MIT 2.853/2.854

Introduction to Manufacturing Systems

Optimization

Stanley B. Gershwin
Laboratory for Manufacturing and Productivity
Massachusetts Institute of Technology

Purpose of Optimization

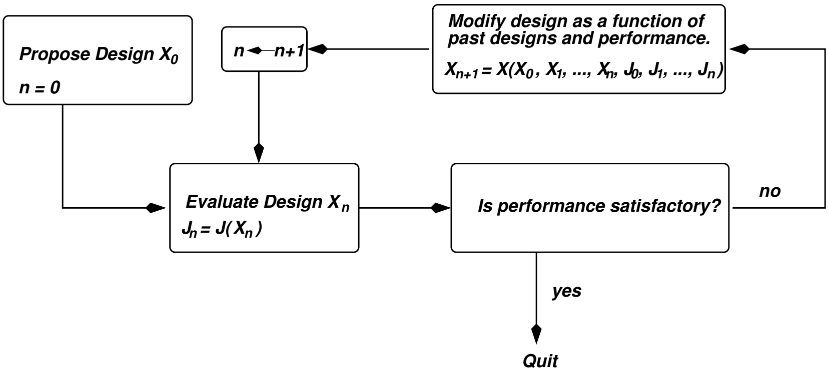
Choosing the *best* of a set of alternatives.

Applications:

- investment, scheduling, system design, product design, etc., etc.

Optimization is sometimes called *mathematical programming.*

Purpose of Optimization



Typically, many designs are tested.

Purpose of Optimization
Issues

- For this to be practical, total computation time must be limited. Therefore, we must control both *computation time per iteration* and *the number of iterations* .
- Computation time per iteration includes evaluation time and the time to determine the next design to be evaluated.
- The technical literature is generally focused on limiting the number of iterations by proposing designs efficiently.
- Reducing computation time per iteration is accomplished by
  - ⋆ using analytical models rather than simulations
  - ⋆ using coarser approximations in early iterations and more accurate evaluation later.

Problem Statement

X is a set of possible choices. J is a scalar function defined on X. h and g are vector functions defined on X.

Problem: Find $x \in X$ that satisfies

$J(x)$ is maximized *(or minimized)* — the *objective*

subject to

$h(x) = 0$ — *equality constraints*

$g(x) \leq 0$ — *inequality constraints*

Taxonomy

- static/dynamic
- deterministic/stochastic
- X set: continuous/discrete/mixed

(Extensions: multi-objective (or multi-criterion) optimization, in which there are multiple objectives that must somehow be reconciled; games, in which there are multiple optimizers, each choosing different xs.)

Continuous Variables

$X = R^n$. J is a scalar function defined on R^n. $h(\in R^m)$ and $g(\in R^k)$ are vector functions defined on R^n.

Problem: Find $x \in R^n$ that satisfies

$J(x)$ is maximized *(or minimized)*

subject to

$h(x) = 0$

$g(x) \leq 0$

Continuous Variables

One-dimensional search

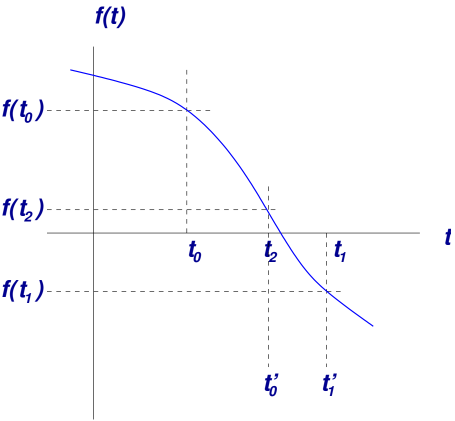
Motivation: Part of some optimization methods; also useful for other purposes.

Find t *such that* $f(t) = 0$.

- This is equivalent to
  Find t *to maximize (or minimize)* $F(t)$
  when $F(t)$ is differentiable, and $f(t) = dF(t)/dt$ is continuous.
- If $f(t)$ is differentiable, maximization or minimization is possible depending on the sign of $d^2F(t)/dt^2$.

Continuous Variables

One-dimensional search



Assume $f(t)$ is decreasing.

- *Binary search:* Guess t_0 and t_1 such that $f(t_0) > 0$ and $f(t_1) < 0$. Let $t_2 = (t_0 + t_1)/2$.
  - ⋆ If $f(t_2) < 0$, then repeat with $t_0' = t_0$ and $t_1' = t_2$.
  - ⋆ If $f(t_2) > 0$, then repeat with $t_0' = t_2$ and $t_1' = t_1$.

Continuous Variables

One-dimensional search

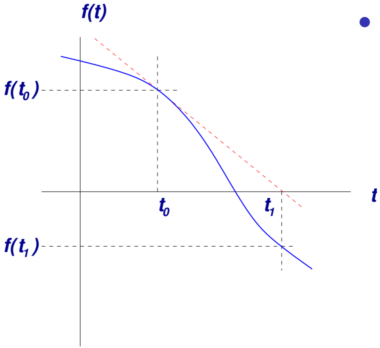
Example:

$$f(t) = 4 - t^2$$

t_0	t_2	$f(t_2)$	t_1
0	1.5	+	3
1.5	2.25	-	3
1.5	1.875	+	2.25
1.875	2.0625	-	2.25
1.875	1.96875	+	2.0625
1.96875	2.015625	-	2.0625
1.96875	1.9921875	+	2.015625
1.9921875	2.00390625	-	2.015625
1.9921875	1.998046875	+	2.00390625
1.998046875	2.0009765625	-	2.00390625
1.998046875	1.99951171875	+	2.0009765625
1.99951171875	2.000244140625	-	2.0009765625
1.99951171875	1.9998779296875	+	2.000244140625
1.9998779296875	2.00006103515625	-	2.000244140625
1.9998779296875	1.99996948242188	+	2.00006103515625
1.99996948242188	2.00001525878906	-	2.00006103515625
1.99996948242188	1.99999237060547	+	2.00001525878906
1.99999237060547	2.00000381469727	-	2.00001525878906
1.99999237060547	1.99999809265137	+	2.00000381469727
1.99999809265137	2.00000095367432	-	2.00000381469727

Continuous Variables

One-dimensional search



- *Newton search, exact tangent:*
  - ⋆ Guess t_0. Calculate $df(t_0)/dt$.
  - ⋆ Choose t_1 so that $f(t_0) + (t_1 - t_0)\frac{df(t_0)}{dt} = 0$.
  - ⋆ Repeat with $t_0' = t_1$ until $|f(t_0')|$ is small enough.

Continuous Variables

One-dimensional search

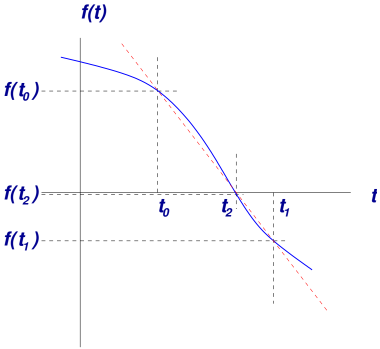
Example:

$f(t) = 4 - t^2$

t_0	$f(t_0)$
3	-
2.16666666666667	-
2.00641025641026	-
2.00001024002621	-
2.00000000002621	-
2	

Continuous Variables

One-dimensional search



- *Newton search, approximate tangent:*
  - ⋆ Guess t_0 and t_1. Calculate approximate slope $s = \frac{f(t_1)-f(t_0)}{t_1-t_0}$.
  - ⋆ Choose t_2 so that $f(t_0) + (t_2 - t_0)s = 0$.
  - ⋆ Repeat with $t_0' = t_1$ and $t_1' = t_2$ until $|f(t_0')|$ is small enough.

Continuous Variables

One-dimensional search

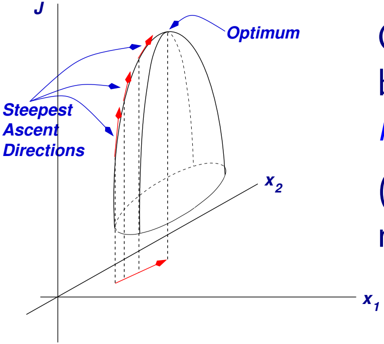
Example:

$$f(t) = 4 - t^2$$

t_0	$f(t_0)$
0	$+$
3	-
1.33333333333333	$+$
1.84615384615385	$+$
2.03225806451613	-
1.99872040946897	$+$
1.99998976002621	$+$
2.0000000032768	-
1.99999999999999	$+$
2	

Continuous Variables

Multi-dimensional optimization



Optimum often found by *steepest ascent* or *hill-climbing* methods.

(*Steepest descent* for minimization.)

Continuous Variables
Multi-dimensional optimization

To maximize $J(x)$, where x is a vector (and J is a scalar function that has *nice* properties):

0. Set $n = 0$. Guess x_0.
1. Evaluate $\frac{\partial J}{\partial x}(x_n)$.
2. Let t be a scalar. Define

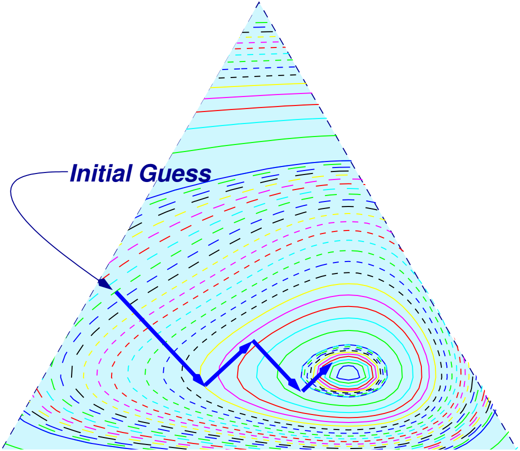
$$J_n(t) = J\left\{x_n + t\frac{\partial J}{\partial x}(x_n)\right\}$$

   Find (by *one-dimensional search*) $t_n^\star$, the value of t that maximizes $J_n(t)$.

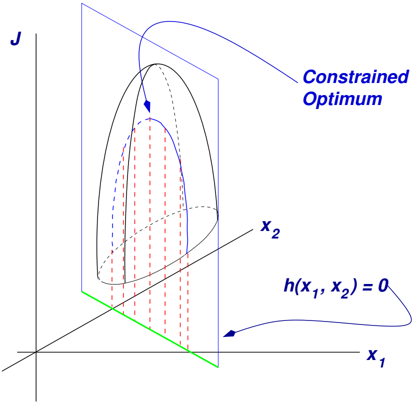
3. Set $x_{n+1} = x_n + t_n^\star \frac{\partial J}{\partial x}(x_n)$.
4. Set $n \leftarrow n + 1$. Go to Step 1.

Continuous Variables

Multi-dimensional optimization



Continuous Variables
Equality constrained



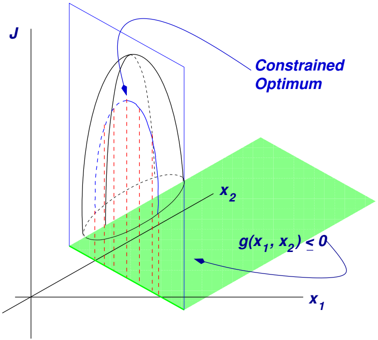
Equality constrained: solution is *on* the constraint surface.

Problems are much easier when constraint is linear, ie, when the surface is a plane.

- In that case, replace $\partial J/\partial x$ by its projection onto the constraint plane.
- *But first: find an initial <u>feasible</u> guess.*

Continuous Variables

Inequality constrained



Inequality constrained: solution is required to be on *one side of* the plane.

Inequality constraints that are satisfied with equality are called *effective* or *active* constraints.

If we knew which constraints would be effective, the problem would reduce to an equality-constrained optimization.

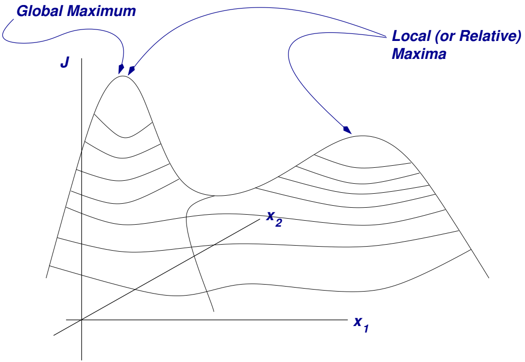
Continuous Variables

Inequality constrained

Optimization problems with continuous variables, objective, and constraints are called *nonlinear programming* problems, especially when at least one of J, h, g are not linear.

Continuous Variables

Inequality constrained



Danger: a search might find a *local,* rather than the *global,* optimum.

Continuous Variables

Inequality constrained

J is a scalar function defined on R^n. $h(\in R^m)$ and $g(\in R^k)$ are vector functions defined on R^n.

Problem: Find $x \in R^n$ that satisfies

$J(x)$ is minimized

subject to

$$h(x) = 0$$

$$g(x) \leq 0$$

See the "KKT Examples" notes.

Continuous Variables

Inequality constrained

- Let x^* be a local minimum.
- Assume all gradient vectors $\partial h_i/\partial x$, $\partial g_j/\partial x$, (where g_j is effective) are linearly independent (the *constraint qualification*).
- Then there exist vectors λ and μ of appropriate dimension ($\mu \geq 0$ component-wise) such that at $x = x^*$,

$$\begin{aligned} \frac{\partial J}{\partial x} + \lambda^T \frac{\partial h}{\partial x} + \mu^T \frac{\partial g}{\partial x} &= 0 \\ \mu^T g &= 0 \end{aligned}$$

Continuous Variables

Inequality constrained

The KKT conditions transform the optimization problem into a problem of simultaneously satisfying a set of equations and inequalities with additional variables (λ and μ):

$$\begin{aligned} h(x) &= 0 \\ g(x) &\leq 0 \\ \mu &\geq 0 \\ \frac{\partial J}{\partial x} + \lambda^T \frac{\partial h}{\partial x} + \mu^T \frac{\partial g}{\partial x} &= 0 \\ \mu^T g &= 0 \end{aligned}$$

Continuous Variables

Inequality constrained

There exist vectors $\lambda \in R^m$ and $\mu \in R^k$ ($\mu_j \geq 0$) such that at $x = x^*$,

$$\frac{\partial J}{\partial x_i} + \sum_{q=1}^{m} \lambda_q \frac{\partial h_q}{\partial x_i} + \sum_{j=1}^{k} \mu_j \frac{\partial g_j}{\partial x_i} = 0, \quad \text{for all } i = 1, ..., n,$$

$$\sum_{j=1}^{k} \mu_j g_j = 0$$

Note: The last constraint implies that

$$g_j(x^*) < 0 \quad \rightarrow \quad \mu_j = 0$$

$$\mu_j > 0 \quad \rightarrow \quad g_j(x^*) = 0.$$

Continuous Variables

Inequality constrained

Problem: In most cases, the KKT conditions are impossible to solve analytically. Therefore *numerical methods* are needed.

No general method is guaranteed to always work because "nonlinear" is too broad a category.

- *Specialized methods:* it is sometime possible to develop a solution technique that works very well for specific problems (eg, J quadratic, h, g linear).

Continuous Variables

Inequality constrained

- *Feasible directions:* Take steps in a feasible direction that will reduce the cost.
  - ⋆ Issue: hard to get the feasible direction when constraints are not linear. Some surfaces will be curved.
- *Gradient Projection:* project gradient onto the plane tangent to the constraint set. Move in that direction a short distance and then move back to the constraint surface.
  - ⋆ Issues: how short a distance? And how do you get back to the constraint surface?

Continuous Variables

Inequality constrained

- *Penalty Methods:*
  1. Transform problem into an unconstrained problem such as

     $$\min \bar{J}(x) = J(x) + KF(h(x), g(x))$$

     where $F(h(x), g(x))$ is positive if $h(x) \neq 0$ or any component of $g(x)$ is positive.
  2. Solve the problem with small positive K and then increase K. The solution for each K is a starting guess for the problem with the next K.

  ⋆ Issues: Intermediate solutions are usually not feasible; and problem gets hard to solve as K increases.

Continuous Variables

Inequality constrained

- There is much software available for optimization. However, *use it with care!!* There are always problems that can defeat any given method. If you use such software, *don't assume that the answer is correct.* **Check it!!!**

    - ⋆ Look at it carefully. Make sure it is intuitively reasonable.
    - ⋆ Do a sensitivity analysis. Vary parameters by a little bit and make sure the solution changes by a little bit. If not, *find out why!*

Linear Programming

- *Definition:* A special case of nonlinear programming in which the objective and the constraints are all linear.
- Many practical applications.
- Efficient solution techniques are available that exploit the linearity.
- Software exists for very large problems.

Linear Programming
Example

Two machines are available 24 hours per day. They are both required to make each of two part types. No time is lost for changeover. The times (in hours) required are:

Part	Machine 1	Machine 2
1	1	2
2	3	4

What is the maximum number of Type 1's we can make in 1000 hours given that the parts are produced in a ratio of 2:1?

Linear Programming
Formulation

Let U_1 be the number of Type 1's produced and let U_2 be the number of Type 2's. Then the number of hours required of Machine 1 is

$$U_1 + 3U_2$$

and the number of hours required of Machine 2 is

$$2U_1 + 4U_2$$

and both of these quantities must be less than 1000.
Also,

$$U_1 = 2U_2.$$

Linear Programming
Formulation

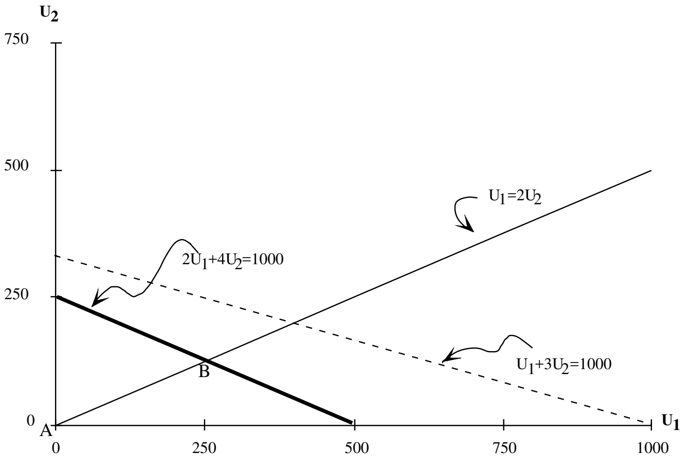
Or,

$$\max U_1$$

subject to

$$\begin{aligned} U_1 + 3U_2 &\leq 1000 \\ 2U_1 + 4U_2 &\leq 1000 \\ U_1 &= 2U_2 \\ U_1 \geq 0; \quad & U_2 \geq 0 \end{aligned}$$

Linear Programming
Graphical representation



Linear Programming
Canonical form — subscript notation

Let $x \in R^n, A \in R^{m \times n}, b \in R^m, c \in R^n$.

$$\min_x \sum_{j=1}^{n} c_j x_j$$

subject to

$$\sum_{j=1}^{n} a_{ij} x_j = b_i, i = 1, \ldots, m$$

$$x_j \geq 0, j = 1, \ldots, n$$

Linear Programming
Canonical form — matrix/vector notation

Or,

$$\min_x c^T x$$

subject to

$$Ax = b$$

$$x \geq 0$$

Here, $\geq$ is interpreted component-wise.

This is the *standard* or *canonical form* of the LP.

Linear Programming
Example

All LPs can be expressed in this form. The example can be written

$$\min(-1)U_1$$

subject to

$$\begin{aligned} U_1 + 3U_2 + U_3 &= 1000 \\ 2U_1 + 4U_2 + U_4 &= 1000 \\ U_1 - 2U_2 &= 0 \\ U_1 \geq 0, U_2 \geq 0, U_3 &\geq 0, U_4 \geq 0 \end{aligned}$$

in which U_3 and U_4 are *slack variables* . Here, they represent the idle times of Machine 1 and Machine 2.

Linear Programming
Slack variables

To put an LP in equivalent canonical form: for every constraint of the form

$$\sum_{j=1}^{n} a_{ij} x_j \leq b_i$$

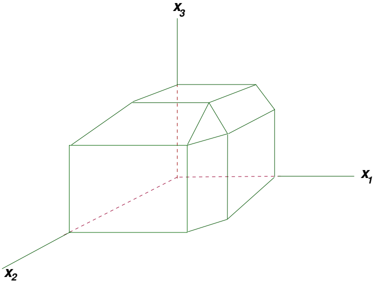
define a new variable x_k and replace this constraint with

$$\sum_{j=1}^{n} a_{ij} x_j + x_k = b_i$$

$$x_k \geq 0$$

Linear Programming
Graphical representation

For this constraint set,



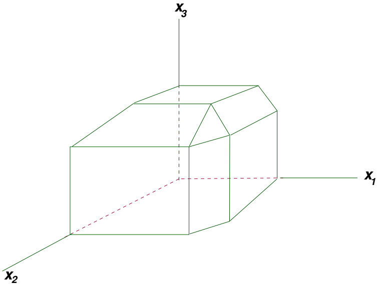
there are 3 variables, no equality constraints, and (at least) 7 inequality constraints (not counting $x_i \geq 0$).

The LP can be transformed into one with 10 variables, (at least) 7 equality constraints, and no inequalities (except for $x_i \geq 0$).

Why "at least"?

Linear Programming

Graphical representation



This set is called a *polyhedron* or a *simplex* .

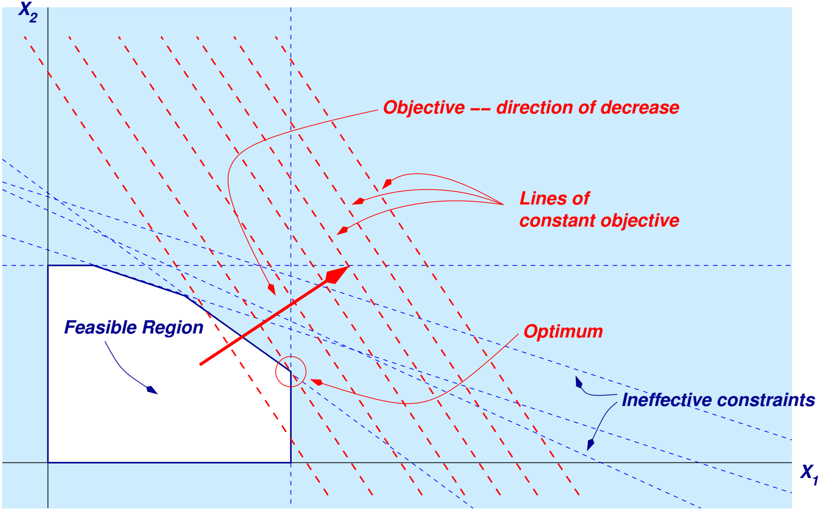
Linear Programming
Definitions

If x satisfies the constraints, it is a *feasible solution* .

If x is feasible and it minimizes $c^T x$, it is an *optimal feasible solution* .

Linear Programming

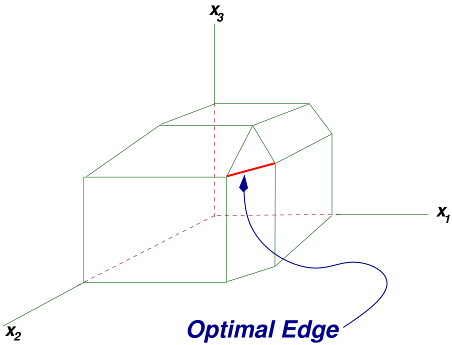
Graphical representation

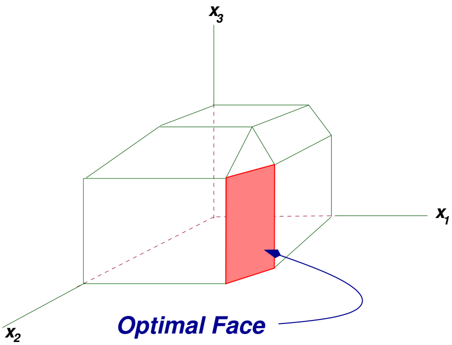


Linear Programming
Cases

- Problem could be *infeasible* — no feasible set — no solution.
- Feasible set could be unbounded.
  - ⋆ Minimum of objective could be unbounded ($-\infty$) — infinite solution.
- Effective constraints could be non-independent — adds complexity to the solution technique.
- c vector could be orthogonal to the boundary of the feasible region — infinite number of solutions.

Linear Programming
Cases





Linear Programming
Basic Solutions

Assume that there are more variables than equality constraints (that $n > m$) and that matrix A has rank m.

Let A_B be a matrix which consists of m columns of A. It is square ($m \times m$). Choose columns such that A_B is invertible.

Then A can be written

$$A = (A_B, A_N)$$

in which A_B is the *basic part* of A. The *non-basic part,* A_N, is the rest of A.

Correspondingly, $x = \begin{pmatrix} x_B \\ x_N \end{pmatrix}$.

Linear Programming
Basic Solutions

Then $Ax = A_B x_B + A_N x_N = b$,

or $x_B = A_B^{-1}(b - A_N x_N)$.

Suppose $x_N = 0$. Then $x_B = A_B^{-1} b$.

If $x_B = A_B^{-1} b \geq 0$ then $x = \begin{pmatrix} A_B^{-1} b \\ 0 \end{pmatrix}$ is feasible and x is a *basic feasible solution.*

- Geometrically: basic feasible solutions are *corners* of the constraint set. Each corner corresponds to a different A_B.

Linear Programming
The Fundamental Theorem

- If there is a feasible solution, there is a *basic* feasible solution.

- If there is an optimal feasible solution, there is an optimal *basic* feasible solution.

Linear Programming
The Simplex Method

- Since there is always a solution at a corner (when the problem is feasible and there is a bounded solution), search for solutions only on corners.

- At each corner, determine which adjacent corner improves the objective function the most. Move there. Repeat until no further improvement is possible.

- Moving to an adjacent corner is equivalent to interchanging one of the columns of A_B with one of the columns of A_N.

Linear Programming
Reduced Cost

Choose a feasible basis. The LP problem can be written

$$\min c_B^T x_B + c_N^T x_N$$

subject to

$$A_B x_B + A_N x_N = b$$

$$x_B \geq 0, x_N \geq 0$$

We can solve the equation for x_B and get

$$x_B = A_B^{-1}(b - A_N x_N)$$

Linear Programming
Reduced Cost

If we eliminate x_B, the problem is

$$\min \left(c_N^T - c_B^T A_B^{-1} A_N \right) x_N$$

subject to

$$\begin{array}{rcl} A_B^{-1} A_N x_N & \leq & A_B^{-1} b \\ x_N & \geq & 0 \end{array}$$

This is an LP (although not in standard form). For $x_N = 0$ to be a *feasible solution,* we must have

$$x_B = A_B^{-1} b \geq 0$$

Linear Programming
Reduced Cost

Define the *reduced cost* $c_R^T = c_N^T - c_B^T A_B^{-1} A_N$. If all components of c_R are non-negative, $x_N = 0$ is *optimal.*

Very simplified explanation of the simplex method:

- Move to an adjacent corner by taking one variable out of the basis and replacing it by one not currently in the basis.
- Add to the basis the column corresponding to the most negative element of c_R.
- Determine which element of the basis would decrease the cost most if it replaced by the new column.
- Stop when no elements of c_R are negative.

Linear Programming
Reduced Cost

Note: if some elements of c_R are 0 and the rest are positive, there are many solutions.

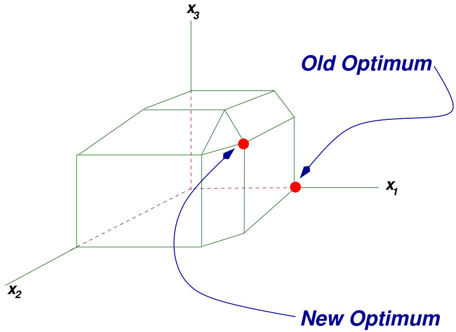
Linear Programming
Sensitivity Analysis

Suppose A, b, or c change by a little bit to A', b', and c'. Then the optimal solution may change. Cases:

- The basic/non-basic partition remains optimal. That is, the reduced cost vector based on the old partition remains all non-negative. The solution changes by a little bit.
- Some elements of the reduced cost go to 0. In that case, there are many solutions.

Linear Programming
Sensitivity Analysis

- Some elements of the reduced cost vector (according to the current partition) become negative. In that case, the basis must change and the solution moves to a new corner. This could mean there is a large change in the solution.



Linear Programming

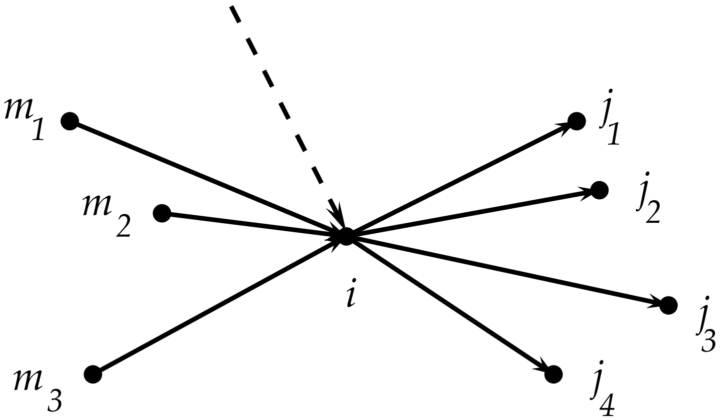
Shadow price

If the optimal value of the LP is $J = c^T x^*$, the *shadow price* of constraint j is

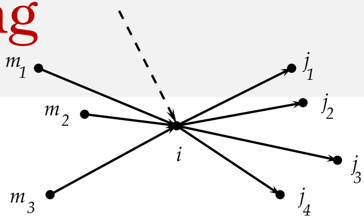
$$\frac{\partial J}{\partial b_j}$$

Interpretation: You should be willing to pay $\dfrac{\partial J}{\partial b_j}\delta b_j$ to increase the right hand side b_j of constraint j by δb_j.

Linear Programming
Network Problems

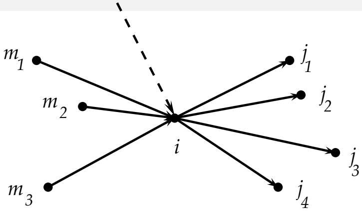


Linear Programming
Network Problems



- Let b_i^k be the flow introduced at node i destined for node k.
- Let x_{ij}^k be the flow on link (i,j) destined for node k. $x_{ij}^k = 0$ if there is no direct link from i to j.
- Let c_{ij}^k be the cost per unit of flow on link (i,j) for flow destined for node k. $c_{ij}^k = \infty$ if there is no direct link from i to j.

Linear Programming
Conservation of flow



Flow into a node = flow out of the node.

$$\sum_{m \neq i} x_{mi}^k + b_i^k = \sum_{j \neq i} x_{ij}^k \text{ for } i \neq k$$

Linear Programming
Network LP

$$\min \sum_{i,j,k} c_{ij}^k x_{ij}^k$$

$$\sum_{m \neq i} x_{mi}^k + b_i^k = \sum_{j \neq i} x_{ij}^k \text{ for all } j, k; \text{ for all } i \neq k$$

$$x_{ij}^k \geq 0 \text{ for all } i, j, k$$

Dynamic Programming

- Optimization over time.
    - ⋆ Decisions made now can have costs or benefits that appear only later, or might restrict later options.

- Deterministic or stochastic.

- *Examples:* investment, scheduling, aerospace vehicle trajectories.

- *Elements:* state, control, objective, dynamics, constraints.

Dynamic Programming
Special Class of NLPs

Objective: $J(x(0)) =$

$$\min_{\substack{u(i),\\ 0\leq i\leq T-1}} \sum_{i=0}^{T-1} L(x(i), u(i)) + F(x(T))$$

such that

Dynamics: $x(i+1) = f(x(i), u(i), i);$ $\quad x(0)$ specified

Constraints: $h(x(i), u(i)) = 0; \quad g(x(i), u(i)) \leq 0.$

Dynamic Programming
Special Class of NLPs

Objective:
$$J(x(0)) = \min_{\substack{u(t),\\ 0 \le t \le T}} \int_0^T g(x(t), u(t))dt + F(x(T))$$

such that

Dynamics:
$$\frac{dx(t)}{dt} = f(x(t), u(t), t); \qquad x(0) \text{ specified}$$

Constraints:
$$h(x(t), u(t)) = 0; \quad g(x(t), u(t)) \le 0.$$

Other topics

- Integer programming/combinatorial optimization
- Stochastic dynamic programming
- Heuristics and meta-heuristics

