \\ 0 & -4\lambda - z & -4x_2 \\ -1 & -4x_2 & 0 \end{vmatrix} = \begin{vmatrix} -z & 0 & -1 \\ 0 & -8 - z & -0.52 \\ -1 & -0.52 & 0 \end{vmatrix} = 0$$

Hence X* will be a maximum of *f* with *f = *f* (X*)=16.07**

Nonlinear Programming

- The Karush-Kuhn-Tucker (KKT) Conditions
  - The KKT conditions are used to solve NLPs of the following type:

$$\max \ (or \ \min) \ z = f(x_1, x_2, \dots, x_n)$$
$$s.t. \quad g_1(x_1, x_2, \dots, x_n) \le b_1$$
$$s.t. \quad g_2(x_1, x_2, \dots, x_n) \le b_2$$
$$\dots$$
$$g_m(x_1, x_2, \dots, x_n) \le b_m$$

Nonlinear Programming

- The Karush-Kuhn-Tucker (KKT) Conditions
  - The KKT conditions are necessary for a point $(\bar{x}_1, \bar{x}_2, \dots, \bar{x}_n)$ to solve the NLP
    - Suppose the NLP is a maximization problem. If $\bar{x} = (\bar{x}_1, \bar{x}_2, \dots, \bar{x}_n)$ is an optimal solution to NLP, then $(\bar{x}_1, \bar{x}_2, \dots, \bar{x}_n)$ must satisfy the *m* constraints in the NLP, and there must exist multipliers $\lambda_1, \lambda_2, \dots, \lambda_m$ satisfying

$$\frac{\partial f(\bar{x})}{\partial x_j} - \sum_{i=1}^{i=m} \lambda_i \frac{\partial g(\bar{x})}{\partial x_j} = 0 \quad (j = 1,2,\dots,n)$$

$$\lambda_i [b_i - g_i(\bar{x})] = 0 \quad (i = 1,2,\dots,m)$$

$$\lambda_i \geq 0 \quad (i = 1,2,\dots,m)$$

Nonlinear Programming

- The KKT Conditions
  - The **necessary and sufficient conditions** that must be satisfied by an optimal solution for a nonlinear programming problem.

Necessary and sufficient conditions for optimality

Problem	Necessary Conditions for Optimality	Also Sufficient If:
One-variable unconstrained	$\frac{df}{dx} = 0$	$f(x)$ concave
Multivariable unconstrained	$\frac{\partial f}{\partial x_j} = 0 \quad (j = 1, 2, \ldots, n)$	$f(\mathbf{x})$ concave
Constrained, nonnegativity constraints only	$\frac{\partial f}{\partial x_j} = 0 \quad (j = 1, 2, \ldots, n)$ (or ≤ 0 if $x_j = 0$)	$f(\mathbf{x})$ concave
General constrained problem	Karush-Kuhn-Tucker conditions	$f(\mathbf{x})$ concave and $g_i(\mathbf{x})$ convex $(i = 1, 2, \ldots, m)$

Nonlinear Programming

- The KKT Conditions

These conditions may be developed as follows: The problem is to maximize $f(x, y)$ subject to

$$g(x, y) \leq 0 \tag{1}$$

Let z be a new variable and let $g(x, y)$ be defined by

$$z^2 = -g(x, y) \tag{2}$$

This condition forces $g(x, y)$ to be non-positive and the problem can now be expressed as: Maximize $f(x, y)$, subject to:

$$g(x, y) + z^2 = 0 \tag{3}$$

In this form the problem can be treated by the method of Lagrange multipliers, since the constraint is now an equality. Let

$$h(x, y, \lambda, z) = f(x, y) - \lambda[g(x, y) + z^2]$$

Nonlinear Programming

- The KKT Conditions

$$h(x, y, \lambda, z) = f(x, y) - \lambda[g(x, y) + z^2]$$

The necessary conditions for a stationary point are:

$$\frac{\partial h}{\partial x} = 0; \qquad \frac{\partial f}{\partial x} - \lambda \frac{\partial g}{\partial x} = 0 \tag{4}$$

$$\frac{\partial h}{\partial y} = 0; \qquad \frac{\partial f}{\partial y} - \lambda \frac{\partial g}{\partial y} = 0 \tag{5}$$

$$\frac{\partial h}{\partial \lambda} = 0; \qquad g(x, y) + z^2 = 0 \tag{6}$$

$$\frac{\partial h}{\partial z} = 0; \qquad 2\lambda z = 0 \tag{7}$$

Nonlinear Programming

- The KKT Conditions

The necessary conditions for a stationary point are:

$$\frac{\partial h}{\partial x} = 0; \quad \frac{\partial f}{\partial x} - \lambda \frac{\partial g}{\partial x} = 0 \tag{4}$$

$$\frac{\partial h}{\partial y} = 0; \quad \frac{\partial f}{\partial y} - \lambda \frac{\partial g}{\partial y} = 0 \tag{5}$$

$$\frac{\partial h}{\partial \lambda} = 0; \quad g(x, y) + z^2 = 0 \tag{6}$$

$$\frac{\partial h}{\partial z} = 0; \quad 2\lambda z = 0 \tag{7}$$

(7) requires that either $\lambda = 0$ or $z = 0$; if $z = 0$, then from (6) $g(x, y)$ also equals zero. Equations (6) and (7) together therefore imply that

$$\lambda g(x, y) = 0 \tag{8}$$

at a stationary point.

Since (6) merely restates the requirement that $g(x, y)$ be non-positive, the necessary conditions for a local extremum are (4), (5), (8), and $g(x, y) \leq 0$.

Nonlinear Programming

- The KKT Conditions

The necessary condition for a point (x, y) to be a local maximum of $f(x, y)$ subject to $g(x, y) \leq 0$ is that a non-negative λ exists such that λ and (x, y) satisfy the following:

$$\frac{\partial f}{\partial x} - \lambda \frac{\partial g}{\partial x} = 0 \tag{9}$$

$$\frac{\partial f}{\partial y} - \lambda \frac{\partial g}{\partial y} = 0$$

$$\lambda g(x, y) = 0$$

$$g(x, y) \leq 0$$

The result is directly applicable to minimizing a convex function, since a maximum point of f is a minimum point of $-f$.

Nonlinear Programming

- The KKT Conditions

Maximize $E(x, y) = 4x + 5y + xy - x^2 - y^2 + 5$
subject to

$$\frac{x}{2} + y \leq w$$

$E(x, y)$ is a concave function Let $g(x, y) = \frac{x}{2} + y - w$

Then the necessary conditions for a maximum from (9) are

$$\frac{\partial E}{\partial x} - \lambda \frac{\partial g}{\partial x} = 4 + y - 2x - \frac{\lambda}{2} = 0 \qquad (10)$$

$$\frac{\partial E}{\partial y} - \lambda \frac{\partial g}{\partial y} = 5 + x - 2y - \lambda = 0 \qquad (11)$$

$$\lambda\left(\frac{x}{2} + y - w\right) = 0 \qquad (12)$$

$$\frac{x}{2} + y - w \leq 0 \qquad (13)$$

$$\lambda \geq 0 \qquad (14)$$

From (12) it follows that either $\lambda = 0$ or $g(x, y) = x/$

Nonlinear Programming

- The KKT Conditions

From (12) it follows that either $\lambda = 0$ or $g(x, y) = x/2 + y - w = 0$.

If $\lambda = 0$, then (10) and (11) reduce to

$$4 + y - 2x = 0$$
$$5 + x - 2y = 0$$

which have the solution $x = \frac{13}{3}$; $y = \frac{14}{3}$

$$\lambda\left(\frac{x}{2} + y - w\right) = 0 \qquad (12)$$

This solution will also satisfy (12), (13), and (14) if $w \geq \frac{41}{6}$, but will not satisfy (13) if $w < \frac{41}{6}$.

Nonlinear Programming

- The KKT Conditions

If $\lambda \neq 0$, then to satisfy (12):

$$\frac{x}{2} + y - w = 0 \qquad \text{or} \qquad y = w - \frac{x}{2}$$

Substitution in (10) and (11) gives

$$x = \tfrac{3}{7} + \tfrac{4}{7}w \qquad y = -\tfrac{3}{14} + \tfrac{5}{7}w \qquad \lambda = \tfrac{41}{7} - \tfrac{6}{7}w$$

If $w < \frac{41}{6}$, then $\lambda > 0$; $w > \frac{41}{6}$, then $\lambda < 0$

The solution may be summarized as follows:

If $w \geq \frac{41}{6}$ the optimum solution is

$$x = \tfrac{13}{3}; \qquad y = \tfrac{14}{3}; \qquad \lambda = 0$$

If $w < \frac{41}{6}$ the optimum solution is

$$x = \tfrac{3}{7} + \tfrac{4}{7}w; \qquad y = -\tfrac{3}{14} + \tfrac{5}{7}w; \qquad \lambda = \tfrac{41}{7} - \tfrac{6}{7}w \ (> 0)$$

Nonlinear Programming

- The KKT Conditions

To generalize the conditions to more than one inequality constraint, consider next the case of two inequality constraints. The problem is to maximize $f(x, y)$ subject to $g_1(x, y) \leq 0$ and $g_2(x, y) \leq 0$. Two new variables z_1 and z_2 are defined and the constraints set equal to z_1^2 and z_2^2; e.g.,

$$-g_1(x, y) = z_1^2; \qquad -g_2(x, y) = z_2^2$$

Two Lagrange multipliers, λ_1 and λ_2, are defined and the function to be maximized is now a function of six variables.

$$h(x, y, \lambda_1, \lambda_2, z_1, z_2) = f(x, y) - \lambda_1[g_1(x, y) + z_1^2] - \lambda_2[g_2(x, y) + z_2^2]$$

Nonlinear Programming

- The KKT Conditions

Following the method used above and eliminating z_1 and z_2 leads to the following statement:

The necessary condition for a point (x, y) to be a maximum of $f(x, y)$, subject to $g_1(x, y) \leq 0$ and $g_2(x, y) \leq 0$, is that non-negative values of λ_1 and λ_2 exist such that λ_1, λ_2, x, and y satisfy the equations:

$$\frac{\partial f}{\partial x} - \lambda_1 \frac{\partial g_1}{\partial x} - \lambda_2 \frac{\partial g_2}{\partial x} = 0 \qquad (15)$$

$$\frac{\partial f}{\partial y} - \lambda_1 \frac{\partial g_1}{\partial y} - \lambda_2 \frac{\partial g_2}{\partial y} = 0 \qquad (16)$$

$$g_1(x, y) \leq 0; \qquad g_2(x, y) \leq 0 \qquad (17)$$

$$\lambda_1 g_1(x, y) = 0; \qquad \lambda_2 g_2(x, y) = 0 \qquad (18)$$

$$\frac{\partial f}{\partial x} - \lambda \frac{\partial g}{\partial x} = 0$$

$$\frac{\partial f}{\partial y} - \lambda \frac{\partial g}{\partial y} = 0$$

$$\lambda g(x, y) = 0$$

$$g(x, y) \leq 0$$

Nonlinear Programming

- The KKT Conditions

Maximize $f(x, y)$
subject to

$$g_1(x, y) \leq 0$$
$$g_2(x, y) \leq 0$$
$$x \geq 0;\ y \geq 0$$

$$\frac{\partial f}{\partial x} - \lambda_1 \frac{\partial g_1}{\partial x} - \lambda_2 \frac{\partial g_2}{\partial x} = 0$$

$$\frac{\partial f}{\partial y} - \lambda_1 \frac{\partial g_1}{\partial y} - \lambda_2 \frac{\partial g_2}{\partial y} = 0$$

$$g_1(x, y) \leq 0; \qquad g_2(x, y) \leq 0$$

$$\lambda_1 g_1(x, y) = 0; \qquad \lambda_2 g_2(x, y) = 0$$

$$\frac{\partial f}{\partial x} - \lambda_1 \frac{\partial g_1}{\partial x} - \lambda_2 \frac{\partial g_2}{\partial x} \leq 0$$

$$\frac{\partial f}{\partial y} - \lambda_1 \frac{\partial g_1}{\partial y} - \lambda_2 \frac{\partial g_2}{\partial y} \leq 0$$

$$x\left(\frac{\partial f}{\partial x} - \lambda_1 \frac{\partial g_1}{\partial x} - \lambda_2 \frac{\partial g_2}{\partial x}\right) = 0$$

$$y\left(\frac{\partial f}{\partial y} - \lambda_1 \frac{\partial g_1}{\partial y} - \lambda_2 \frac{\partial g_2}{\partial y}\right) = 0$$

$$g_1(x, y) \leq 0; \qquad g_2(x, y) \leq 0$$

$$\lambda_1 g_1(x, y) = 0; \qquad \lambda_2 g_2(x, y) = 0$$

$$x \geq 0;\ y \geq 0$$

$$\lambda_1 \geq 0;\ \lambda_2 \geq 0$$

Nonlinear Programming

- The KKT Conditions for general case

A point $(x_1, x_2, \ldots, x_n)$ maximizes a function $f(x_1, \ldots, x_n)$ subject to $g_j(x_1, \ldots, x_n) \leq 0$, $j = 1, \ldots, m$ if there exists a set of non-negative values $\lambda_1, \ldots, \lambda_m$ such that

$$\frac{\partial f}{\partial x_i} - \sum_{j=1}^{m} \lambda_j \frac{\partial g_j}{\partial x_i} = 0$$

$$\lambda_j g_j = 0$$

$$g_j \leq 0$$

$$\lambda_j \geq 0$$

$$\frac{\partial f}{\partial x} - \lambda_1 \frac{\partial g_1}{\partial x} - \lambda_2 \frac{\partial g_2}{\partial x} = 0$$

$$\frac{\partial f}{\partial y} - \lambda_1 \frac{\partial g_1}{\partial y} - \lambda_2 \frac{\partial g_2}{\partial y} = 0$$

$$g_1(x, y) \leq 0; \qquad g_2(x, y) \leq 0$$

$$\lambda_1 g_1(x, y) = 0; \qquad \lambda_2 g_2(x, y) = 0$$

- **These conditions are sufficient if f is concave and g_i are all convex.**

Nonlinear Programming

- The KKT Conditions for general case

The necessary conditions if the x's are to be non-negative are

$$\frac{\partial f}{\partial x_i} - \sum_{j=1}^{m} \lambda_j \frac{\partial g_j}{\partial x_i} \leq 0$$

$$x_i \left(\frac{\partial f}{\partial x_i} - \sum_{j=1}^{m} \lambda_j \frac{\partial g_j}{\partial x_i} \right) = 0$$

$$g_j \lambda_j = 0$$

$$g_j \leq 0$$

$$x_i \geq 0$$

$$\lambda_j \geq 0$$

$$\frac{\partial f}{\partial x} - \lambda_1 \frac{\partial g_1}{\partial x} - \lambda_2 \frac{\partial g_2}{\partial x} \leq 0$$

$$\frac{\partial f}{\partial y} - \lambda_1 \frac{\partial g_1}{\partial y} - \lambda_2 \frac{\partial g_2}{\partial y} \leq 0$$

$$x \left(\frac{\partial f}{\partial x} - \lambda_1 \frac{\partial g_1}{\partial x} - \lambda_2 \frac{\partial g_2}{\partial x} \right) = 0$$

$$y \left(\frac{\partial f}{\partial y} - \lambda_1 \frac{\partial g_1}{\partial y} - \lambda_2 \frac{\partial g_2}{\partial y} \right) = 0$$

$$g_1(x, y) \leq 0; \qquad g_2(x, y) \leq 0$$

$$\lambda_1 g_1(x, y) = 0; \qquad \lambda_2 g_2(x, y) = 0$$

$$x \geq 0; \quad y \geq 0$$

$$\lambda_1 \geq 0; \quad \lambda_2 \geq 0$$

Nonlinear Programming

- The KKT Conditions for general case

Maximize $f(\mathbf{x})$.

Subject to

$g_i(\mathbf{x}) \le b_i$, for $i = 1, 2, \ldots, m$,

$\mathbf{x} \ge \mathbf{0}$

THEOREM: Assume that $f(\mathbf{x})$, $g_1(\mathbf{x})$, $g_2(\mathbf{x})$, . . . , $g_m(\mathbf{x})$ are *differentiable* functions satisfying certain regularity conditions.[3] Then

$$\mathbf{x}^* = (x_1^*, x_2^*, \ldots, x_n^*)$$

can be an *optimal solution* for the nonlinear programming problem only if there exist m numbers $u_1, u_2, \ldots, u_m$ such that *all* the following *KKT conditions* are satisfied:

1. $\dfrac{\partial f}{\partial x_j} - \sum_{i=1}^{m} u_i \dfrac{\partial g_i}{\partial x_j} \le 0$
2. $x_j^* \left(\dfrac{\partial f}{\partial x_j} - \sum_{i=1}^{m} u_i \dfrac{\partial g_i}{\partial x_j} \right) = 0$

} at $\mathbf{x} = \mathbf{x}^*$, for $j = 1, 2, \ldots, n$.

3. $g_i(\mathbf{x}^*) - b_i \le 0$
4. $u_i[g_i(\mathbf{x}^*) - b_i] = 0$

} for $i = 1, 2, \ldots, m$.

5. $x_j^* \ge 0$, for $j = 1, 2, \ldots, n$.
6. $u_i \ge 0$, for $i = 1, 2, \ldots, m$.

$$\frac{\partial f}{\partial x_i} - \sum_{j=1}^{m} \lambda_j \frac{\partial g_j}{\partial x_i} \le 0$$

$$x_i \left(\frac{\partial f}{\partial x_i} - \sum_{j=1}^{m} \lambda_j \frac{\partial g_j}{\partial x_i} \right) = 0$$

$$g_j \lambda_j = 0$$

$$g_j \le 0$$

$$x_i \ge 0$$

$$\lambda_j \ge 0$$

Nonlinear Programming

- Quadratic Programming.
  - A quadratic programming problem (QPP) is an NLP in which each term in the objective function is of degree $2, 1, or\ 0$ and all constraints are linear
  - In practice, the method of complementary pivoting is most often used to solve QPPs.
    - The quadratic programming problem differs from the linear programming problem only in that the objective function also includes x_j^2 and $x_i x_j\ (i \neq j)$

Nonlinear Programming

- Quadratic Programming
  - The matrix form of a quadratic programming problem is

$$\begin{aligned} &\text{Maximize} && f(\mathbf{x}) = \mathbf{cx} - \tfrac{1}{2}\mathbf{x}^T\mathbf{Q}\mathbf{x}, \\ &\text{subject to} \\ &&& \mathbf{Ax} \leq \mathbf{b} \quad \text{and} \quad \mathbf{x} \geq \mathbf{0}, \end{aligned}$$

where $\mathbf{c}$ is a row vector, $\mathbf{x}$ and $\mathbf{b}$ are column vectors, $\mathbf{Q}$ and $\mathbf{A}$ are matrices, and the superscript T denotes the transpose (see Appendix 4). The q_{ij} (elements of Q) are given constants such that $q_{ij} = q_{ji}$

    - Q is a symmetrical matrix.
  - The algebraic form of the objective function of this quadratic programming problem is

$$f(\mathbf{x}) = \mathbf{cx} - \tfrac{1}{2}\mathbf{x}^T\mathbf{Q}\mathbf{x} = \sum_{j=1}^{n} c_j x_j - \tfrac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} q_{ij} x_i x_j.$$

If $i = j$ in this double summation, then $x_i x_j = x_j^2$.

Nonlinear Programming

- Quadratic Programming.
  - To illustrate the notation, consider the following example of a quadratic programming problem.

$$\begin{aligned} \text{Maximize} \quad & f(x_1, x_2) = 15x_1 + 30x_2 + 4x_1x_2 - 2x_1^2 - 4x_2^2, \\ \text{subject to} \quad & x_1 + 2x_2 \le 30 \\ & x_1 \ge 0, \qquad x_2 \ge 0. \end{aligned}$$

$$\begin{aligned} \text{Maximize} \quad & f(\mathbf{x}) = \mathbf{cx} - \tfrac{1}{2}\mathbf{x}^T\mathbf{Qx}, \\ \text{subject to} \quad & \mathbf{Ax} \le \mathbf{b} \quad \text{and} \quad \mathbf{x} \ge \mathbf{0}, \end{aligned}$$

Nonlinear Programming

- Quadratic Programming.

$$\begin{aligned}\text{Maximize} \quad & f(x_1, x_2) = 15x_1 + 30x_2 + 4x_1x_2 - 2x_1^2 - 4x_2^2, \\ \text{subject to} \quad & x_1 + 2x_2 \le 30 \\ & x_1 \ge 0, \qquad x_2 \ge 0.\end{aligned}$$

$$f(x_1, x_2) = 15x_1 + 30x_2 - \tfrac{1}{2}(4x_1^2 - 4x_2x_1 - 4x_1x_2 + 8x_2^2)$$

$$\mathbf{c} = [15 \quad 30], \quad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \quad \mathbf{Q} = \begin{bmatrix} 4 & -4 \\ -4 & 8 \end{bmatrix}, \quad \mathbf{A} = [1 \quad 2], \quad \mathbf{b} = [30].$$

$$\begin{aligned}\text{Note that} \quad \mathbf{x}^T\mathbf{Q}\mathbf{x} &= [x_1 \quad x_2]\begin{bmatrix} 4 & -4 \\ -4 & 8 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \\ &= [(4x_1 - 4x_2) \quad (-4x_1 + 8x_2)]\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \\ &= 4x_1^2 - 4x_2x_1 - 4x_1x_2 + 8x_2^2 \\ &= q_{11}x_1^2 + q_{21}x_2x_1 + q_{12}x_1x_2 + q_{22}x_2^2.\end{aligned}$$

Nonlinear Programming

- Quadratic Programming.

$$\begin{aligned} \text{Maximize} \quad & f(x_1, x_2) = 15x_1 + 30x_2 + 4x_1x_2 - 2x_1^2 - 4x_2^2, \\ \text{subject to} \quad & x_1 + 2x_2 \le 30 \\ & x_1 \ge 0, \qquad x_2 \ge 0. \end{aligned}$$

$$f(\mathbf{x}) = \mathbf{cx} - \tfrac{1}{2}\mathbf{x}^T\mathbf{Qx} = \sum_{j=1}^{n} c_j x_j - \tfrac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} q_{ij} x_i x_j.$$

In this case, $f(x_1, x_2)$ can be rewritten as

$$f(x_1, x_2) = 15x_1 + 30x_2 - \tfrac{1}{2}(4x_1^2 - 4x_2x_1 - 4x_1x_2 + 8x_2^2)$$

Nonlinear Programming

- Quadratic Programming: KKT conditions for quadratic programming

$$\begin{aligned}\text{Maximize} \quad & f(x_1, x_2) = 15x_1 + 30x_2 + 4x_1x_2 - 2x_1^2 - 4x_2^2, \\ \text{subject to} \quad & x_1 + 2x_2 \le 30 \\ & x_1 \ge 0, \qquad x_2 \ge 0.\end{aligned}$$

$$g_1(x_1, x_2) = x_1 + 2x_2$$

1($j = 1$). $15 + 4x_2 - 4x_1 - u_1 \le 0.$
2($j = 1$). $x_1(15 + 4x_2 - 4x_1 - u_1) = 0.$
1($j = 2$). $30 + 4x_1 - 8x_2 - 2u_1 \le 0.$
2($j = 2$). $x_2(30 + 4x_1 - 8x_2 - 2u_1) = 0.$
3. $x_1 + 2x_2 - 30 \le 0.$
4. $u_1(x_1 + 2x_2 - 30) = 0.$
5. $x_1 \ge 0, \qquad x_2 \ge 0.$
6. $u_1 \ge 0.$

1. $\dfrac{\partial f}{\partial x_j} - \sum_{i=1}^{m} u_i \dfrac{\partial g_i}{\partial x_j} \le 0$

2. $x_j^* \left(\dfrac{\partial f}{\partial x_j} - \sum_{i=1}^{m} u_i \dfrac{\partial g_i}{\partial x_j} \right) = 0$

} for $j = 1, 2, \ldots, n.$

3. $g_i(\mathbf{x}^*) - b_i \le 0$
4. $u_i[g_i(\mathbf{x}^*) - b_i] = 0$

} for $i = 1, 2, \ldots, m.$

5. $x_j^* \ge 0,$ for $j = 1, 2, \ldots, n.$
6. $u_i \ge 0,$ for $i = 1, 2, \ldots, m.$

Nonlinear Programming

- Quadratic Programming.

$1(j=1).\ 15 + 4x_2 - 4x_1 - u_1 \le 0. \rightarrow -4x_1 + 4x_2 - u_1 + y_1 = -15$

$2(j=1).\ x_1(15 + 4x_2 - 4x_1 - u_1) = 0.$

$1(j=2).\ 30 + 4x_1 - 8x_2 - 2u_1 \le 0. \rightarrow 4x_1 - 8x_2 - 2u_1 + y_2 = -30$

$2(j=2).\ x_2(30 + 4x_1 - 8x_2 - 2u_1) = 0.$

$3.\ x_1 + 2x_2 - 30 \le 0. \rightarrow x_1 + 2x_2 + v_1 = 30$

$4.\ u_1(x_1 + 2x_2 - 30) = 0.$

$5.\ x_1 \ge 0, \quad x_2 \ge 0.$

$6.\ u_1 \ge 0.$

$1(j=1).\ 15 + 4x_2 - 4x_1 - u_1 \le 0.$

$2(j=1).\ x_1(15 + 4x_2 - 4x_1 - u_1) = 0. \rightarrow x_1 y_1 = 0$

$1(j=2).\ 30 + 4x_1 - 8x_2 - 2u_1 \le 0.$

$2(j=2).\ x_2(30 + 4x_1 - 8x_2 - 2u_1) = 0. \rightarrow x_2 y_2 = 0$

$3.\ x_1 + 2x_2 - 30 \le 0.$

$4.\ u_1(x_1 + 2x_2 - 30) = 0. \rightarrow u_1 v_1 = 0$

$5.\ x_1 \ge 0, \quad x_2 \ge 0.$

$6.\ u_1 \ge 0.$

Nonlinear Programming

- Quadratic Programming.

For each of these three pairs—(x_1, y_1), (x_2, y_2), (u_1, v_1)—the two variables are called **complementary variables**, because only one of the two variables can be nonzero.

We have $x_1y_1 + x_2y_2 + u_1v_1 = 0$ called the **complementarity constraint**

We now have the desired convenient form for the entire set of conditions shown below.

$$\begin{aligned} 4x_1 - 4x_2 + u_1 - y_1 &= 15 \\ -4x_1 + 8x_2 + 2u_1 - y_2 &= 30 \\ x_1 + 2x_2 + v_1 &= 30 \end{aligned}$$

$$x_1 \geq 0, \quad x_2 \geq 0, \quad u_1 \geq 0, \quad y_1 \geq 0, \quad y_2 \geq 0, \quad v_1 \geq 0$$

$$x_1y_1 + x_2y_2 + u_1v_1 = 0$$

Nonlinear Programming

- Quadratic Programming.

$$
\begin{aligned}
4x_1 - 4x_2 + u_1 - y_1 &= 15 \\
-4x_1 + 8x_2 + 2u_1 - y_2 &= 30 \\
x_1 + 2x_2 + v_1 &= 30 \\
x_1 \geq 0, \quad x_2 \geq 0, \quad u_1 \geq 0, \quad y_1 \geq 0, \quad y_2 \geq 0, \quad v_1 \geq 0 & \\
x_1 y_1 + x_2 y_2 + u_1 v_1 = 0 &
\end{aligned}
$$

This form is particularly convenient because, except for the complementarity constraint, these conditions are linear programming constraints.

Nonlinear Programming

- Quadratic Programming
  - For any quadratic programming problem, its KKT conditions can be reduced to this same convenient form containing just linear programming constraints plus one complementarity constraint. In matrix notation, this general form is

$$\begin{aligned} \mathbf{Qx} + \mathbf{A}^T\mathbf{u} - \mathbf{y} &= \mathbf{c}^T, \\ \mathbf{Ax} + \mathbf{v} &= \mathbf{b}, \\ \mathbf{x}^T\mathbf{y} + \mathbf{u}^T\mathbf{v} &= 0, \end{aligned} \qquad \mathbf{x} \geq \mathbf{0}, \quad \mathbf{u} \geq \mathbf{0}, \quad \mathbf{y} \geq \mathbf{0}, \quad \mathbf{v} \geq \mathbf{0},$$

$$\mathbf{c} = [15 \quad 30], \quad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \quad \mathbf{Q} = \begin{bmatrix} 4 & -4 \\ -4 & 8 \end{bmatrix}, \quad \mathbf{A} = [1 \quad 2], \quad \mathbf{b} = [30].$$

  - where the elements of the column vector **u** are Lagrange multipliers and the elements of the column vectors **y** and **v** are slack variables
  - The original problem is reduced to the equivalent problem of finding a feasible solution to these constraints.

Nonlinear Programming

- Quadratic Programming
  - The KKT conditions for quadratic programming are nothing more than linear programming constraints, except for the complementary constrain.
  - The complementary constraint simply implies that it is not permissible for both complementary variables of any pair to be basic variables when BF solutions are considered.
  - As a result, the problem reduces to finding an initial BF solution to any linear programming problem that has the constraints shown below (obtained from the KKT conditions for quadratic programming).

Nonlinear Programming

- Quadratic Programming
  - As a result, the problem reduces to finding an initial BF solution to any linear programming problem that has the constraints shown below (obtained from the KKT conditions for quadratic programming)

$$\begin{aligned} \mathbf{Qx} + \mathbf{A}^T\mathbf{u} - \mathbf{y} &= \mathbf{c}^T, \\ \mathbf{Ax} + \mathbf{v} &= \mathbf{b}, \\ \mathbf{x} \geq \mathbf{0}, \quad \mathbf{u} \geq \mathbf{0}, \quad \mathbf{y} \geq \mathbf{0}, \quad \mathbf{v} &\geq \mathbf{0}, \\ \mathbf{x}^T\mathbf{y} + \mathbf{u}^T\mathbf{v} &= 0, \end{aligned}$$

  - In the simple case where $\mathbf{c}^T \leq 0$ and $\mathbf{b} \geq 0$, we can easily find a feasible solution

$$\mathbf{x} = \mathbf{0}, \mathbf{u} = \mathbf{0}, \mathbf{y} = -\mathbf{c}^T, \mathbf{v} = \mathbf{b}$$

    - which satisfies the above constraints and thus is the optimal solution for the quadratic programming problem
    - In the case where $\mathbf{c}^T > 0$ or $\mathbf{b} < 0$, we have to introduce artificial variables as we do for the Big M method or the two-phase method.

Nonlinear Programming

- Quadratic Programming.
  - So we use phase 1 of the two-phase method to find a BF solution satisfying the constraints obtained from the KKT conditions
  - Specifically, we apply the simplex method with one modification to the following linear programming problem

$$\text{Minimize} \qquad Z = \sum_j z_j,$$

  - subject to the linear programming constraints obtained from the KKT conditions, but with the artificial variables z_j included.
  - The one modification in the simplex method is the following change in the procedure for selecting an entering basic variable.

Nonlinear Programming

- Quadratic Programming

$$\begin{aligned} \text{Maximize} \quad & f(x_1, x_2) = 15x_1 + 30x_2 + 4x_1x_2 - 2x_1^2 - 4x_2^2, \\ \text{subject to} \quad & x_1 + 2x_2 \le 30 \\ & x_1 \ge 0, \qquad x_2 \ge 0. \end{aligned}$$

  - As can be verified from the convexity test, $f(x_1, x_2)$ is strictly concave, so the modified simplex method can be applied.

Nonlinear Programming

- Quadratic Programming
  - After the artificial variables are introduced, the linear programming problem to be addressed by the modified simplex method is

$$
\begin{aligned}
&\text{Minimize} \quad Z = z_1 + z_2, \\
&\text{subject to} \\
&4x_1 - 4x_2 + u_1 - y_1 + z_1 = 15 \\
&-4x_1 + 8x_2 + 2u_1 - y_2 + z_2 = 30 \\
&x_1 + 2x_2 + v_1 = 30 \\
&x_1 y_1 + x_2 y_2 + u_1 v_1 = 0 \\
&x_1 \geq 0,\ x_2 \geq 0,\ u_1 \geq 0,\ y_1 \geq 0,\ y_2 \geq 0, \\
&v_1 \geq 0,\ z_1 \geq 0,\ z_2 \geq 0.
\end{aligned}
$$

$$
\begin{gathered}
\mathbf{Qx} + \mathbf{A}^T\mathbf{u} - \mathbf{y} = \mathbf{c}^T, \\
\mathbf{Ax} + \mathbf{v} = \mathbf{b}, \\
\mathbf{x} \geq \mathbf{0}, \quad \mathbf{u} \geq \mathbf{0}, \quad \mathbf{y} \geq \mathbf{0}, \quad \mathbf{v} \geq \mathbf{0}, \\
\mathbf{x}^T\mathbf{y} + \mathbf{u}^T\mathbf{v} = 0,
\end{gathered}
$$

Nonlinear Programming

- Quadratic Programming
  - For each of the three pairs of complementary variables $(x_1, y_1), (x_2, y_2), (u_1, v_1)$, whenever one of the two variables already is a basic variable, the other variable is excluded as a candidate for the entering basic variable
  - The initial set of basic variables z_1, z_2, v_1 gives an initial BF solution that satisfies the complementary constraint.

Nonlinear Programming

- Quadratic Programming
  -

Iteration	Basic Variable	Eq.	Z	x_1	x_2	u_1	y_1	y_2	v_1	z_1	z_2	Right Side
0	Z	(0)	−1	0	−4	−3	1	1	0	0	0	−45
	z_1	(1)	0	4	−4	1	−1	0	0	1	0	15
	z_2	(2)	0	−4	8	2	0	−1	0	0	1	30
	v_1	(3)	0	1	2	0	0	0	1	0	0	30
1	Z	(0)	−1	−2	0	−2	1	$\frac{1}{2}$	0	0	$\frac{1}{2}$	−30
	z_1	(1)	0	2	0	2	−1	$-\frac{1}{2}$	0	1	$\frac{1}{2}$	30
	x_2	(2)	0	$-\frac{1}{2}$	1	$\frac{1}{4}$	0	$-\frac{1}{8}$	0	0	$\frac{1}{8}$	$3\frac{3}{4}$
	v_1	(3)	0	2	0	$-\frac{1}{2}$	0	$\frac{1}{4}$	1	0	$-\frac{1}{4}$	$22\frac{1}{2}$
2	Z	(0)	−1	0	0	$-\frac{5}{2}$	1	$\frac{3}{4}$	1	0	$\frac{1}{4}$	$-7\frac{1}{2}$
	z_1	(1)	0	0	0	$\frac{5}{2}$	−1	$-\frac{3}{4}$	−1	1	$\frac{3}{4}$	$7\frac{1}{2}$
	x_2	(2)	0	0	1	$\frac{1}{8}$	0	$-\frac{1}{16}$	$\frac{1}{4}$	0	$\frac{1}{16}$	$9\frac{3}{8}$
	x_1	(3)	0	1	0	$-\frac{1}{4}$	0	$\frac{1}{8}$	$\frac{1}{2}$	0	$-\frac{1}{8}$	$11\frac{1}{4}$
3	Z	(0)	−1	0	0	0	0	0	0	1	1	0
	u_1	(1)	0	0	0	1	$-\frac{2}{5}$	$-\frac{3}{10}$	$-\frac{2}{5}$	$\frac{2}{5}$	$\frac{3}{10}$	3
	x_2	(2)	0	0	1	0	$\frac{1}{20}$	$-\frac{1}{40}$	$\frac{3}{10}$	$-\frac{1}{20}$	$\frac{1}{40}$	9
	x_1	(3)	0	1	0	0	$-\frac{1}{10}$	$\frac{1}{20}$	$\frac{2}{5}$	$\frac{1}{10}$	$-\frac{1}{20}$	12

Nonlinear Programming

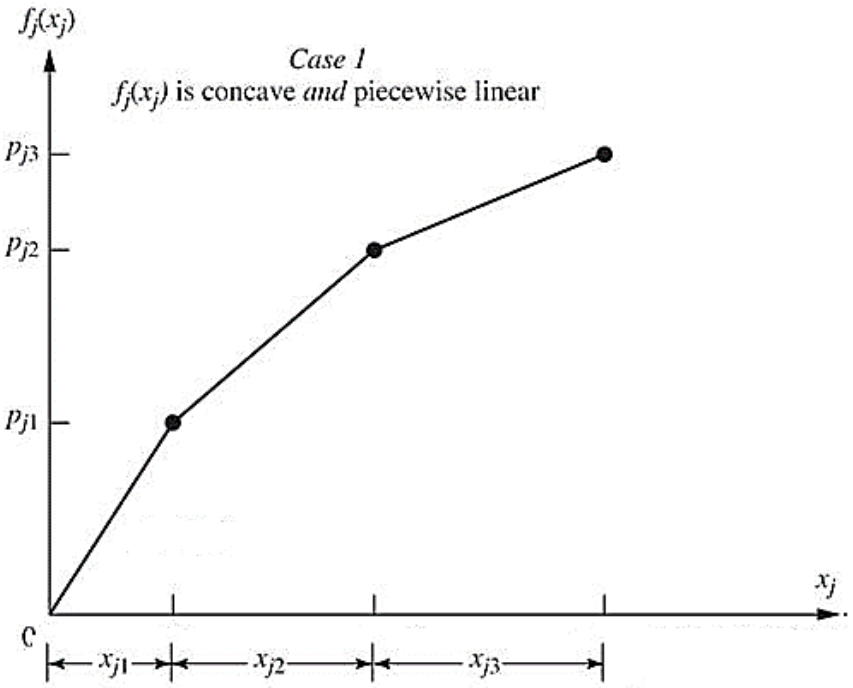
- Separable Programming
  - In Separable programming , it is assumed that the objective function $f(x)$ is concave, that each of the constraint functions $g_i(x)$ is convex, and that all these functions are separable functions
    - To simplify the discussion, we focus here on the special case where the convex and separable $g_i(x)$ are linear functions, just as for linear programming. Thus only the objective function requires special treatment.
    - A concave and separable function can be expressed as a sum of concave functions of individual variables.

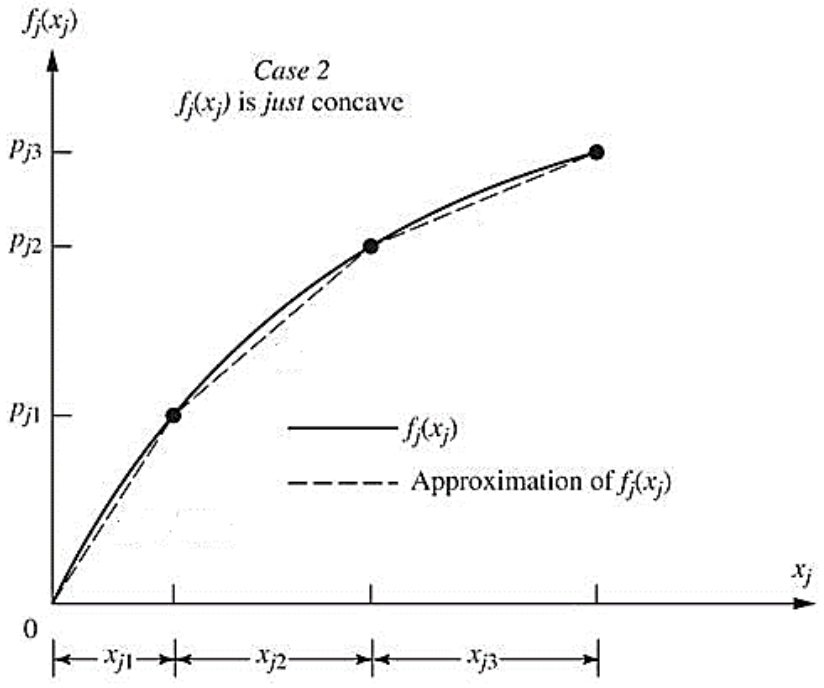
$$max\ (or\ \min) z = \sum_{j=1}^{m} f_j(x_j),\ s.t. \quad \sum_{j=1}^{n} g_{ij}(x_j) \le b_i, \quad i = 1, \dots, m$$

  - They are called **seperable programming problems** that are often solved by approximating each $f_j(x)$ and $g_{ij}(x_j)$ by a piecewise linear function.

Nonlinear Programming

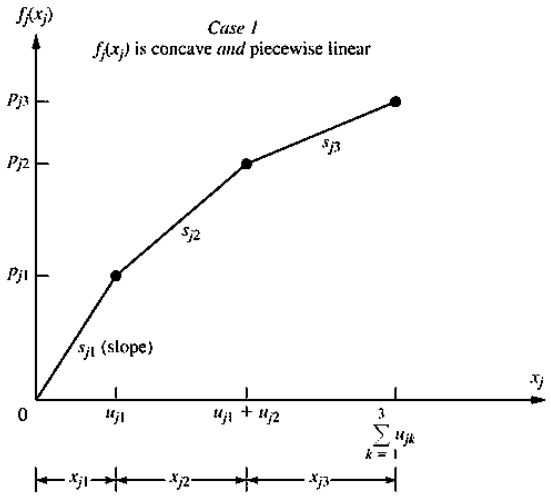
- Separable Programming
  - Each $f_j(x_j)$ has a shape such as either case shown in the figure below over the feasible range of values of x_j

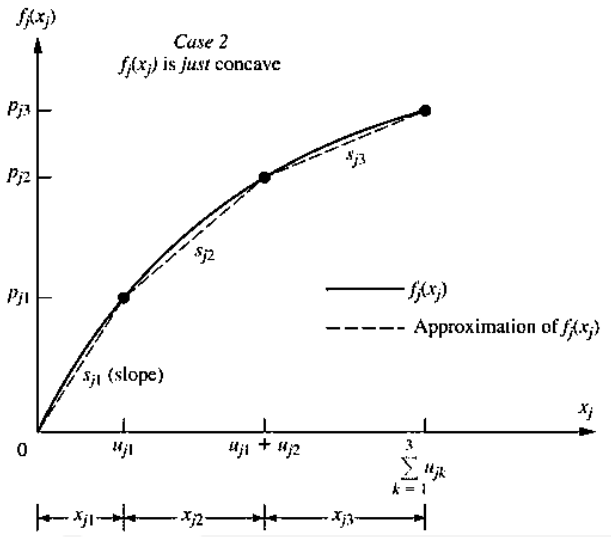




Nonlinear Programming

- Separable Programming
  - In case 1, the slope decreases only at certain breakpoints, so that $f_j(\mathbf{x}_j)$ is a piecewise linear function (a sequence of connected line segments)
  - In case 2, the slope may decrease continuously as $\mathbf{x}_j$ increases, so that $f_j(\mathbf{x}_j)$ is a general concave function. Any such function can be approximated as closely as desired by a piecewise linear function





Nonlinear Programming

- Separable Programming
  - The key to rewriting a piecewise linear function as a linear function is to use a separate variable for each line segment.
  - To illustrate, consider the piecewise linear function in case 1 or the approximating piecewise linear function in case 2
    - Introduce three new variables x_{j1}, x_{j2}, and x_{j3} and set

$$x_j = x_{j1} + x_{j2} + x_{j3},$$

where $0 \le x_{j1} \le u_{j1}, \quad 0 \le x_{j2} \le u_{j2}, \quad 0 \le x_{j3} \le u_{j3}.$

Then use the slopes s_{j1}, s_{j2}, and s_{j3} to rewrite $f_j(x_j)$ as

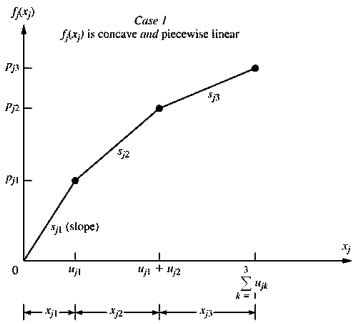
$$f_j(x_j) = s_{j1}x_{j1} + s_{j2}x_{j2} + s_{j3}x_{j3},$$

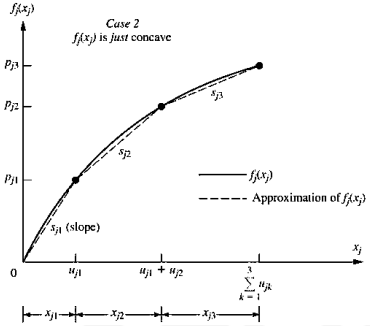
with the *special restriction* that

$x_{j2} = 0$ whenever $x_{j1} < u_{j1}$, $x_{j3} = 0$ whenever $x_{j2} < u_{j2}$.

Nonlinear Programming

- Separable Programming
  - Special restriction





$$x_{j2} = 0 \text{ whenever } x_{j1} < u_{j1}, \quad x_{j3} = 0 \text{ whenever } x_{j2} < u_{j2}.$$

To see why this special restriction is required, suppose that $x_j = 1$, where $u_{jk} > 1$ $(k = 1, 2, 3)$, so that $f_j(1) = s_{j1}$.

$$x_{j1} + x_{j2} + x_{j3} = 1$$

$$\begin{aligned} x_{j1} = 1, \quad x_{j2} = 0, \quad x_{j3} = 0 &\implies f_j(1) = s_{j1}, \\ x_{j1} = 0, \quad x_{j2} = 1, \quad x_{j3} = 0 &\implies f_j(1) = s_{j2}, \\ x_{j1} = 0, \quad x_{j2} = 0, \quad x_{j3} = 1 &\implies f_j(1) = s_{j3}, \end{aligned}$$

and so on, where $s_{j1} > s_{j2} > s_{j3}$.

However, the special restriction permits only the first possibility

Nonlinear Programming

- Separable Programming
  - To write down the complete model in the preceding notation, let n_j be the number of line segments in $f_j(x_j)$, so that

$$x_j = \sum_{k=1}^{n_j} x_{jk}$$

would be substituted throughout the original model and

$$f_j(x_j) = \sum_{k=1}^{n_j} s_{jk} x_{jk}$$

would be substituted into the objective function for $j = 1, 2, \ldots, n$. The resulting model is

$$\text{Maximize} \quad Z = \sum_{j=1}^{n} \left(\sum_{k=1}^{n_j} s_{jk} x_{jk} \right),$$

subject to

$$\sum_{j=1}^{n} a_{ij} \left(\sum_{k=1}^{n_j} x_{jk} \right) \le b_i, \qquad \text{for } i = 1, 2, \ldots, m$$

$$x_{jk} \le u_{jk}, \qquad \text{for } k = 1, 2, \ldots, n_j;\ j = 1, 2, \ldots, n$$

$$x_{jk} \ge 0, \qquad \text{for } k = 1, 2, \ldots, n_j;\ j = 1, 2, \ldots, n$$

$$x_{jk} = 0 \text{ whenever } x_{j,k-1} < u_{j,k-1}, \text{ for } k = 1, 2, \ldots, n_j;\ j = 1, 2, \ldots, n$$

Nonlinear Programming

- Separable Programming
  - Unfortunately, the special restriction does not fit into the required format for linear programming constraints.
  - However, our $f_j(x_j)$ are assumed to be concave so that an algorithm for maximizing $f(x)$ automatically gives the highest priority to using x_{j1} when increasing x_j from zero, the next highest priority to using x_{j2}, and so on, without even including the special restriction explicitly in the model. This observation leads to the following key property

Nonlinear Programming

- Separable Programming

KEY PROPERTY OF SEPARABLE PROGRAMMING: When $f(\mathbf{x})$ and the $g_i(\mathbf{x})$ satisfy the assumptions of separable programming, and when the resulting piecewise linear functions are rewritten as linear functions, deleting the *special restriction* gives a *linear programming model* whose optimal solution automatically satisfies the special restriction.

$$\text{Maximize} \quad Z = \sum_{j=1}^{n} \left(\sum_{k=1}^{n_j} s_{jk} x_{jk} \right),$$

subject to

$$\sum_{j=1}^{n} a_{ij} \left(\sum_{k=1}^{n_j} x_{jk} \right) \le b_i, \qquad \text{for } i = 1, 2, \ldots, m$$

$$x_{jk} \le u_{jk}, \qquad \text{for } k = 1, 2, \ldots, n_j; \; j = 1, 2, \ldots, n$$

$$x_{jk} \ge 0, \qquad \text{for} \qquad k = 1, 2, \ldots, n_j; \; j = 1, 2, \ldots, n$$

Nonlinear Programming

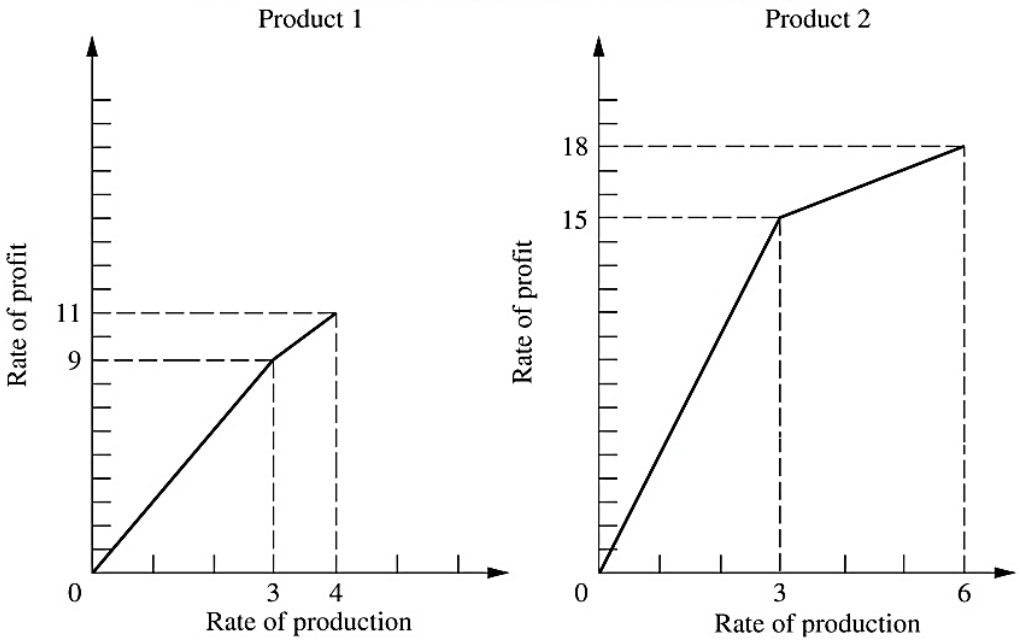
- Separable Programming
  - After obtaining an optimal solution for the model, you then can calculate

$$x_j = \sum_{k=1}^{n_j} x_{jk},$$

  - for $j = 1, 2, \ldots, n$ in order to identify an optimal solution for the original separable programming program or its piecewise linear approximation.

Nonlinear Programming

- Separable Programming
  - Example



$$\begin{aligned} \text{Maximize} \quad & Z = 3x_1 + 5x_2, \\ \text{subject to} \quad & x_1 \leq 4 \\ & 2x_2 \leq 12 \\ & 3x_1 + 2x_2 \leq 18 \\ & x_1 \geq 0, \quad x_2 \geq 0. \end{aligned}$$

Nonlinear Programming

- Separable Programming
  -

Maximize $Z = 3x_1 + 5x_2$,
subject to
$x_1 \le 4$
$2x_2 \le 12$
$3x_1 + 2x_2 \le 18$
$x_1 \ge 0, \quad x_2 \ge 0$

We now need to modify this model to fit the new situation described above. For this purpose, let the production rate for product 1 be $x_1 = x_{1R} + x_{1O}$, where x_{1R} is the production rate achieved on regular time and x_{1O} is the incremental production rate from using overtime. Define $x_2 = x_{2R} + x_{2O}$ in the same way for product 2. Thus, in the notation of the general linear programming model for separable programming given just before this example, $n = 2$, $n_1 = 2$, and $n_2 = 2$.

In particular, the new linear programming problem is to determine the values of x_{1R}, x_{1O}, x_{2R}, and x_{2O} so as to

Maximize $Z = 3x_{1R} + 2x_{1O} + 5x_{2R} + x_{2O}$,
subject to
$x_{1R} + x_{1O} \le 4$
$2(x_{2R} + x_{2O}) \le 12$
$3(x_{1R} + x_{1O}) + 2(x_{2R} + x_{2O}) \le 18$
$x_{1R} \le 3, \quad x_{1O} \le 1, \quad x_{2R} \le 3, \quad x_{2O} \le 3$
$x_{1R} \ge 0, \quad x_{1O} \ge 0, \quad x_{2R} \ge 0, \quad x_{2O} \ge 0$

Note that the upper bound constraints in the next-to-last row of the model make the first two functional constraints *redundant*, so these two functional constraints can be deleted.

Nonlinear Programming

- Separable Programming

However, there is one important factor that is not taken into account explicitly in this formulation. Specifically, there is nothing in the model that requires all available regular time for a product to be fully utilized before any overtime is used for that product. In other words, it may be feasible to have $x_{1O} > 0$ even when $x_{1R} < 3$ and to have $x_{2O} > 0$ even when $x_{2R} < 3$. Such solutions would not, however, be acceptable to management. (Prohibiting such solutions is the *special restriction* discussed earlier in this section.)

Now we come to the *key property of separable programming*. Even though the model does not take this factor into account explicitly, the model does take it into account implicitly! Despite the model's having excess ''feasible'' solutions that actually are unacceptable, any *optimal* solution for the model is *guaranteed* to be a legitimate one that does not replace any available regular-time work with overtime work.

Therefore, the simplex method can be safely applied to this model to find the most profitable acceptable product mix.

$$x_{1R} = 3, x_{1O} = 1, x_{2R} = 3, x_{2O} = 0$$

(x_1=4 and x_2=3) is the optimal solution

Nonlinear Programming- Advance

- Augmented Lagrange Multiplier Method
  - The Augmented Lagrange Multiplier Method combines the Lagrange multiplier and the penalty function methods. Consider the following equality constrained problem:

$$\max\ (or\ \min)\ z\ =\ f(x_1, x_2, \dots, x_n)$$
$$s.t. \quad g_1(x_1, x_2, \dots, x_n) \le b_1$$
$$s.t. \quad g_2(x_1, x_2, \dots, x_n) \le b_2$$
$$\dots$$
$$g_m(x_1, x_2, \dots, x_n) \le b_m$$

Nonlinear Programming- Advance

- Augmented Lagrange Multiplier Method
  - The Augmented Lagrange Multiplier Method combines the Lagrange multiplier and the penalty function methods. Consider the following equality constrained problem:

$$min\ f(x_1, x_2, \dots, x_n)$$
$$s.t.\ \ h_j(X) = 0; j = 1,2, \dots, p, p < n$$

  - The Lagrangian corresponding to the above equations is given by:

$$L(\mathbf{X},\boldsymbol{\lambda}) = f(\mathbf{X}) + \sum_{j=1}^{p} \lambda_j h_j(\mathbf{X})$$

where λ_j, j=1,2,…,p are the Lagrange multipliers.

Nonlinear Programming- Advance

- Augmented Lagrange Multiplier Method
  - The necessary conditions for a stationary point of $L(X,\lambda)$ include the equality constraints, equation

$$h_j(\mathbf{X}) = 0, \quad j = 1,2,\ldots,p, \quad p < n$$

  - The exterior penalty function approach is used to define the new objective function A(**X, λ,** r_k) termed the augmented Lagrangian function as

$$A(\mathbf{X},\boldsymbol{\lambda},r_k) = f(\mathbf{X}) + \sum_{j=1}^{p} \lambda_j h_j(\mathbf{X}) + r_k \sum_{j=1}^{p} h_j^2(\mathbf{X})$$

where r_k is the penalty parameter.

- It can be noted that the function A reduces to the Lagrangian if r_k=0 and to the ϕ function used in the classical penalty function method if all λ_j=0

Nonlinear Programming- Advance

- Augmented Lagrange Multiplier Method
  - It can be shown that if the Lagrange multipliers are fixed at their optimum values λ_j^*, the minimization of A(**X**, λ, r_k) gives the solution of the problem stated in the equations

$$min\ f(x_1, x_2, \dots, x_n)$$
$$s.t.\ \ h_j(X) = 0; j = 1,2, \dots, p, p < n$$

in one step for any value of r_k. In such a case, there is no need to minimize the function A for an increasing sequence of values of r_k. Since the values of λ_j^* are not known in advance, an iterative scheme is used to find the solution of the problem.

  - In the first iteration (k=1), the values of $\lambda_j^{(k)}$ are chosen as zero, the value of r_k is set equal to an arbitrary constant, and the function A is minimized with respect to **X** to find $\mathbf{X}^{*(k)}$. The values of $\lambda_j^{(k)}$ and r_k are then updated to start the next iteration.

Nonlinear Programming- Advance

- Augmented Lagrange Multiplier Method
  - For this, the necessary conditions for the stationary point of L, given by the equation

$$L(\mathbf{X},\boldsymbol{\lambda}) = f(\mathbf{X}) + \sum_{j=1}^{p} \lambda_j h_j(\mathbf{X})$$

are written as:

$$\frac{\partial L}{\partial x_i} = \frac{\partial f}{\partial x_i} + \sum_{j=1}^{p} \lambda_j^* \frac{\partial h_j}{\partial x_i} = 0, \qquad i = 1,2,\ldots,n$$

where λ_j^* denote the values of Lagrange multipliers at the stationary point of L.

Nonlinear Programming- Advance

- Augmented Lagrange Multiplier Method
  - Similarly, the necessary conditions for the minimum of A can be expressed as:

$$\frac{\partial A}{\partial x_i} = \frac{\partial f}{\partial x_i} + \sum_{j=1}^{p} (\lambda_j + 2r_k h_j) \frac{\partial h_j}{\partial x_i} = 0, \qquad i = 1,2,\ldots,n$$

  - A comparison of the right hand sides of the above two equations yield:

$$\lambda_j^* = \lambda_j + 2r_k h_j, \qquad j = 1,2,\ldots,p$$

Nonlinear Programming- Advance

- Augmented Lagrange Multiplier Method
  - These equations are used to update the values of λ_j as

$$\lambda_j^{(k+1)} = \lambda_j^{(k)} + 2r_k h_j(\mathbf{X}^{(k)}), \qquad j = 1,2,\ldots,p$$

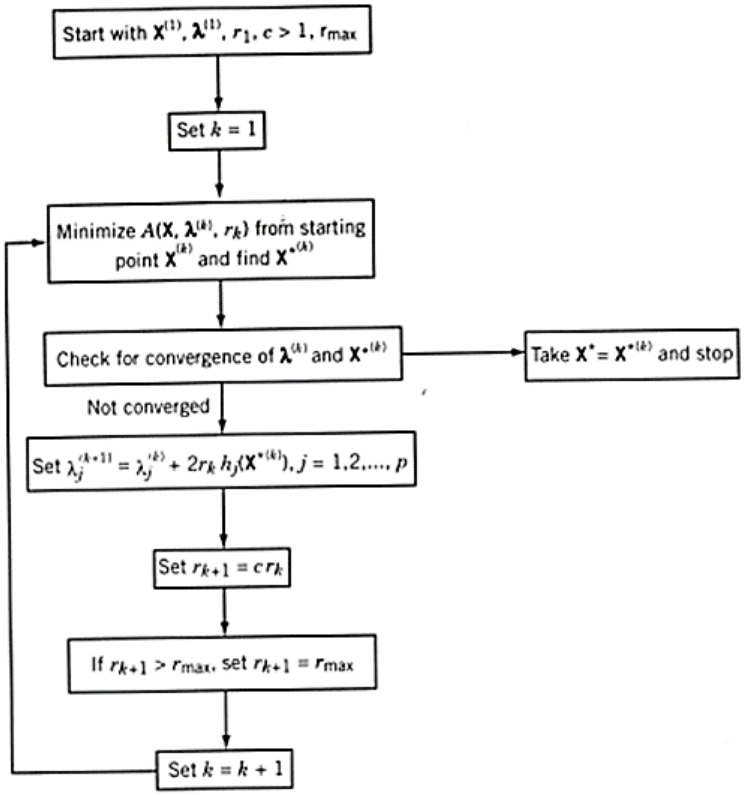
where $\mathbf{X}^{(k)}$ denote the starting vector used in the minimization of A. The value of r_k is updated as:

$$r_{k+1} = cr_k, \qquad c > 1$$

  - The function A is then minimized with respect to **X** to find $\mathbf{X}^{*(k+1)}$ and the iterative process is continued until convergence is achieved for $\lambda_j^{(k)}$ or **X*.** If the value of r_{k+1} exceeds a prespecified maximum value r_{max}, it is set equal to r_{max}.

Nonlinear Programming- Advance

- Augmented Lagrange Multiplier Method
  -



Nonlinear Programming- Advance

- Augmented Lagrange Multiplier Method
  - Inequality constrained problems
    - Consider the following inequality-constrained problem:

Minimize $f(\boldsymbol{X})$

Subject to $$g_j(\mathbf{X}) \leq 0, \quad j = 1,2,\ldots,m$$

- To apply the ALM method, the above inequality constraint is first converted to equality constraints as:

$$g_j(\mathbf{X}) + y_j^2 = 0, \quad j = 1,2,\ldots,m$$

where y_j^2 are the slack variables.

Nonlinear Programming- Advance

- Augmented Lagrange Multiplier Method
  - Then the augmented Lagrangian function is constructed as:

$$A(\mathbf{X},\boldsymbol{\lambda},\mathbf{Y},r_k) = f(\mathbf{X}) + \sum_{j=1}^{m} [g_j(\mathbf{X}) + y_j^2] + \sum_{j=1}^{m} r_k [g_j(\mathbf{X}) + y_j^2]^2$$

  - The vector of slack variables **Y** is given by:

$$\mathbf{Y} = \begin{Bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{Bmatrix}$$

  - If the slack variables y_j $j=1,2,\ldots,m$ are considered as additional unknowns, the function A is to be minimized with respect to **X** and **Y** for specified values of λ_j and r_k. This increases the problem size.

Nonlinear Programming- Advance

- Augmented Lagrange Multiplier Method
  - It can be shown that the function *A* given by the equation

$$A(\mathbf{X},\boldsymbol{\lambda},\mathbf{Y},r_k) = f(\mathbf{X}) + \sum_{j=1}^{m} [g_j(\mathbf{X}) + y_j^2] + \sum_{j=1}^{m} r_k [g_j(\mathbf{X}) + y_j^2]^2$$

$$A(\mathbf{X},\boldsymbol{\lambda},r_k) = f(\mathbf{X}) + \sum_{j=1}^{m} \lambda_j \alpha_j + r_k \sum_{j=1}^{m} \alpha_j^2$$

where

$$\alpha_j = \max \left\{ g_j(\mathbf{X}), -\frac{\lambda_j}{2r_k} \right\}$$

Nonlinear Programming- Advance

- Augmented Lagrange Multiplier Method
  - Thus the solution of the problem can be obtained by minimizing the function A given by

$$A(\mathbf{X},\boldsymbol{\lambda},r_k) = f(\mathbf{X}) + \sum_{j=1}^{m} \lambda_j \alpha_j + r_k \sum_{j=1}^{m} \alpha_j^2$$

as in the case of equality constrained problems using the update formula

$$\lambda_j^{(k+1)} = \lambda_j^{(k)} + 2r_k\alpha_j^{(k)}, \qquad j = 1,2,\ldots,m$$

Nonlinear Programming- Advance

- Augmented Lagrange Multiplier Method
  - It is to be noted that the function A given by the equation

$$A(\mathbf{X},\boldsymbol{\lambda},r_k) = f(\mathbf{X}) + \sum_{j=1}^{m} \lambda_j \alpha_j + r_k \sum_{j=1}^{m} \alpha_j^2$$

is continuous and has continuous first derivatives but has discontinuous second derivatives with respect to **X** at g_j **(X)=-** λ_j /2 r_k. Hence a second order method can not be used to minimize the function A

Nonlinear Programming- Advance

- Augmented Lagrange Multiplier Method
  - Mixed equality and inequality constrained problems
    - Consider the following general optimization problem:

$$\text{Minimize } f(\mathbf{X})$$

subject to

$$g_j(\mathbf{X}) \leq 0, \quad j = 1,2,\ldots,m$$
$$h_j(\mathbf{X}) = 0, \quad j = 1,2,\ldots,p$$

Nonlinear Programming- Advance

- Augmented Lagrange Multiplier Method
  - Mixed equality and inequality constrained problems
    - This problem can be solved by combining the procedures of the two preceding sections. The augmented Lagrangian function is defined as:

$$A(\mathbf{X},\boldsymbol{\lambda},r_k) = f(\mathbf{X}) + \sum_{j=1}^{m} \lambda_j \alpha_j + \sum_{j=1}^{p} \lambda_{m+j} h_j(\mathbf{X}) + r_k \sum_{j=1}^{m} \alpha_j^2 + r_k \sum_{j=1}^{p} h_j^2(\mathbf{X})$$

where α is given by:

$$\alpha_j = \max\left\{ g_j(\mathbf{X}), -\frac{\lambda_j}{2r_k} \right\}$$

Nonlinear Programming- Advance

- Augmented Lagrange Multiplier Method
  - Mixed equality and inequality constrained problems
    - The solution of the given problem above can be found by minimizing the function A as in the case of equality constrained problems using the update formula:

$$\lambda^{(k+1)} = \lambda_j^{(k)} + 2r_k \max\left\{g_j(\mathbf{X}), -\frac{\lambda_j^{(k)}}{2r_k}\right\}, \qquad j = 1,2,\ldots,m$$

$$\lambda_{m+j}^{(k+1)} = \lambda_{m+j}^{(k)} + 2r_k h_j(\mathbf{X}), \qquad j = 1,2,\ldots,p$$

Nonlinear Programming- Advance

- Augmented Lagrange Multiplier Method
  - The ALM method has several advantages. As stated earlier, the value of r_k need not be increased to infinity for convergence. The starting design vector $\mathbf{X}^{(1)}$ need not be feasible. Finally, it is possible to achieve $g_j(\mathbf{X})=0$ and $h_j(\mathbf{X})=0$ precisely and the nonzero values of the Lagrange multipliers ($\lambda_j \neq 0$) identify the active constraints automatically.

Nonlinear Programming- Advance

- Augmented Lagrange Multiplier Method
  - **Example**

$$\text{Minimize } f(\mathbf{X}) = 6x_1^2 + 4x_1x_2 + 3x_2^2$$

subject to

$$h(\mathbf{X}) = x_1 + x_2 - 5 = 0$$

using the ALM method.

Nonlinear Programming- Advance

- Augmented Lagrange Multiplier Method
  - The augmented Lagrangian function can be constructed as:

$$A(\mathbf{X},\lambda r_k) = 6x_1^2 + 4x_1x_2 + 3x_2^2 + \lambda(x_1 + x_2 - 5) + r_k(x_1 + x_2 - 5)^2$$

  - For the stationary point of A, the necessary conditions yield

$$\partial A/\partial x_i = 0,\ i = 1,2,$$

$$x_1(12 + 2r_k) + x_2(4 + 2r_k) = 10r_k - \lambda$$
$$x_1(4 + 2r_k) + x_2(6 + 2r_k) = 10r_k - \lambda$$

  - The solutions of the above equations yield:

$$x_1 = \frac{-90r_k^2 + 9r_k\lambda - 6\lambda + 60r_k}{(14 - 5r_k)(12 + 2r_k)}$$

$$x_2 = \frac{20r_k - 2\lambda}{14 - 5r_k}$$

Nonlinear Programming- Advance

- Augmented Lagrange Multiplier Method
  - Let the value of r_k be fixed at 1and select a value of $\lambda^{(1)}=0$. This gives

$$x_1^{*(1)} = -\frac{5}{21}, \quad x_2^{*(1)} = \frac{20}{9} \quad \text{with} \quad h = -\frac{5}{21} + \frac{20}{9} - 5 = -3.01587$$

  - For the next iteration

$$\lambda^{(2)} = \lambda^{(1)} + 2r_k h(\mathbf{X}^{*(1)}) = 0 + 2(1)\,(-3.01587) = -6.03175$$

  - Substituting this value of λ along with $r_k=1$ in x_1 and x_2:

$$x_1^{*(2)} = -0.38171, \quad x_2^{*(2)} = 3.56261$$

$$\text{with} \quad h = -0.38171 + 3.56261 - 5 = -1.81910$$

  - This procedure can be continued until some specified convergence is satisfied.

Nonlinear Programming- Advance

- Augmented Lagrange Multiplier Method

$\lambda^{(k)}$	r_k	$x_1^{*(k)}$	$x_2^{*(k)}$	Value of h
0.00000	1.00000	−0.23810	2.22222	−3.01587
−6.03175	1.00000	−0.38171	3.56261	−1.81910
−9.66994	1.00000	−0.46833	4.37110	−1.09723
−11.86441	1.00000	−0.52058	4.85876	−0.66182
−13.18806	1.00000	−0.55210	5.15290	−0.39919
−13.98645	1.00000	−0.57111	5.33032	−0.24078
−14.46801	1.00000	−0.58257	5.43734	−0.14524
−14.75848	1.00000	−0.58949	5.50189	−0.08760
−14.93369	1.00000	−0.59366	5.54082	−0.05284
−15.03937	1.00000	−0.59618	5.56430	−0.03187

Nonlinear Programming- Advance

- Augmented Lagrange Multiplier Method
  -

Nonlinear Programming- Advance

- Augmented Lagrange Multiplier Method
  -