最適化アルゴリズムの基礎

2022/04/20

スタートアップゼミ #4

D1 黛 風雅 / Mayuzumi Fuga

今日の内容

- 最適化問題とは
- 離散最適化と連続最適化
- 最適化問題の主要解法
- ネットワーク最適化問題：Dijkstra法
- 動的計画法
- 線形計画問題：単体法
- 双対問題

最適化問題とは

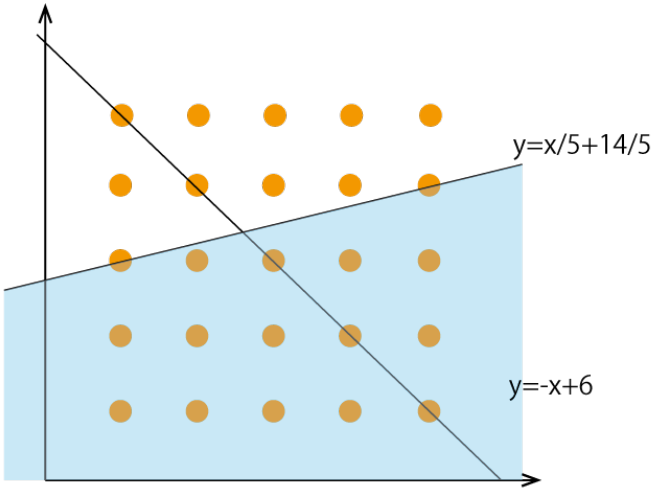
Minimize $f(x)$ 目的関数(最大化の場合は$-f(x)$)

s.t. $x \in S$ 制約条件

such thatまたは
subject to

$g_i(x) \leq b, i = 0, \ldots, m$ 不等式制約

$h_j(x) = c, i = 0, \ldots, r$ 等式制約



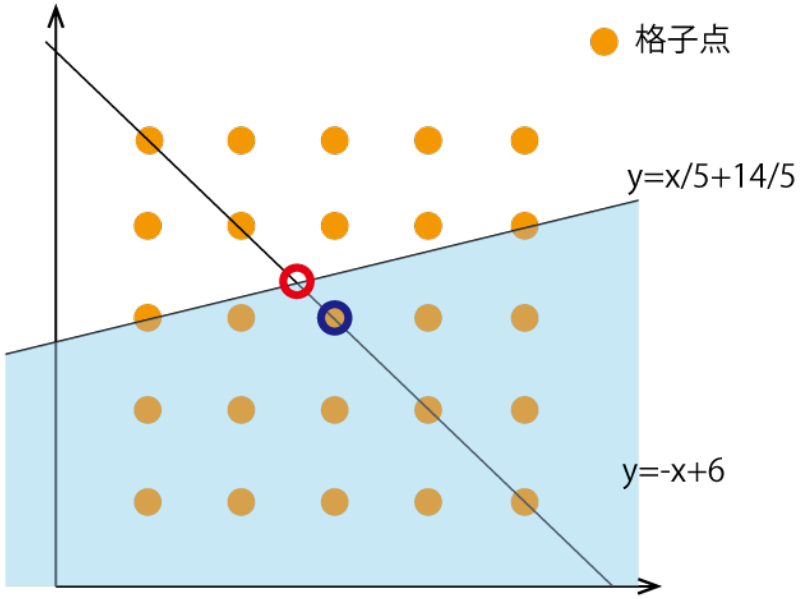
例

Max $f(x) = -x + 6$

s.t. $(g(x) =) \frac{1}{5}x + \frac{14}{5} \leq 0$

S … 不等式制約によるxの実行可能領域

離散最適化と連続最適化



Max $\quad f(x) = -x + 6$

s.t. $\quad (g(x) =) \frac{1}{5}x + \frac{14}{5} \leq 0$

最適化問題の解

- ○ … xが連続変数$(x \in \mathbb{R})$のとき
- ○ … xが離散変数$(x \in \mathbb{Z})$のとき

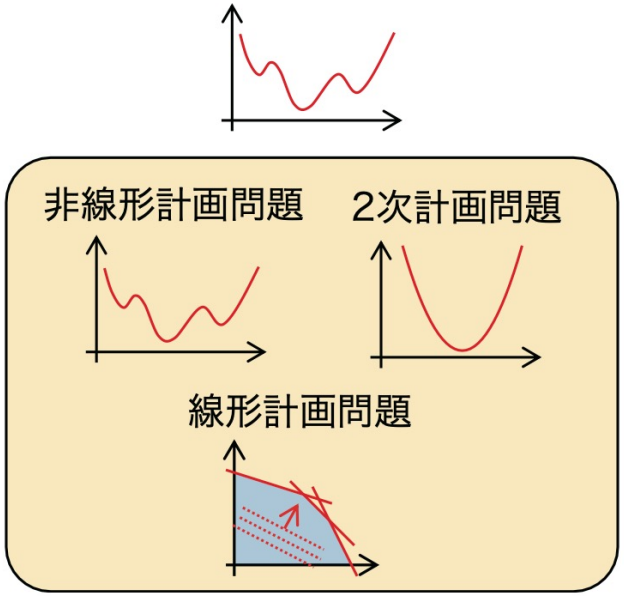
最適化問題は，変数が連続か離散かによって区別される

- ○ を求めるだけであれば，連立方程式を解くだけでよいが
- ○ は連立方程式では求解できないので問題としてより難しい

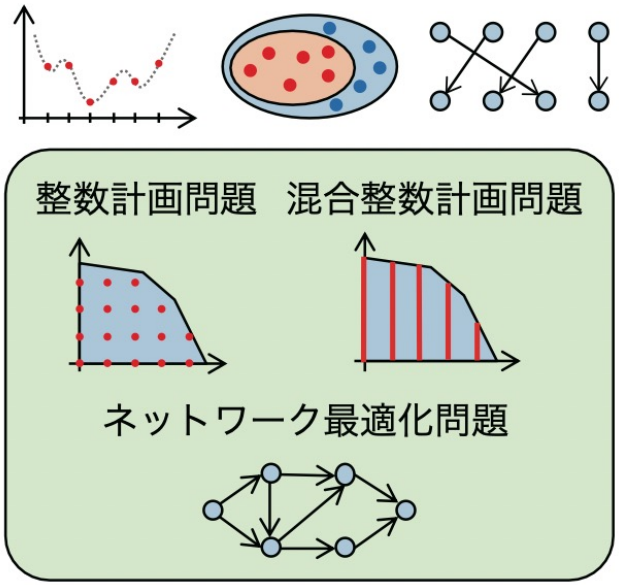
離散最適化と連続最適化

最適化問題は，変数が連続か離散かによって区別される

連続最適化



離散最適化
（組合せ最適化）



梅谷(2021)より

最適化問題の主要な解法

連続最適化

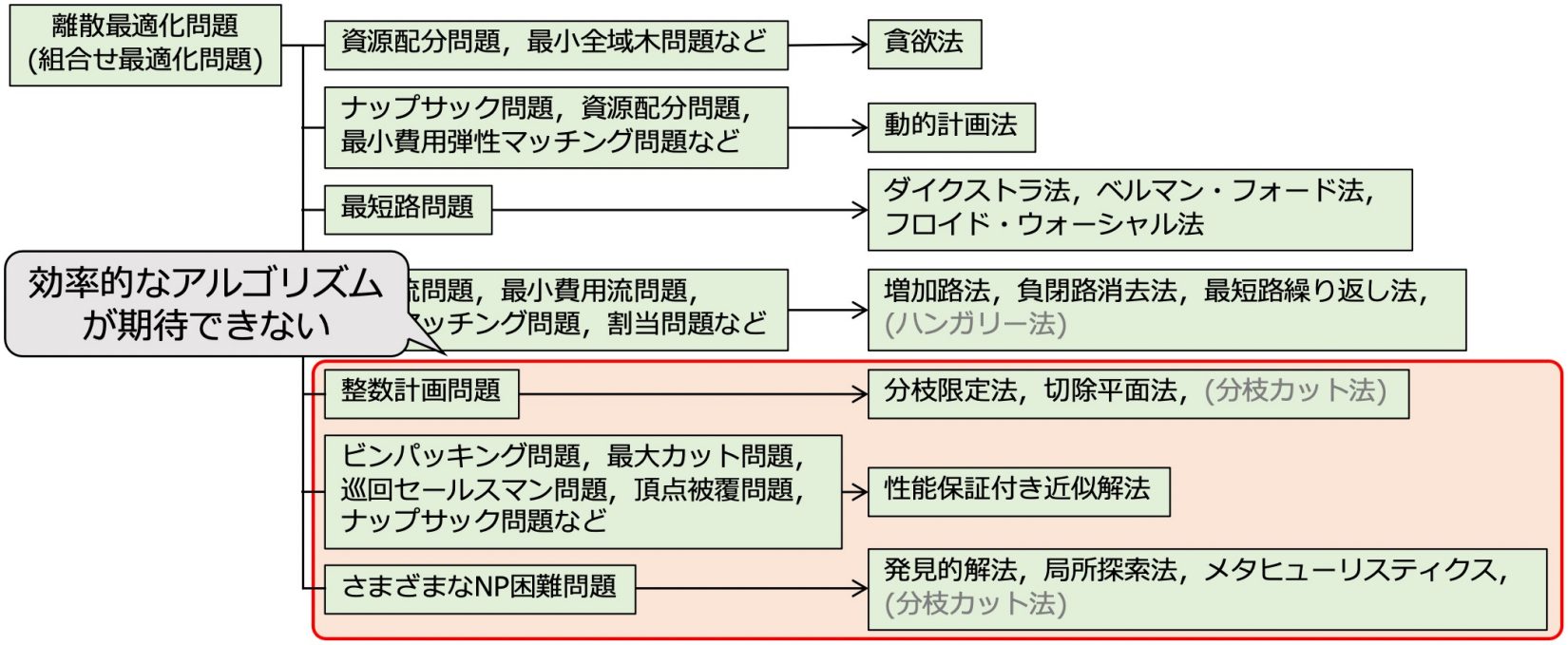
- 線形計画問題(LP)
  ーシンプレックス法(単体法)
  ー内点法
- 非線形計画問題(NLP)
  ーラグランジュ乗数法

離散最適化
(組合せ最適化)

- ネットワーク最適化問題
  ーDijkstra法：最短路問題
- 整数計画問題(IP)
  ー分枝限定法
  ー切除平面法(Gomory Cut)
- 混合整数計画問題(MIP)
  ー分枝限定法

引用　分枝限定法と切除平面法

離散最適化問題の小類型と解法



- 問題の性質によって，効率的なアルゴリズムが開発可能なものもあるが，効率的に解けない問題は多い
- 今回のゼミでは前者のアルゴリズムを扱います

ネットワーク最適化問題

グラフ構造における線形の不等式制約のもとで線形の目的関数を最大化（最小化）させる整数値の未知変数を決定する問題

- **最短路問題(shortest path problem)**

  均衡配分アルゴリズムでも登場

- 最小木問題(minimum spanning tree problem)
- 最大流問題(maximum flow problem)
- 最小費用流問題(minimum cost flow problem)

<- 多項式オーダーの解法が開発されている

- 多品種流問題 <- 多項式オーダーの解法がない

最短路問題の定式化

- ネットワーク：頂点集合V，辺集合Eからなる有向グラフ$G=(V,E)$
- リンク (i,j) 間の費用(ここでは距離)： $c_{i,j}$
- 決定変数… リンク (i,j) が最短路に含まれるか否か： $x_{i,j}$
- 始点 s，終点 t 間の最短路を求める

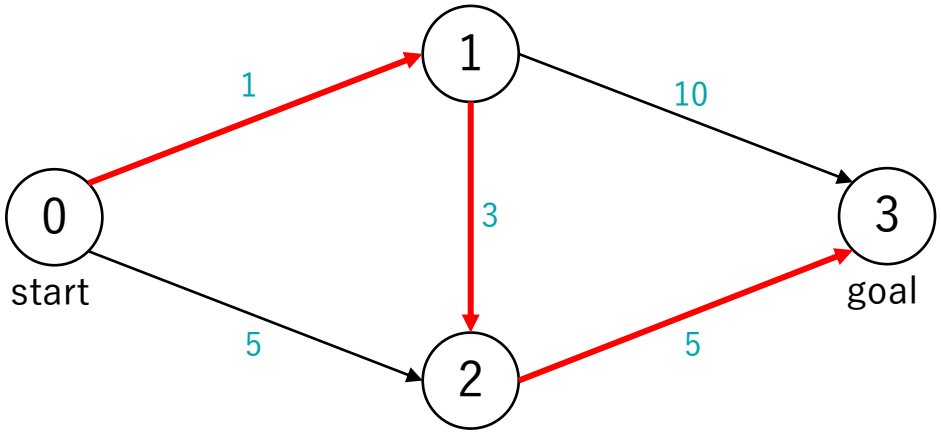
$$\min \quad \sum_{(i,j)\in E} c_{i,j}x_{i,j}$$

$$\text{s.t.} \quad \sum_{(s,j)\in E} x_{s,j} - \sum_{(j,s)\in E} x_{j,s} = 1$$

$$\sum_{(t,j)\in E} x_{t,j} - \sum_{(j,t)\in E} x_{j,t} = -1$$

$$\sum_{(i,j)\in E} x_{i,j} - \sum_{(j,i)\in E} x_{j,i} = 0, \forall i \in V \setminus \{s,t\}$$

$$x_{i,j} \in \{0,1\}, \forall (i,j) \in E$$



$s=0, t=3$ のとき

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

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

最短経路問題

１始点最短路問題

- Dijkstra法
- A*法
- Bellman-ford法(ラベル修正法)
  - コストが負の値の辺が含まれていても計算可能

cf. 全対間最短路問題(多品種流への応用)

多品種流問題では，多数の始点・終点間の最短路の計算が求められる．全始点に１始点最短路のアルゴリズムを適用するよりも効率的な解法が開発されている．

- Floyd-Warshall 法
- Murchland法

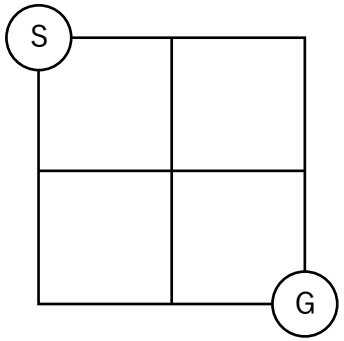
最短経路探索アルゴリズムの必要性

全列挙して最短路を調べることも可能だが，，

ネットワークサイズが大きくなると組合せ爆発で計算が困難に．

1×1	2通り
2×2	12通り
3×3	184通り
4×4	8,512通り
5×5	1,262,816通り
6×6	575,780,564通り
7×7	789,360,053,252通り
8×8	3,266,598,486,981,642通り
9×9	41,044,208,702,632,496,804通り
10×10	1,568,758,030,464,750,013,214,100通り
11×11	182,413,291,514,248,049,241,470,885,236通り

⋮



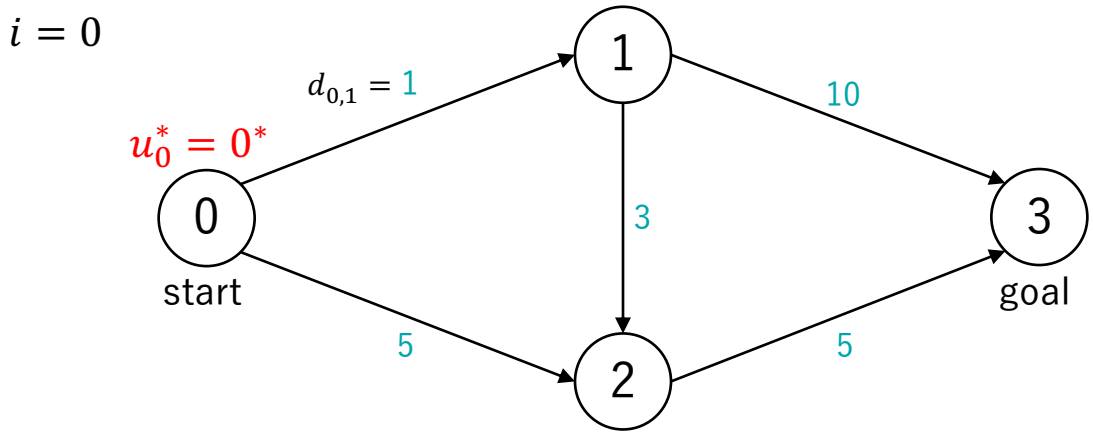
『フカシギの数え方』おねえさんといっしょ！みんなで数えてみよう！
https://youtu.be/Q4gTV4r0zRs

Dijkstra法

1. 始点s以外の全てのノードを終点とし，
   リンク $l_{ij} = (i,j)$の長さをd_{ij}とする．
2. 始点ノードに永久ラベル$u_0^* = 0$を与える．$i = 0$とする．
3. ノードiを始点に持つ全てのリンクl_{ij}に対して，
   終点ノードjが永久ラベルを持っていないとき，
   i. ノードjが一時ラベルを持っていない場合，
      一時ラベル$u_j = u_i^* + d_{ij}$を与える．
   ii. ノードjが一時ラベルu_jを持っており，その一時ラベルu_jが$u_i^* + d_{ij}$よりも大きい場合，この一時ラベルを$u_i^* + d_{ij}$の値に改訂する．
4. 一時ラベル全体の中で最小の値を持つもの1つ選び，iをそのノード番号とする．また，一時ラベルu_iを永久ラベルu_i^*に変える．
5. すべての集中ノードに永久ラベルがつくまで，3・4を繰り返す．

Dijkstra法

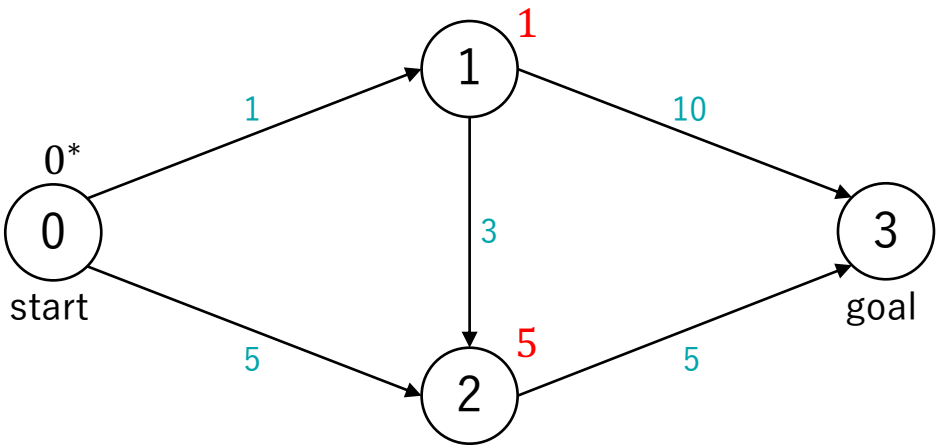
1. 始点s以外の全てのノードを終点とし，
   リンク $l_{ij} = (i, j)$の長さをd_{ij}とする．
2. 始点ノードに永久ラベル$u_0^* = 0$を与える．$i = 0$とする．



Dijkstra法

3. ノードiを始点に持つ全てのリンクl_{ij}に対して，終点ノードjが永久ラベルを持っていないとき，
   i. ノードjが一時ラベルを持っていない場合，(条件1)
      一時ラベル$u_j = u_i^* + d_{ij}$を与える．
   ii. ノードjが一時ラベルu_jを持っており，(条件2-1)
      その一時ラベルu_jが$u_i^* + d_{ij}$よりも大きい場合，(条件2-2)
      この一時ラベルu_jを$u_i^* + d_{ij}$の値に改訂する．
4. 一時ラベル全体の中で最小の値を持つもの1つ選び，iをそのノード番号とする．また，一時ラベルu_iを永久ラベルu_i^*に変える．

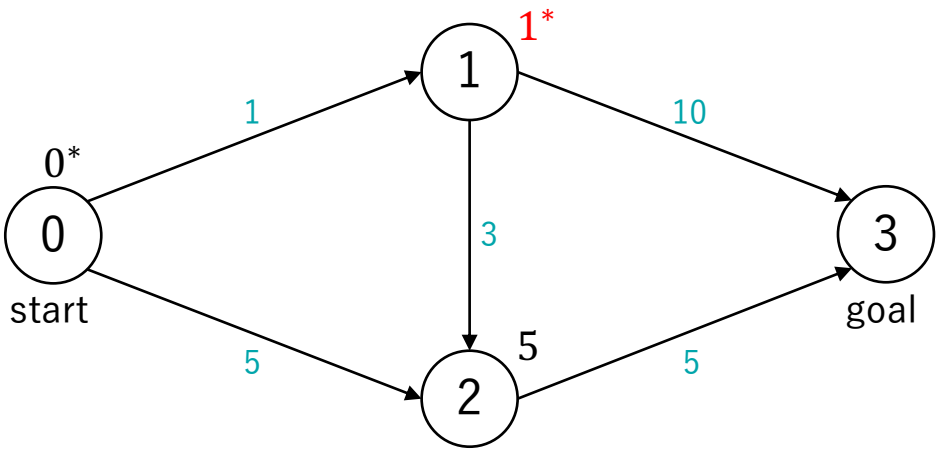
$i = 0$



Dijkstra法

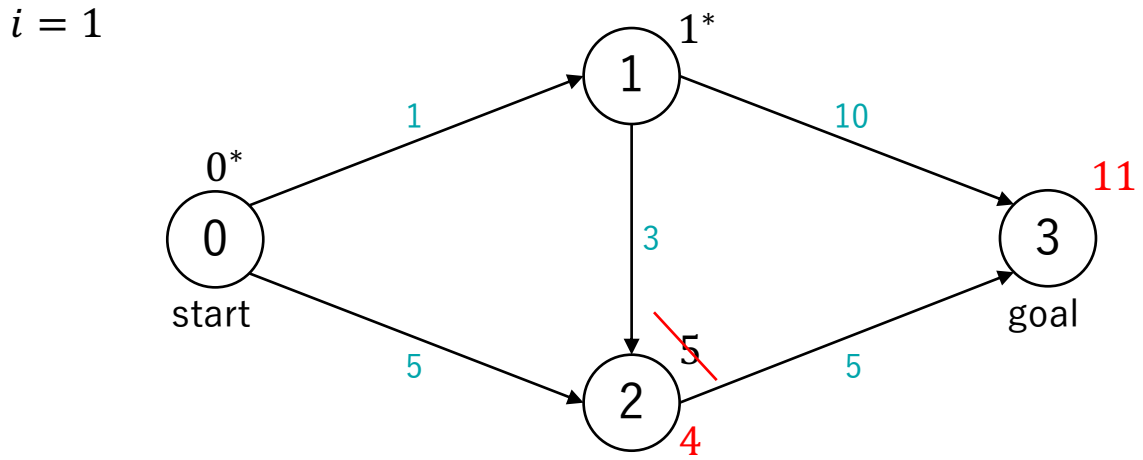
3. ノードiを始点に持つ全てのリンクl_{ij}に対して，終点ノードjが永久ラベルを持っていないとき，
   i. ノードjが一時ラベルを持っていない場合，(条件1)
      一時ラベル$u_j = u_i^* + d_{ij}$を与える．
   ii. ノードjが一時ラベルu_jを持っており，(条件2-1)
      その一時ラベルu_jが$u_i^* + d_{ij}$よりも大きい場合，(条件2-2)
      この一時ラベルu_jを$u_i^* + d_{ij}$の値に改訂する．
4. 一時ラベル全体の中で最小の値を持つもの1つ選び，iをそのノード番号とする．また，一時ラベルu_iを永久ラベルu_i^*に変える．

$i = 1$



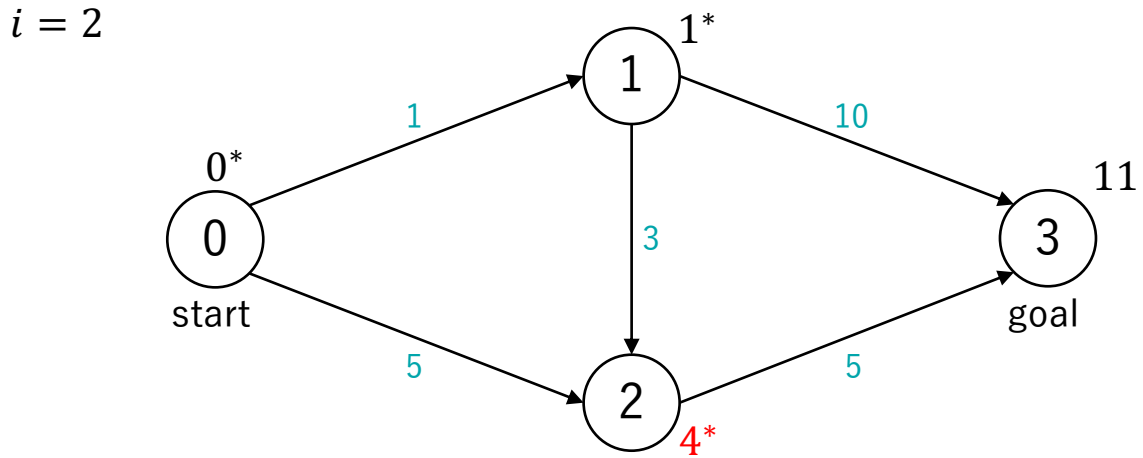
Dijkstra法

3. ノードiを始点に持つ全てのリンクl_{ij}に対して，終点ノードjが永久ラベルを持っていないとき，
   i. ノードjが一時ラベルを持っていない場合，(条件1)
      一時ラベル$u_j = u_i^* + d_{ij}$を与える．
   ii. ノードjが一時ラベルu_jを持っており，(条件2-1)
      その一時ラベルu_jが$u_i^* + d_{ij}$よりも大きい場合，(条件2-2)
      この一時ラベルu_jを$u_i^* + d_{ij}$の値に改訂する．
4. 一時ラベル全体の中で最小の値を持つもの1つ選び，iをそのノード番号とする．また，一時ラベルu_iを永久ラベルu_i^*に変える．



Dijkstra法

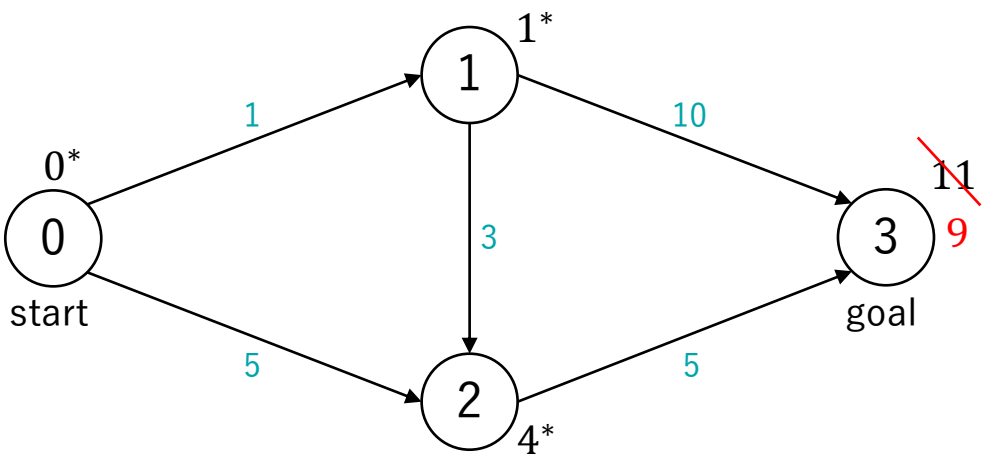
3. ノードiを始点に持つ全てのリンクl_{ij}に対して，終点ノードjが永久ラベルを持っていないとき，
   i. ノードjが一時ラベルを持っていない場合，(条件1)
      一時ラベル$u_j = u_i^* + d_{ij}$を与える．
   ii. ノードjが一時ラベルu_jを持っており，(条件2-1)
      その一時ラベルu_jが$u_i^* + d_{ij}$よりも大きい場合，(条件2-2)
      この一時ラベルu_jを$u_i^* + d_{ij}$の値に改訂する．
4. 一時ラベル全体の中で最小の値を持つもの1つ選び，iをそのノード番号とする．また，一時ラベルu_iを永久ラベルu_i^*に変える．



Dijkstra法

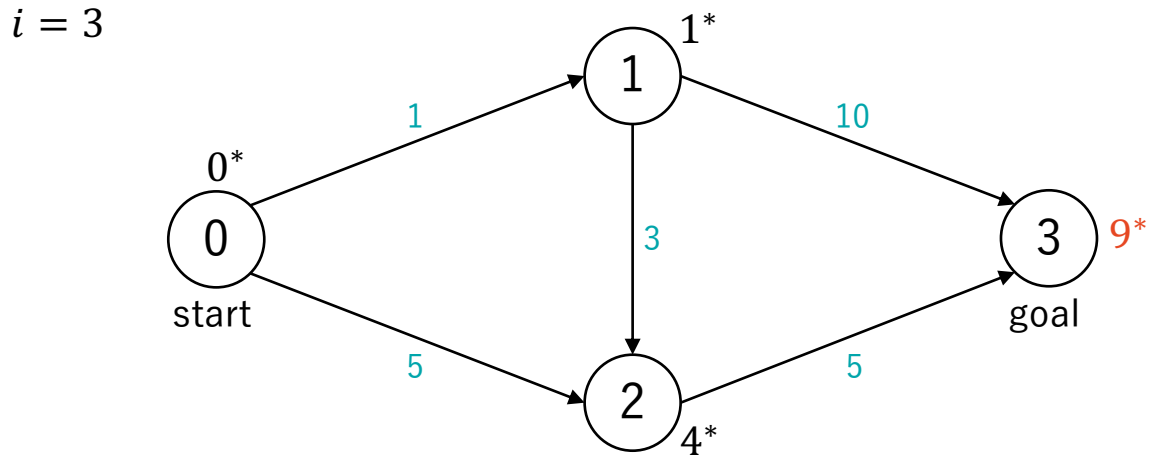
3. ノードiを始点に持つ全てのリンクl_{ij}に対して，終点ノードjが永久ラベルを持っていないとき，
   i. ノードjが一時ラベルを持っていない場合，(条件1)
      一時ラベル$u_j = u_i^* + d_{ij}$を与える．
   ii. ノードjが一時ラベルu_jを持っており，(条件2-1)
      その一時ラベルu_jが$u_i^* + d_{ij}$よりも大きい場合，(条件2-2)
      この一時ラベルu_jを$u_i^* + d_{ij}$の値に改訂する．
4. 一時ラベル全体の中で最小の値を持つもの1つ選び，iをそのノード番号とする．また，一時ラベルu_iを永久ラベルu_i^*に変える．

$i = 2$



Dijkstra法

3. ノードiを始点に持つ全てのリンクl_{ij}に対して，終点ノードjが永久ラベルを持っていないとき，
   i. ノードjが一時ラベルを持っていない場合，(条件1)
      一時ラベル$u_j = u_i^* + d_{ij}$を与える．
   ii. ノードjが一時ラベルu_jを持っており，(条件2-1)
      その一時ラベルu_jが$u_i^* + d_{ij}$よりも大きい場合，(条件2-2)
      この一時ラベルu_jを$u_i^* + d_{ij}$の値に改訂する．
4. 一時ラベル全体の中で最小の値を持つもの1つ選び，iをそのノード番号とする．また，一時ラベルu_iを永久ラベルu_i^*に変える．



A*法

- Dijkstra法の改良アルゴリズム
- 終点までの距離の推定値（ヒューリスティック関数）を利用
- 基本的にダイクストラ法よりも高速だが，ダイクストラ法と異なり，一回に計算できるのは起終点1ペアに対する最短経路のみ
- あるノードnを通る最短経路のコスト$f^*(n)$を考える際に，
  スタートからのコスト$g^*(n)$と，ゴールまでのコスト$h^*(n)$の和を考える．

$$f^*(n) = g^*(n) + h^*(n)$$

探索の過程で近づけていくことができる
→動的計画法的＝Dijkstra法

ゴールに辿り着くまで分からない

人間が設計した推定値$h(n)$を与える！
ヒューリスティック関数

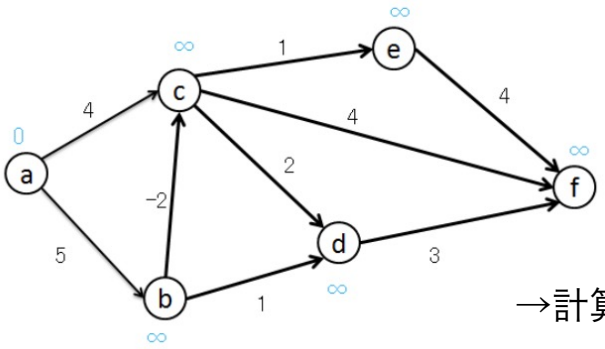
- ヒューリスティック関数には，
  マンハッタン距離やユークリッド距離がよく用いられる．
  - ✓ マンハッタン距離$h(n) = x(n) + y(n)$
  - ✓ ユークリッド距離$h(n) = \sqrt{\{x(n)\}^2 + \{y(n)\}^2}$

https://nw.tsuda.ac.jp/lec/BellmanFord/

Bellman-Ford法

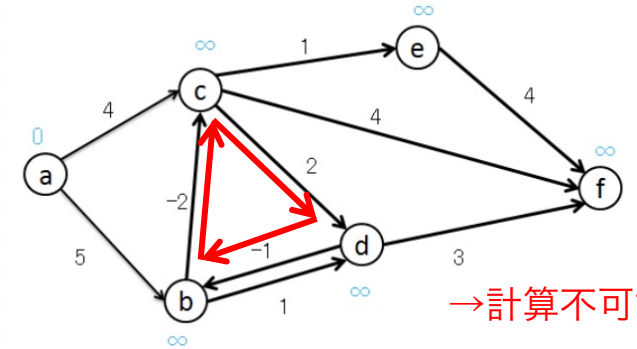
- 各頂点に与えられたコストを，より小さい値で置き換えていくアルゴリズム.
- Dijkstra法・A*アルゴリズムよりも計算量が多い.
- コストが負の値の辺が含まれていても計算可能（Dijkstra法，A*アルゴリズムでは不可能）.
- ただし，グラフに負閉路 (negative cycle、辺の重みの総和が負になる閉路) が含まれるとき、 その閉路を通過することによりいくらでも重みを小さくできるので最短経路は決まらない.

負の値の辺あり&負閉路なし



→計算可能

負の値の辺あり&負閉路あり



→計算不可能

動的計画法(Dynamic Programming)

- 動的計画法(DP)では，，，

「それまでに得られた小さな部分問題の最適解を利用して，より大きな部分問題の最適解を求める手続きをボトムアップに積み上げ，同じ部分問題を繰り返し解くことなく効率的に最適解を求めることができる」

適用例)
- 最短路問題
- ナップサック問題
- 資源配分問題

動的計画法(Dynamic Programming)

- 最適化問題を部分構造最適性を満足する部分問題に分解可能であれば，部分問題の最適解を統合して元の問題の**最適解を求解できる**

… 部分構造最適性(or 最適性原理)

「元の問題の最適解が，ある部分問題の最適解を含む」

Dijkstra法＝動的計画法的アルゴリズム(DP)

最短路問題に対するアルゴリズムは，
有向グラフ $G = (V, E)$ の以下の性質に基づいて設計されている．

始点sから各頂点$v \in V$へのある路の長さをf_vとする．このとき全ての頂点vについてf_vが最短路の長さであるための必要十分条件は，

$$f_v \leq f_u + d_e, \qquad e = (u, v) \in E \qquad \text{(三角不等式)}$$

が成り立つこと

▶ **部分構造最適性** を満たす

線形計画問題(LP)

- 線形関数を目的関数とする最適化問題
- 変数を整数のみに限定した場合，これを整数計画問題(IP)と呼ぶ
- 特に変数が0,1のみの場合，0-1整数計画問題(BIP)と呼ぶ
- 一部の変数が整数変数で残りの変数が連続変数の場合，
  これを混合整数計画問題(MIP)と呼ぶ

等式標準形

Minimize $\sum_j^n c_j x_j$

s.t. $a_i^{\mathrm{T}} x_j = b_i, i = 0, \dots, m$

$x_j \geq 0, i = 0, \dots, n$

$x_j \in \mathbb{R}, a_i \in \mathbb{R}^n, b_i \in \mathbb{R}, c_j \in \mathbb{R}$

- どのような線形計画問題も，適当な変形によって等式標準形に書き下せる
- 解法が標準形に対して開発されているため，最初に等式標準形に書き直す

等式標準形への変形

等式標準形

$$\text{Minimize} \quad \sum_j^n c_j x_j$$

$$\text{s.t.} \quad a_i^{\mathrm{T}} x_j = b_i, i = 0, \ldots, m$$

$$x_j \geq 0, i = 0, \ldots, n$$

例)

$$\begin{aligned} \text{Max} \quad & 2x_1 + x_2 + 3 \\ \text{s.t.} \quad & x_1 + 3x_2 \leq 3 \\ & x_1 - x_2 \leq 1 \\ & x_1 \geq 0 \end{aligned}$$

$$\begin{aligned} \text{Min} \quad & -2x_1 - x_2 \\ \text{s.t.} \quad & x_1 + 3x_2 + s_1 = 3 \\ & x_1 - x_2 + s_2 = 1 \\ & x_1, s_1, s_2 \geq 0 \end{aligned}$$

操作①スラック変数 $s_1, s_2 \geq 0$ の導入

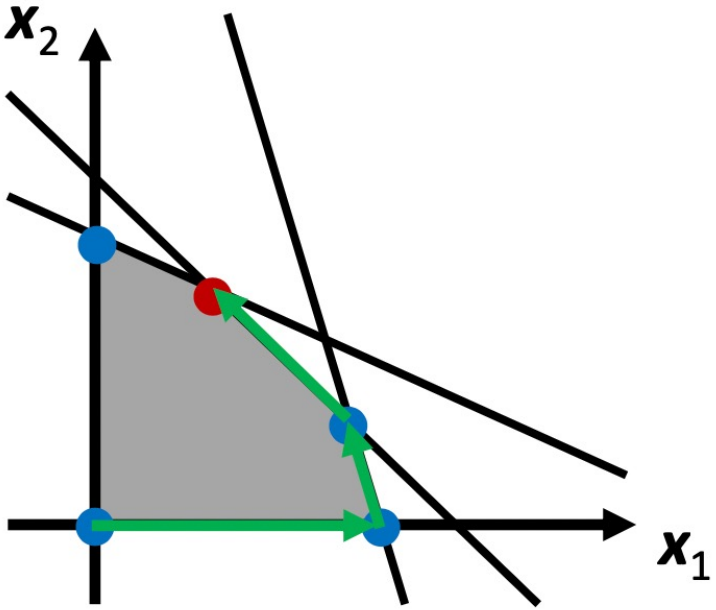
操作②非負変数 $z_1, z_2 \geq 0$ の導入 → $x_2 = z_1 - z_2$

$$\begin{aligned} \text{Min} \quad & -2x_1 - (z_1 - z_2) \\ \text{s.t.} \quad & x_1 + 3(z_1 - z_2) + s_1 = 3 \\ & x_1 - (z_1 - z_2) + s_2 = 1 \\ & x_1, s_1, s_2, z_1, z_2 \geq 0 \end{aligned}$$

引用　単体法を理解しよう！(水野, 2019)

単体法(シンプレックス法)

- 線形計画問題の基礎的な解法 (G. Dantzig, 1947)
- 凸多面体の任意の頂点(実行可能解)から出発, 目的関数値が改善する隣接頂点への移動を繰り返す
- 各頂点でn本の線形式からなる連立方程式を解く



単体法(シンプレックス法)

例題) あるメーカーA が利益最大化を目指して生産計画を立てたい
x_1, x_2を求める最適化問題を定式化して，単体法で解け.

	ビターチョコ x_1	ミルクチョコ x_2	使用可能量
カカオマス y_1	2	1	70(unit/day)
カカオバター y_2	3	4	180(unit/day)
利益	￥6 (/unit)	￥4 (/unit)	

$$
\begin{aligned}
\textbf{Max}\quad & 6x_1 + 4x_2 \\
\textbf{s.t.}\quad & 2x_1 + x_2 \leq 70 \\
& 3x_1 + 4x_2 \leq 180 \\
& x_1, x_2 \geq 0 \\
& x_1, x_2 \in \mathbb{R}
\end{aligned}
$$

等式標準形

$$
\begin{aligned}
\textbf{Min}\quad & -6x_1 - 4x_2 \\
\textbf{s.t.}\quad & 2x_1 + x_2 + s_1 = 70 \\
& 3x_1 + 4x_2 + s_2 = 180 \\
& x_1, x_2, s_1, s_2 \geq 0 \\
& x_1, x_2, s_1, s_2 \in \mathbb{R}
\end{aligned}
$$

単体法(シンプレックス法)

Step 1　制約条件に含まれる等式数と同じ数の変数(=2個)を選ぶ

$$\begin{aligned} \textbf{Min} \quad & -6x_1 - 4x_2 \\ \textbf{s.t.} \quad & 2x_1 + x_2 + s_1 = 70 \\ & 3x_1 + 4x_2 + s_2 = 180 \end{aligned}$$

以後省略

$$x_1, x_2, s_1, s_2 \geq 0$$

$$x_1, x_2, s_1, s_2 \in \mathbb{R}$$

- s_1, s_2 を選ぶとして，これらを**基底変数**と呼ぶ
- x_1, x_2 はこのとき，**非基底変数**と呼ぶ

Step 2　等式標準形を辞書形に変換する

$$\begin{aligned} \textbf{Min} \quad & u = -6x_1 - 4x_2 \\ \textbf{s.t.} \quad & s_1 = 70 - 2x_1 - x_2 \\ & s_2 = 180 - 3x_1 - 4x_2 \end{aligned}$$

- 基底変数と目的関数を非基底変数の関数で表す形を**辞書**と呼ぶ

単体法(シンプレックス法)

Step 3　実行可能な初期解を選択する

- $x_1 = x_2 = 0$ を代入し,
  $$(x_1, x_2, s_1, s_2) = (0,0,70,180)$$
  を得る
- 目的関数の x_1, x_2 の係数が負なので，より大きい値を取れる
  <- どれくらい増やして良いか？

Step 4　非負制約を用いて非基底変数を更新する

Min　$u = -6x_1 - 4x_2$

s.t.
$$s_1 = 70 - 2x_1 - x_2$$
$$s_2 = 180 - 3x_1 - 4x_2$$

$x_2 = 0$ を固定し，$s_1, s_2 \geq 0$から，

$$x_1 \leq 35$$
$$x_1 \leq 60$$

$x_1 = 35$,
またこのとき $s_1 = 0$

単体法(シンプレックス法)

Step 5　非基底変数と基底変数を入れ替える

- 値が得られた非基底変数: x_1 を基底変数とし，
  対応して値が得られた基底変数: s_1 を非基底変数とするように
  辞書を書き換える

$$\begin{aligned} \textbf{Min} \quad & u = -6x_1 - 4x_2 \\ \textbf{s.t.} \quad & s_1 = 70 - 2x_1 - x_2 \\ & s_2 = 180 - 3x_1 - 4x_2 \end{aligned}$$

$$\begin{aligned} \textbf{Min} \quad & u = -210 - x_2 + 3s_1 \\ \textbf{s.t.} \quad & x_1 = 35 - \frac{1}{2}x_2 - \frac{1}{2}s_1 \\ & s_2 = 75 - \frac{5}{2}x_2 + \frac{3}{2}s_1 \end{aligned}$$

x_2 の係数が負なので，
さらに x_2 の値を増加させられる

- $x_2 = s_1 = 0$ を代入し，次の解
  $(x_1, x_2, s_1, s_2) = (35,0,0,75)$
  を得る

Step 6　Step 4, 5を目的関数の係数が全て 0 以上になるまで繰り返す

演習：単体法(シンプレックス法)

Step 4　非負制約を用いて非基底変数を更新する

$$\text{Min} \quad u = -210 - x_2 + 3s_1$$

$$\text{s.t.} \quad x_1 = 35 - \frac{1}{2}x_2 - \frac{1}{2}s_1$$

$$s_2 = 75 - \frac{5}{2}x_2 + \frac{3}{2}s_1$$

Step 5　非基底変数と基底変数を入れ替える

演習：単体法(シンプレックス法)

Step 4　非負制約を用いて非基底変数を更新する

Min　$u = -210 - x_2 + 3s_1$

s.t.　$x_1 = 35 - \frac{1}{2}x_2 - \frac{1}{2}s_1$

$s_2 = 75 - \frac{5}{2}x_2 + \frac{3}{2}s_1$

$x_2 \leq 70$

$x_2 \leq 30$

$s_1 = 0$ を固定し，$x_1, s_2 \geq 0$から，

$x_2 = 30$,
またこのとき $s_2 = 0$

Step 5　非基底変数と基底変数を入れ替える

非基底変数: x_2 を基底変数として
s_2 を新たに非基底変数とする

$x_2 = 30 + \frac{3}{5}s_1 - \frac{2}{5}s_2 \leftrightarrow s_2 = 75 - \frac{5}{2}x_2 + \frac{3}{2}s_1$

$u = -240 + \frac{12}{5}s_1 + \frac{2}{5}s_2$

$x_1 = 20 - \frac{27}{10}s_1 + \frac{1}{5}s_2$

$x_2 = 30 + \frac{3}{5}s_1 - \frac{2}{5}s_1$

最適な辞書
目的関数の係数が全て0以上

$s_1 = s_2 = 0$ を代入し，最適解
$(x_1, x_2, s_1, s_2) = (20,30,0,0)$ を得る

単体法の課題

- 初期解として$(x_1 = x_2 = 0)$ を採用したが，初期解によっては最適解に到達しないこともある
- この場合，最初に「条件を満たす初期解」を見つける 二段階単体法を使う
- 単体法を理解しよう！(水野, 2019) で詳しく解説されています.

引用　NTT Data 線形計画法

楕円体法・内点法

- 単体法(Dantzig, 1947)は理論的には多項式時間アルゴリズムではない

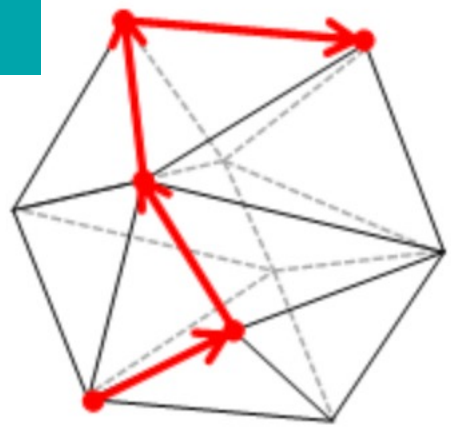
→ 後により理論的に効率的なアルゴリズムが提案された

Khachiyan(1979): **楕円体法**　　初の多項式時間アルゴリズムを提案

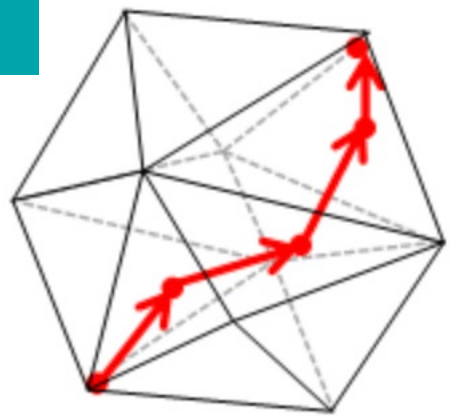
Karmarker(1984): **内点法**　　実用的にも優れた性能を持つ

- 単体法では，辺をたどり頂点から頂点に移動して最適解に近づく．
- これに対して，内点法では立体内を横切って解に収束する

単体法



内点法



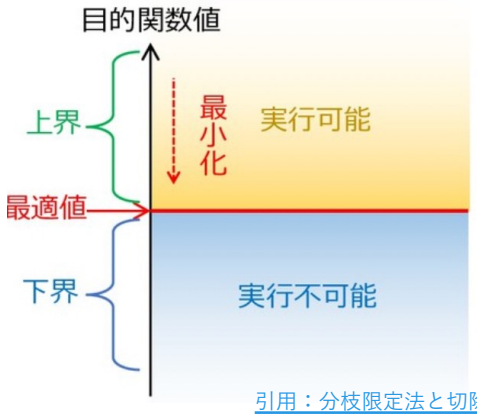
整数計画問題年表

今野浩, 役に立つ一次式-整数計画法「気まぐれな王女」の50年, 日本評論社, 2005
より抜粋

1947	Dantzig 単体法を提案
1954	Dantzig=Fulkerson=Johnson 42都市を対象とするTSPを解く
1956	Ford=Fulkerson ネットワーク・フロー理論を創始
1957	Gomory 切除平面法を提案し，整数計画法を立ち上げ
1960	Dantzig 整数計画法の重要性を明確にする論文を発表
1960	Land=Doig 分枝限定法を考案
1965	Balas 0-1整数計画問題に対する加法的解法を提案
1970	Held=Karp 大型巡回セールスマン問題を分枝限定法で解く
1971	Benichou et al. MPS/370(IBM370)をリリース
1971	Karp NP完全理論
1971	国際数理計画法学会を設立
1973	Padberg 集合被覆関数などのファセット構造を解明
1974	Balas 離接計画法を発表
1983	Crowder=Johnson=Padberg 大規模スケジューリング問題を解く
1984	Karmarkar 内点法を発表し，線形計画法の新時代を拓く
1987	Padberg=Rinaldi 大型巡回セールスマン問題を解く
1988	小島=水野=吉瀬=Megiddo 主・双対内点法を提案
1989	AT&Tベル研究所 KORBXをリリース
1990	Lustig=Marsten=Shanno OB1をリリース
1995	CPLEXがOB1を吸収し，New CPLEXに生まれ変わる
1998	BIXBY CPLEX 5.0 をリリース．整数計画ルーチン効率を改善
2003	ILOG CPLEX 9.0 をリリース．整数計画法の新時代到来
2003	Gomory=Johnson=Araoz コーナー多面体に関する論文を発表
2003	Dash Optimization XPRESS-MP ニュー・バージョンをリリース

線形計画問題の双対問題

- ある最適化問題の最適値の良い上界を求める問題を**双対問題**と呼ぶ
- このとき元の問題を**主問題**を呼ぶ



引用：分枝限定法と切除平面法

定義

主問題

$$\max \quad \Sigma c_j x_j$$
$$\text{s.t.} \quad \Sigma a_{ij} x_j \leqq b_i$$

$$\Sigma x_i(\Sigma a_{ij} y_i) \leqq \Sigma x_j c_j$$

$$\Leftrightarrow \Sigma x_j \underbrace{(\Sigma a_{ij} y_i)}_{b_j} \leqq \Sigma x_j c_j$$

$$\Sigma y_i(\Sigma a_{ij} x_j) \leqq \Sigma y_i b_i$$

$$\Leftrightarrow \Sigma x_j \underbrace{(\Sigma a_{ij} y_i)}_{c_j} \leqq \Sigma y_i b_i$$

双対問題

$$\min \quad \Sigma b_i y_i$$
$$\text{s.t.} \quad \Sigma a_{ij} y_i = c_j$$
$$y_j \geqq 0$$

双対性

主問題と双対問題の関係

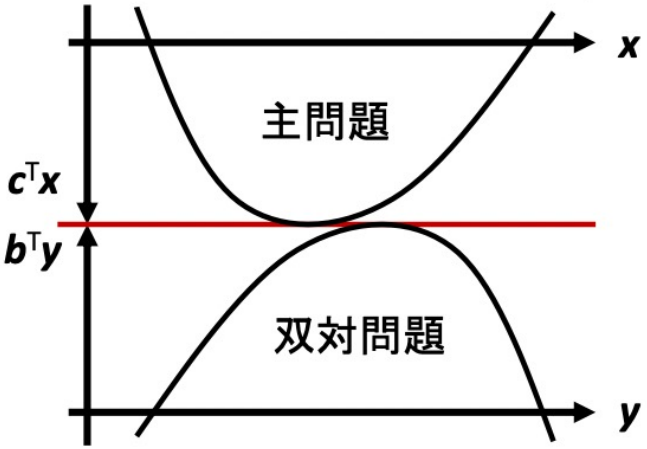
弱双対定理

主問題の実行可能解xの目的関数値 ≧ 双対問題の実行可能解yの目的関数値

$$c^{\mathrm{T}}x \geqq b^{\mathrm{T}}y$$

強双対定理

主問題が最適解を持つなら双対問題も最適解をもち，それらの最適値は一致



双対問題

	ビターチョコ x_1	ミルクチョコ x_2	使用可能量
カカオマス y_1	2	1	70(unit/day)
カカオバター y_2	3	4	180(unit/day)
利益	￥6 (/unit)	￥4 (/unit)	

あるメーカーA が利益最大化を
目指して生産計画を立てたい

ある企業B が買取価格を抑えつつ,
全ての生産要素を買収したい

主問題

$$\begin{aligned} \textbf{Max} \quad & 6x_1 + 4x_2 \\ \textbf{s.t.} \quad & 2x_1 + x_2 \leq 70 \\ & 3x_1 + 4x_2 \leq 180 \\ & x_1, x_2 \geq 0 \\ & x_1, x_2 \in \mathbb{R} \end{aligned}$$

双対問題

チョコ1単位の製造に使用する原料の量を買い取るには,
チョコ1単位の利益以上の金額を提示すべき

$$\begin{aligned} \textbf{Min} \quad & 70y_1 + 180y_2 \\ \textbf{s.t.} \quad & 2y_1 + 3y_2 \geq 6 \\ & y_1 + 4y_2 \geq 4 \\ & y_1, y_2 \geq 0 \\ & y_1, y_2 \in \mathbb{R} \end{aligned}$$

目的関数値が一致！

課題

発表： 5/6(金)

社基の五月祭の勤務シフトスケジューリング問題を定式化して，
その求解アルゴリズムを設計してみましょう．

参考図書

- 寒野善博, 土谷隆, 基礎系数学 最適化と変分法, 東京大学工学教程編纂委員会
- 久保幹雄, J.P. ペドロソ, メタヒューリスティクスの数理, 共立出版
- 相吉英太郎, 安田恵一郎, メタヒューリスティクスと応用, 電気学会
- J. ホロムコヴィッチ, 計算困難問題に対するアルゴリズム理論(組み合わせ最適化・ランダマイゼーション・近似・ヒューリスティクス), Springer(訳: 和田幸一, 増澤利光, 元木光雄)
- 藤沢克樹, 梅谷俊治, 応用に役立つ50の最適化問題, 朝倉書店, 2009
- 梅谷俊治, しっかり学ぶ数理最適化問題 モデルからアルゴリズムまで, 講談社, 2020