Probabilistic Algorithms

Paul Cotofrei

information management institute

PA 2016

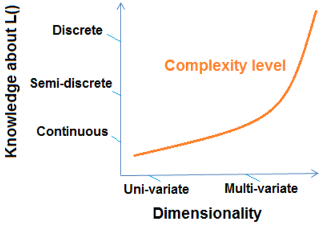
Outline

Deterministic search and optimization

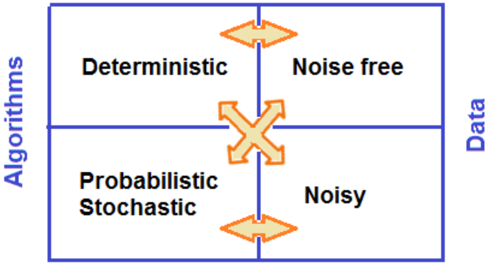
- Optimization for simple problems
- Optimization for complex problems

Complexity of optimization problems

- Low complexity = easy problem; high complexity = hard problem
- Several factors influence the "degree of complexity" of an optimization problem
  - Dimension of domain **D**: low dimension -> low complexity
  - Knowledge about the structure of energy landscape (image of $L(\mathbf{d})$): more knowledge (i.e. gradient) -> less complexity
  - Shape of domain **D**: non-convex -> high complexity



Algorithms and Data for Optimization problems



Outline

Direct Search

Hypothesis

- One dimensional continuous function
- $D = [a, b]$ (convex domain)
- $L(\cdot)$ can be evaluated in any point d

Direct Search methods : use only loss function values to locate the minimum point

- **Exhaustive Search** - use a sequence of equally spaced points

Step 1 Set n the number of intermediate points and calculate $\delta = (b - a)/n$

Step 2 Find the point(s) $d_i = a + (i - 1)\delta$, $i = 0..n$ minimizing $L(d_i)$

Bracketing Methods

Supplementary knowledge: there is a single minima of $L()$ in $[a, b]$, i.e. $L()$ is unimodal

Idea : find a lower and an upper bound of an interval containing the minimum

Step 1 Set n the number of intermediate points and calculate $\delta = (b - a)/n$

Step 2 Set $d_1 = a$, $d_2 = d_1 + \delta$, $d_3 = d_2 + \delta$

Step 3 If $L(d_1) \geq L(d_2) \leq L(d_3)$

THEN return (d_1, d_3)

ELSE $d_1 = d_2$, $d_2 = d_3$, $d_3 = d_2 + \delta$

Step 4 If $d_3 \leq b$

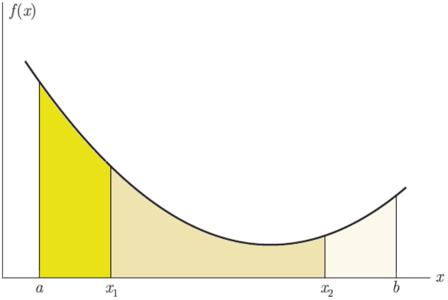
THEN goto Step 3

ELSE ?

Region-Elimination Methods

- Improve the accuracy of the solution when the (unique) global minimum belongs to (a, b)
- Let be $d_1 < d_2$ two points in (a, b).

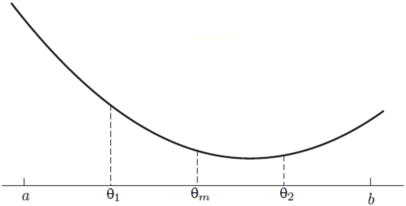
  The fundamental rules for region-elimination methods are:
  - **R1**: if $L(d_1) > L(d_2)$ then the minimum does not lie in (a, d_1)
  - **R2**: if $L(d_1) < L(d_2)$ then the minimum does not lie in (d_2, b)
  - **R3**: if $L(d_1) = L(d_2)$ then the minimum does not lie in (a, d_1) and (d_2, b)



A typical single-variable unimodal function with function values at two distinct points.

Bisection method

- Three points d_1, d_m, d_2 are chosen in the interval (a, b)
- The search space (i.e. (a, b)) is divided in four regions
- The region-elimination rules are used to eliminate a portion of the search space based on function values at the three chosen points
- If all the points $\{a, d_1, d_m, d_2, b\}$ are equidistant, then the eliminated portion represents 50% of the search space
- End condition: the length of the search space is less than a given ϵ



Three points θ_1, θ_m, and θ_2 used in the interval halving method.

An implementation

Step 1 Set $\Delta = b - a$, $d_m = (a + b)/2$ and calculate $L(d_m)$; Set the precision value ε

Step 2 Set $d_1 = a + \Delta/4$, $d_2 = b - \Delta/4$; Calculate $L(d_1)$ and $L(d_2)$

Step 3 If $L(d_1) < L(d_m)$ THEN
$b = d_m$, $d_m = d_1$; Goto Step 5

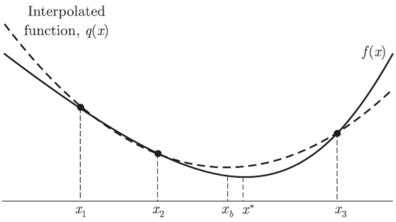
Step 4 If $L(d_2) < L(d_m)$ THEN
$a = d_m$, $d_m = d_2$; Goto Step 5
ELSE $a = d_1$, $b = d_2$, $d_m = (a + b)/2$;

Step 5 $\Delta = b - a$; If $\Delta \geq \varepsilon$ THEN Goto Step 2
ELSE return (a, b)

Remark: The best convergence rate is obtained if the length ratio between two successive search spaces (a, b) is $(1 + \sqrt(5))/2$ (known as golden number).

Successive Quadratic Estimation Method

- For uni-modal function, only the relative function values $L(d_i)$ were used to guide the search
- A new piece of knowledge: a quadratic polynomial is also an uni-modal function
- Idea:
  - use the magnitude and sign of function values to estimate the quadratic polynomial fitting these points
  - take the minimum of the fitted function as an estimate of the minimum of $L()$
  - To determine the quadratic polynomial, only three points are necessary



Mathematical background

- Consider three points $(d_1, L(d_1))$, $(d_2, L(d_2))$ and $(d_3, L(d_3))$ in R^2, with $d_1 < d_2 < d_3$
- The minimum point $\bar{d}$ for the quadratic polynomial determined by these points is defined by

$$\bar{d} = \frac{d_1 + d_2}{2} - \frac{a_1}{2a_2}$$

where

$$a_1 = \frac{L(d_2) - L(d_1)}{d_2 - d_1}$$

and

$$a_2 = \frac{1}{d_3 - d_2}\left(\frac{L(d_3) - L(d_1)}{d_3 - d_1} - a_1\right)$$

Powell's Algorithm

Step 1 Set Δ the step size, $d_1 = (a+b)/2$ and $d_2 = d_1 + \Delta$

Step 2 If $L(d_1) > L(d_2)$ THEN $d_3 = d_1 + 2\Delta$ ELSE $d_3 = d_1 - \Delta$

Step 3 Calculate $L_{min} = \min(L(d_1), L(d_2), L(d_3))$ and d_{min} the point d_i that corresponds to L_{min}

Step 5 Use the points d_1, d_2 and d_3 to calculate $\bar{d}$

Step 6 If $|L_{min} - L(\bar{d})| > \varepsilon$ THEN

- Among the four points $(d_1, d_2, d_3, \bar{d})$ save the best point and two bracketing it, if possible; otherwise, save the best three points. Relabel them according to $d_1 < d_2 < d_3$ and goto Step 3

ELSE d^* is the best of current four points

Gradient-based Methods

- The loss function $L()$ is continuous and defined on R^n, $n \geq 1$
- New information about $L()$: the gradient information is available
- The gradient $g(\mathbf{d})$ of $L(\mathbf{d})$ is the vector of first order derivatives

$$g(\mathbf{d}) = \frac{\partial L(\mathbf{d})}{\partial \mathbf{d}} = \begin{bmatrix} \frac{\partial L(\mathbf{d})}{\partial d_1} \\ \frac{\partial L(\mathbf{d})}{\partial d_2} \\ \vdots \\ \frac{\partial L(\mathbf{d})}{\partial d_n} \end{bmatrix}$$

where $\mathbf{d} = [d_1, ..., d_n]^T$

  - First-order condition for unconstrained local minimization problem: $g(\mathbf{d}^*) = 0$ (optimization problem type P2)

Gradient-based Methods

- Hessian matrix $\mathbf{H}(\mathbf{d})$ of $L(\mathbf{d})$ is the matrix of second order derivatives

$$\mathbf{H}(\mathbf{d}) = \frac{\partial^2 L(\mathbf{d})}{\partial \mathbf{d} \partial \mathbf{d}^T} = \begin{pmatrix} \frac{\partial^2 L(\mathbf{d})}{\partial d_1 \partial d_1} & \frac{\partial^2 L(\mathbf{d})}{\partial d_1 \partial d_2} & \cdots & \frac{\partial^2 L(\mathbf{d})}{\partial d_1 \partial d_n} \\ \frac{\partial^2 L(\mathbf{d})}{\partial d_2 \partial d_1} & \frac{\partial^2 L(\mathbf{d})}{\partial d_2 \partial d_2} & \cdots & \frac{\partial^2 L(\mathbf{d})}{\partial d_2 \partial d_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 L(\mathbf{d})}{\partial d_n \partial d_1} & \frac{\partial^2 L(\mathbf{d})}{\partial d_n \partial d_2} & \cdots & \frac{\partial^2 L(\mathbf{d})}{\partial d_n \partial d_n} \end{pmatrix}$$

- Hessian matrix useful in characterizing the shape of L and to distinguish, among the zeros of gradient $g(\mathbf{d})$, the minimum points, the maximum points and the inflexion points;

Gradient and Hessian in practice

- Calculate the gradient of
  - $L(t) = t\sin(2t + \pi)$
  - $L(\theta) = \frac{2}{3}t_1^3 - 4t_1t_2 + 2t_2^2 - 3t_1 + t_2 + 6$, where $\theta = (t_1, t_2)^T$
  - $L(\theta) = t_1^2 + 2t_2^2 + 3t_3^2 + 3t_1t_2 + 4t_1t_3 - 3t_2t_3$, where $\theta = (t_1, t_2, t_3)^T$
- Calculate the Hessian of
  - $L(\theta) = (t_1 - 2)^4 + (t_1 - 2t_2)^2$
  - $L(\theta) = t_1^2 + 2t_2^2 + 3t_3^2 + 3t_1t_2 + 4t_1t_3 - 3t_2t_3$

Steepest Descent Update

- Descend direction: a vector $\mathbf{v} \in R^n$ such that $g(\mathbf{d})^T \cdot \mathbf{v} < 0$
- If $\mathbf{v}$ descend direction, then $L(\mathbf{d} + \lambda \mathbf{v}) < L(\mathbf{d})$, for any $\lambda \in (0, u), u > 0$
- Steepest descent direction: $\mathbf{v} = -g(\mathbf{d})$
- Recursion: $\hat{\mathbf{d}}_{k+1} = \hat{\mathbf{d}}_k - a_k g(\hat{\mathbf{d}}_k)$
- $a_k > 0$ is the *step size*
  1. a_k is constant;
  2. a_k is the minimum of the function $h(a) = L(\hat{\mathbf{d}}_k - a g(\hat{\mathbf{d}}_k))$;
  3. a_k is a predefined sequence depending only on the index k
- Stopping criteria: $||\hat{\mathbf{d}}_{k+1} - \hat{\mathbf{d}}_k|| \leq \varepsilon_1$, $||g(\hat{\mathbf{d}}_k|| \leq \varepsilon_2$ or reaching maximum number of iterations M

Cauchy's Steepest Descent Method

Step 1 Choose M, ε_1, ε_2 and $\hat{\mathbf{d}}_0$; Set $k = 0$

Step 2 Calculate $g(\hat{\mathbf{d}}_k)$;

IF $||g(\hat{\mathbf{d}}_k|| \leq \varepsilon_1$ or $k \geq M$ return $\hat{\mathbf{d}}_k$;

Step 3 Consider the new loss function $h(\alpha) = L(\hat{\mathbf{d}}_k - \alpha g(\hat{\mathbf{d}}_k))$ defined on $[0, b)$ and estimate the minimum $\alpha^* = a_k$ using any optimization method for univariate and unimodal functions

- one stopping criterion for this subproblem is $|g(\hat{\mathbf{d}}_k - \alpha^* g(\hat{\mathbf{d}}_k))g(\hat{\mathbf{d}}_k)| \leq \varepsilon_2$

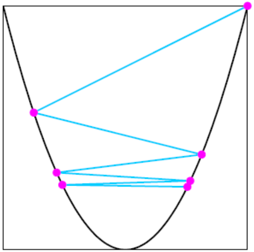
Step 4 Calculate $\hat{\mathbf{d}}_{k+1} = \hat{\mathbf{d}}_k - a_k g(\hat{\mathbf{d}}_k)$.

If $\frac{||\hat{\mathbf{d}}_{k+1} - \hat{\mathbf{d}}_k||}{||\hat{\mathbf{d}}_k||} > \varepsilon_1$ THEN $k = k + 1$; goto Step 2

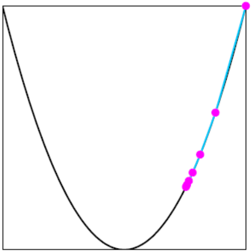
ELSE return $\hat{\mathbf{d}}_{k+1}$

Convergence issues

- To avoid the second optimisation problem (for linear function $h(\alpha)$, the step size a_k may be defined in advance.
- But a sequence of estimates $\hat{\mathbf{d}}_k$ such that $L(\hat{\mathbf{d}}_{k+1}) < L(\hat{\mathbf{d}}_k)$ is not guarantee to converge towards the minimum θ^*
- The steps a_k may be "too long" or "too shorts"



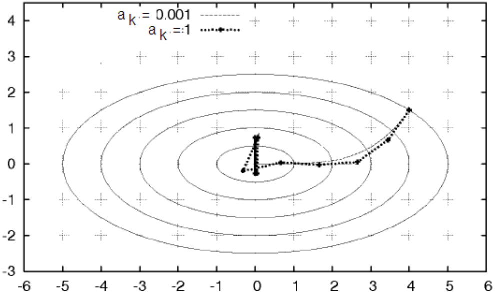
(a) Steps are too long.



(b) Steps are too short.

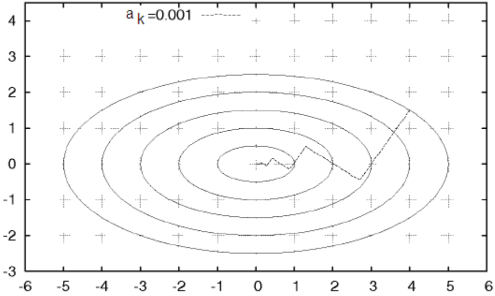
- There exist specific conditions to prevent too long or too short steps.

Steepest Descent in multiple dimensions



- Long steps ($a_k = 1$):
  - the vicinity of θ^* is reached very fast;
  - but then the estimates oscillate around the minimum;
- Small steps ($a_k = 0.001$)
  - long time to reach the vicinity of θ^*
  - may freeze near θ^*

Steepest Descent in multiple dimensions



- If the cost of calculating $g(\hat{\mathbf{d}}_k)$ is too expensive
  1. determine a direction of search from one local gradient calculation ($g(\hat{\mathbf{d}}_0)$)
  2. search along this direction until a local minimum is found (i.e., until $L(\hat{\mathbf{d}}_{k+1}) > L(\hat{\mathbf{d}}_k)$)
  3. calculate a new local gradient (i.e. $g(\hat{\mathbf{d}}_k)$)
  4. search along these new direction a second local minimum
  5. repeat until the gradient is sufficiently close to zero

Newton-Raphson Method

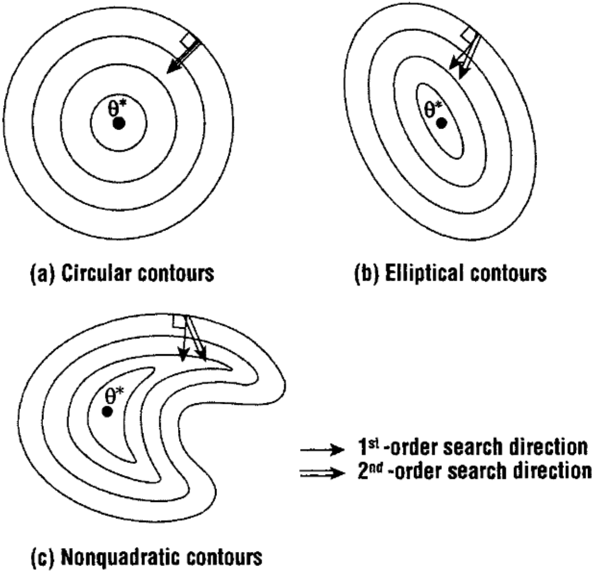
Remark: Steepest descent method is sensitive to transformations and scaling.

- Use Hessian matrix for scaling:

$$\hat{\theta}_{k+1} = \hat{\theta}_k - H(\hat{\theta}_k)^{-1} g(\hat{\theta}_k), k = 0, 1, 2, ...$$

- If L quadratic, the algorithm converges in one step !
- Modifications to the basic algorithm need for nonlinear loss functions having H not positive definite at **d** away from $\mathbf{d}^*$
- Algorithm is transform invariant and unaffected by large scaling difference in the **d** elements.

Behavior of Steepest Descent and Newton-Raphson



Relative Convergence Rates

- Theoretical analysis based on convergence rates of iterates $\hat{\mathbf{d}}_k$ where k is iteration counter
- Let $\mathbf{d}^*$ represent optimal value of $\mathbf{d}$
- For deterministic optimization, a standard rate result is:

$$\|\hat{\mathbf{d}}_{k+1} - \mathbf{d}^*\| = O(\|\hat{\mathbf{d}}_k - \mathbf{d}^*\|^c), c \geq 1$$

  - Steepest descent: $c = 1$
  - Newton-Raphson: $c = 2$
- Stochastic rate inherently slower in theory and practice

Outline

Deterministic search and optimization

Optimization for simple problems

Optimization for complex problems

Exact optimization for complex problems

- Complex problem
  - Constrained domain $\mathbf{D} \in R^n$
  - Discrete domain $\mathbf{D} \in R^n$
- Linear problem: $L(\mathbf{d}) = \sum_{i=1}^{n} c_i d_i + \vartheta = \mathbf{c}^T \mathbf{d} + \vartheta$, under the constraints $d_i \geq 0$ and $\mathbf{Ad} \leq \mathbf{b}$, with $\mathbf{A} \in R^{m \times n}, \mathbf{d} \in R^n, \mathbf{b} \in R^m, \mathbf{b} > 0$
- $L()$ also called *objective function*; $\{d_1, ..., d_n\}$ also called *decision variables*
  - Min $L(d_1, d_2) = 3d_1 - 4d_2 + 12$, under the constraints

    $$\begin{cases} 6d_1 + d_2 & \leq & 10 \\ -d_1 + 3d_2 & \leq & 4 \\ -3d_1 - d_2 & \leq & 9 \end{cases}$$
  - $\mathbf{A}_{3\times 2} = \begin{pmatrix} 6 & 2 \\ -1 & 3 \\ -3 & -1 \end{pmatrix}, \mathbf{b} = (10, 4, 9)^T, \mathbf{c} = (3, -4), \vartheta = 12$
- **SIMPLEX** algorithm (1945, Danzig)
- The constraints determine a multidimensional polytope (an n simplex). It can be proved mathematically that the optimal solution $\mathbf{d}^*$ (if exists) is one of the vertex of the simplex.

Basic concepts and principles

- Standard form: $\min(\mathbf{c}^T\mathbf{d})$ under the constraints
  - $\mathbf{Ad} = \mathbf{b}$ and $\mathbf{d} \geq 0$, where
  - $\mathbf{A} = M_{m\times(m+n)}, \mathbf{b} \in R^m, \mathbf{b} > 0, \mathbf{d} \in R^{m+n}$ and $rank(\mathbf{A}) = m$
- Consider m linearly independent columns of matrix $\mathbf{A}$. Denote $\mathbf{B}$ the matrix formed by these columns (so $det(\mathbf{B}) \neq 0$), and $\mathbf{N}$ the matrix formed by all other columns. Re-index the coordinates of $\mathbf{d}$ in order to re-position the columns of $\mathbf{A}$ such that $\mathbf{A} = [\mathbf{B}, \mathbf{N}]$. Denote $\mathbf{d}_B$ and $\mathbf{d}_N$ the corresponding sub-vectors ($\mathbf{d} = [\mathbf{d}_B, \mathbf{d}_N]$).

Th. 1 If the linear problem has a finite optimal solution $\mathbf{d}^*$ then $\mathbf{d}^*$ belongs to the set of extreme points (the vertices) of the simplex Θ.

Th. 2 Any extreme point of the simplex corresponds to a solution $\mathbf{d} = [\mathbf{B}^{-1}\mathbf{b}, \mathbf{0}]^T$ for a given matrix $\mathbf{B}$.

Th. 3 If a vertex v_i of the simplex Θ is not the optimal solution, then there is at least a neighbor vertex v_j (connected with v_i by an edge) for which the objective function has a lower value.

Th. 4 If $\mathbf{d}$ is an extreme point and the criterion $\mathbf{d}_N^T - \mathbf{d}_B^T\mathbf{B}^{-1}\mathbf{N} \geq 0$ (denoted the *optimality criterion*) holds, then $\mathbf{d}$ is the optimal solution.

An overview of the SIMPLEX algorithm

The simplex algorithm idea: iterate between the extreme points (vertices) of the simplex Θ, each such point being determined by a particular squared sub-matrix of the constraints matrix, until one of them meets the optimality criterion.

Step 1 Select an extreme point $\hat{\mathbf{d}}_0$ of the simplex (using Th. 2); $k = 0$

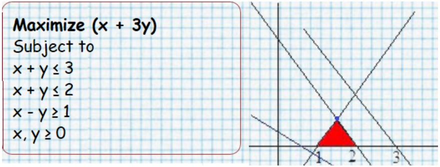
Step 2 Check the optimality criterion for $\hat{\mathbf{d}}_k$ (Th. 4).
If holds, THEN return $\mathbf{d}^* = \hat{\mathbf{d}}_k$

Step 3 Check criteria for the unbound/infeasible case.
If hold, STOP

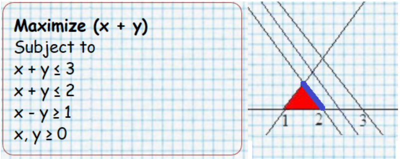
Step 4 Select another extreme point $\hat{\mathbf{d}}_{new}$ (Th. 1), neighbor of $\hat{\mathbf{d}}_k$ and having a lower value for $L()$ (Th. 3); Set $k = k + 1$ and $\hat{\mathbf{d}}_k = \hat{\mathbf{d}}_{new}$. Goto Step 2.

Possible cases

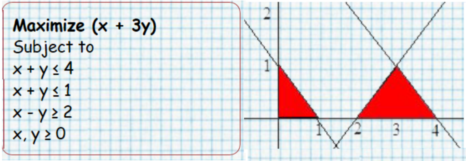
- Finite optimal solution



- Infinite optimal solutions



- No solution



Simplex algorithm

Step 0 (Initialization) Transform inequalities constraints in equalities constraints by introducing slack variables $d_{n+j}, j = 1 \dots m$.

$$
\begin{array}{rcl}
a_{11}d_1 + a_{12}d_2 + \cdots + a_{1n}d_n + d_{n+1} & = & b_1 \\
a_{21}d_1 + a_{22}d_2 + \cdots + a_{2n}d_n + d_{n+2} & = & b_2 \\
& \vdots & \\
a_{m1}d_1 + a_{m2}d_2 + \cdots + a_{mn}d_n + d_{n+m} & = & b_m
\end{array}
$$

Consider the index vector $\sigma(i) = i, i = 1...n+m$. Denote $d_1, ..., d_n$ the decision variables and $d_{n+1}, ..., d_{n+m}$ the basis variables.

Fill the simplex table

	d_1	$\dots$	d_s	$\dots$	d_n		
d_{n+1}	a_{11}	$\dots$	a_{1s}	$\dots$	a_{1n}	b_1	b_1/a_{1s}
$\vdots$	$\vdots$		$\vdots$		$\vdots$	$\vdots$	$\vdots$
d_{n+r}	a_{r1}	$\dots$	a_{rs}	$\dots$	a_{rn}	b_r	b_r/a_{rs}
$\vdots$	$\vdots$		$\vdots$		$\vdots$	$\vdots$	$\vdots$
d_{n+m}	a_{m1}	$\dots$	a_{ms}	$\dots$	a_{mn}	b_m	b_r/a_{ms}
	c_1	$\dots$	c_s	$\dots$	c_n	ϑ	

Note: $\hat{\mathbf{d}}_0$ *is defined by* $\{d_1 = 0, ..., d_n = 0, d_{n+1} = b_1, ..., d_{n+m} = b_m\}$

Simplex algorithm (cont.)

Step 1 If $c_j \geq 0, \forall j = 1..n$ then STOP. $L(\mathbf{d}^*) = \vartheta$, where

$$\mathbf{d}^*_{\sigma(i)} = \begin{cases} 0 & \text{if } 1 \leq i \leq n \\ b_{i-n} & \text{if } n+1 \leq i \leq n+m \end{cases}$$

Step 2 Select the *pivot column* s such that $c_s = min\{c_j | c_j < 0, 1 \leq j \leq n\}$

Step 3 If $a_{is} \leq 0, \forall i = 1..m$ then STOP (no finite solution)

Step 4 Select the *pivot row* r such that $\frac{b_r}{a_{rs}} = min\left\{\frac{b_i}{a_{is}} | a_{is} > 0, 1 \leq i \leq m\right\}$

Note: Two extreme points $\hat{\mathbf{d}}_i$ and $\hat{\mathbf{d}}_j$ are neighbors if their sets of basic variables differs only in one coordinates

Step 5 Refresh the simplex tableau:

$$(a_{ij})_{new} = \begin{cases} a_{ij} - \frac{a_{is}a_{rj}}{a_{rs}}, & i \neq r, j \neq s \\ \frac{1}{a_{rs}}, & i = r, j = s \\ -\frac{a_{is}}{a_{rs}}, & i \neq r, j = s \\ \frac{a_{rj}}{a_{rs}}, & i = r, j \neq s \end{cases} \qquad (b_i)_{new} = \begin{cases} b_i - \frac{b_r a_{is}}{a_{rs}}, & i \neq r \\ \frac{b_r}{a_{rs}}, & i = r \end{cases}$$

$$(c_j)_{new} = \begin{cases} c_j - \frac{c_s a_{rj}}{a_{rs}}, & j \neq s \\ -\frac{c_s}{a_{rs}}, & j = s \end{cases} \qquad \vartheta_{new} = \vartheta + \frac{c_s b_r}{a_{rs}}$$

Simplex algorithm (cont.)

Step 6 Interchange $\sigma(s)$ with $\sigma(n+r)$ (the decision variable $d_{\sigma(s)}$ is exchanged with the basis variable $d_{\sigma(n+r)}$);
Goto Step 1

For a maximization problem, the changes are:

- The STOP condition becomes $c_j \leq 0, \forall j$
- The pivot column s is given by $c_s = max\{c_j | c_j > 0\}$

Remark 1: For particular instances the algorithm may infinitely cycle, but several methods (as Blands' rule) were proposed to eliminate this behavior.

Remark 2: The worst-cas complexity of Simplex algorithm is exponential time, but in practice the most cases can be solved in a polynomial time.

Example

Min $L(x_1, x_2) = -x_1 - 2x_2$,

$$\begin{cases} x_1 & \leq 4 \\ x_1 + x_2 & \leq 6 \\ -x_1 + x_2 & \leq 1 \end{cases}$$

- Initial Simplex table and σ vector of index

	d_1	d_2	**b**	b_i/a_{is}
d_3	1	0	4	
d_4	1	1	6	
d_5	-1	1	1	
	-1	-2	0	

$\sigma = (1, 2, 3, 4, 5)$

- Step 1: $\exists j = 1$ such that $c_j = -1 < 0$
- Step 2: $s = 2$;
- Step 3: $\exists i = 2$ such that $a_{is} = 1 > 0$
- Step 4: $r = 3$

	d_1	d_2	**b**	b_i/a_{is}
d_3	1	0	4	∞
d_4	1	1	6	6
d_5	-1	$\boxed{1}$	1	1
	-1	-2	0	

Example (cont.)

- Step 5 and 6: New simplex table and σ vector (σ_2 and σ_5 were exchanged)

	d_1	d_5	**b**	b_i/a_{is}
d_3	1	0	4	
d_4	2	-1	5	
d_2	-1	1	1	
	-3	2	-2	

$\sigma = (1, 5, 3, 4, 2)$

- Step 1: $\exists j = 1$ such that $c_1 = -3 < 0$
- Step 2: $s = 1$
- Step 3: $\exists i = 1$ such that $a_{is} = 1 > 0$
- Step 4: $r = 2$

	d_1	d_5	**b**	b_i/a_{is}
d_3	1	0	4	4
d_4	$\boxed{2}$	-1	5	2.5
d_2	-1	1	1	∞
	-3	2	-2	

- Step 5 and 6: New simplex table and σ vector (σ_1 and σ_4 were exchanged):

	d_4	d_5	
d_3	-0.5	0.5	1.5
d_1	0.5	-0.5	2.5
d_2	0.5	0.5	3.5
	1.5	0.5	-9.5

$\sigma = (4, 5, 3, 1, 2)$

- Step 1: $\forall j = 1..2, c_j > 0$; STOP; The optimal solution: $d_{\sigma_1} = d_4 = 0$, $d_{\sigma_2} = d_5 = 0$, $d_{\sigma_3} = d_3 = 1.5$, $d_{\sigma_4} = d_1 = 2.5$, $d_{\sigma_5} = d_2 = 3.5$

Integer optimization

- Constraint: all or some of $\mathbf{d}$ coordinates must take integer values (or values from a finite, discrete set).
- Special case: integer linear optimization problem
- Heuristic approach: solve the relaxed linear problem (without the constraint on integer); for all $i = 1..n$, if d_i^* is integer: ok; if not, take the rounded d_i^*. No guarantee to find the optimal solution !
- The Gomory algorithm

Step 1 Solve the relaxed problem (with Simplex algorithm).

Step 2 If some coordinates of $\mathbf{d}^*$ are not integer, then add a new constraint such that

1. The solution $\mathbf{d}^*$ becomes non feasible
2. Each feasible integer solution of the integer linear problem fulfills the new constraint

Step 3 Consider the enlarged problem ($m \leftarrow m + 1$) and return to Step 1

Gomory cup

- Consider a feasible solution $\mathbf{d} = [d_1, ..., d_n]$ (d_i integer, $\forall i = 1..n$) and the constraint $a_1 d_1 + ... + a_n d_n = b$, with $a_i, b \in R$
- If $\lfloor x \rfloor$ is the smaller integer $\leq x$, then we can express a_i as $\lfloor a_i \rfloor + (a_i - \lfloor a_i \rfloor) = \lfloor a_i \rfloor + f_i$, and b as $\lfloor b \rfloor + (b - \lfloor b \rfloor) = \lfloor b \rfloor + f$.
- Therefore,

$$\sum_{i=1}^{n} a_i d_i = b \Leftrightarrow \sum_{i=1}^{n} (\lfloor a_i \rfloor + f_i) d_i = \lfloor b \rfloor + f \Leftrightarrow \sum_{i=1}^{n} f_i d_i - f = \lfloor b \rfloor - \sum_{i=1}^{n} \lfloor a_i \rfloor d_i$$

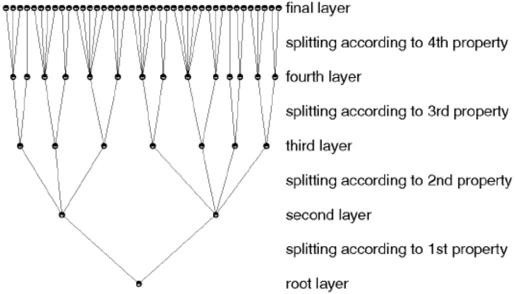
- In the last equality, the right-hand side is integer; consequently, $\sum_{i=1}^{n} f_i d_i - f$ is also integer
- Because $\sum_{i=1}^{n} f_i d_i \geq 0$ and $0 \leq f < 1$, necessary $\sum_{i=1}^{n} f_i d_i - f$ is positive, e.g.

$$f_1 d_1 + ... + f_n d_n \geq f$$

  for every feasible solution.

Branch & Bound

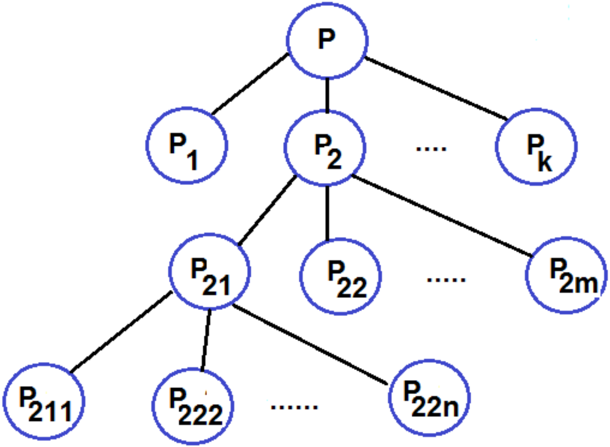
- Combinatorial optimization problems
- If not expressed as linear problems, simplex algorithm can't be applied
- The only way: searching through the set of feasible solutions
- To avoid a complete search - create an appropriate search tree



Branch & Bound (cont.)

- **Branching** step:
  - All the leaves of one subbranch of a given node have one/more properties in common, different from all configurations of all others subbranches starting from the same node
  - Splitting procedure for all nodes at level a: based on the same property
    - Ex: if d_i takes values in $\{v_1, v_2, v_3\}$, then the left subbranch will contain all feasible solutions having $d_i = v_1$, the middle subbranch those having $d_i = v_2$ and the last subbranch the solutions having $d_i = v_3$
  - Each node is equivalent with a particular optimization problem. The branching step split this problem in several sub-problems.

Search Problem Tree



Branch & Bound (cont.)

- **Bounding** step:
  - Evaluation procedure: a node contains sufficient information to allow the estimation, by only examining the node, of a local lower bound for the loss function values of all leaves of outgoing subtree
  - A global upper bound for the optimum value may be given by
    1. a good solution already known
    2. an approximate solution returned by an heuristic, or
    3. the loss function value for the first node
  - During the search process, better solutions may be found; the upper bound is consequently adjusted (decreased)
- **Pruning** step:
  - If the global upper bound is smaller than the local lower bound at a specific branching node, then all the outgoing subbranches can be "cut off"
    - its impossible for any configuration in one of the subbranches going out to be the optimum solution.

General algorithm

Init List of active sub-problems (nodes) $L_P = \{\mathbf{P}\}$;

Set the upper bound for the optimal solution $UB(\mathbf{P})$ as ∞ or as $L(\mathbf{d})$ for a particular $\mathbf{d}$ (so $L(\mathbf{d}^*) \leq UB(\mathbf{P})$)

Step 1 WHILE $L_P \neq \emptyset$:

- Select an active sub-problem $\mathbf{P}_i$ from L_P
- Calculate the lower bound $LB(\mathbf{P}_i)$ satisfying $L(\mathbf{d})_{\mathbf{d} \in \mathbf{P}_i} \geq LB(\mathbf{P}_i)$
- If $LB(\mathbf{P}_i) \geq UB(\mathbf{P})$, delete $\mathbf{P}_i$ from L_P
- If not, either solve $\mathbf{P}_i$ and update $U(\mathbf{P})$ if necessary, or split $\mathbf{P}_i$ in sub-problems and add them to L_P

Example: Linear Integer Optimization

P: Min $L(x_1, x_2) = x_1 - 2x_2$, u.c.

$$\begin{cases} -4x_1 + 6x_2 & \leq 9 \\ x_1 + x_2 & \leq 4 \\ x_1, x_2 & \geq 0 \\ x_1, x_2 & \text{integers} \end{cases}$$

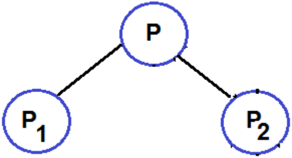
- The lower bound $LB(\mathbf{P_i})$ is $L(x^*)$, where x^* is the optimal solution of the relaxed linear problem
- If $x^* = (x_1^*, x_2^*)$ has real coordinates, take one of these $(x_i^*, i \in \{1, 2\})$ and split the linear problem in two sub-problems:
  - $\mathbf{P}_1$: the initial problem **P** with a supplementary constraint $x_i \geq \lceil x_i^* \rceil$
  - $\mathbf{P}_2$: the initial problem **P** with a supplementary constraint $x_i \leq \lfloor x_i^* \rfloor$
  - *Remark*: x^* is not a feasible solution for none of two sub-problems !

Example (cont.)

Init $L_P = \{\mathbf{P}\}$; $UB = \infty$

Step1 $x^* = (1.5, 2.5)$, $LB(\mathbf{P}) = -3.5 < UB$

Step2 $\mathbf{P}_1$ = $\mathbf{P}$ with new constraint $x_2 \geq \lceil x_2^* \rceil = 3$; $\mathbf{P}_2$ = $\mathbf{P}$ with new constraint $x_2 \leq \lfloor x_2^* \rfloor = 2$; $L_P = \{\mathbf{P}_1, \mathbf{P}_2\}$

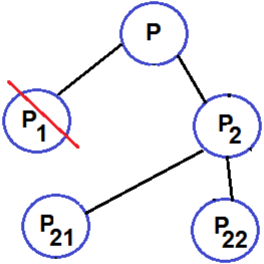


Step3 $\mathbf{P}_1$ has no feasible solution solution; $L_P = \{\mathbf{P}_2\}$

Example (cont.)

Step4 x^* for $\mathbf{P}_2$ is $(0.75, 2)$, and $LB(\mathbf{P_2}) = -3.25 < UB = \infty$

Step5 $\mathbf{P}_{21}$ = $\mathbf{P}_2$ with new constraint $x_1 \geq \lceil x_1^* \rceil = 1$; $\mathbf{P}_{22}$ = $\mathbf{P}_2$ with new constraint $x_1 \leq \lfloor x_1^* \rfloor = 0$; $L_P = \{\mathbf{P}_{21}, \mathbf{P}_{22}\}$



Example (cont.)

Step6 x^* for $\mathbf{P}_{21}$ is $(1,2)$; $LB(\mathbf{P_{21}}) = -3$ and $UB = -3$ (we have a sol. with integer coordinates); $L_P = \{\mathbf{P}_{22}\}$

Step7 x^* for $\mathbf{P}_{22}$ is $(0, 1.5)$; $LB(\mathbf{P_{22}}) = -3 \geq UB = -3$; $\mathbf{P}_{22}$ is deleted.

Step7 $L_P = \{\}$, the optimal solution for $\mathbf{P}$ is $x^* = (1,2)$