最適化

2023/04/19 15:00-16:30
スタートアップゼミ #4
D2 黛 風雅 / Mayuzumi Fuga

1. 最適化問題の導入

1.1　最適化とは
1.2　離散最適化と連続最適化
1.3　計算量と複雑性

2. 最適化問題の代表的解法

2.1　ネットワーク最適問題ー最短経路問題
2.2　線形計画問題ー単体法
2.3　多目的最適化とパレートフロンティア
2.4　双対問題

1. 最適化の導入

【目次】

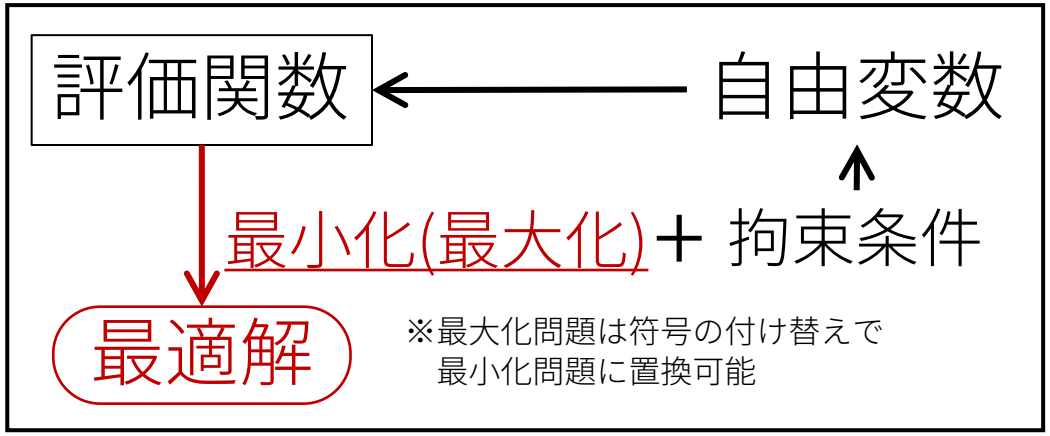
3.1　基本的な問題設定と停留条件
3.2　離散時間LQ制御問題
3.3　動的計画法

最適化問題とは

自由変数を含む工学的問題へのアプローチ
▶ 最も望ましい結果を得る自由変数を決定する

自由変数の関数として実数値で表現される場合(交通なら距離や費用など)
▶ 評価関数 (または目的関数/コスト関数)と呼ぶ

最適化問題



- 自由変数が有限個である
  最適化問題 >>> 数理計画問題

- 評価関数と拘束条件が
1次式のとき ▶ 線形計画問題(LP)
1次式以外のとき ▶ 非線形計画問題(NLP)
※特に2次式のとき ▶ 2次計画問題

⟺ 有限個の自由変数に限らず関数を決定したいとき，
評価関数は「関数の関数 ＝ **汎関数** 」になる．
汎関数の最小化/最大化問題 >>> **変分問題**

最適化問題とは

実際にはどう記述されるか？

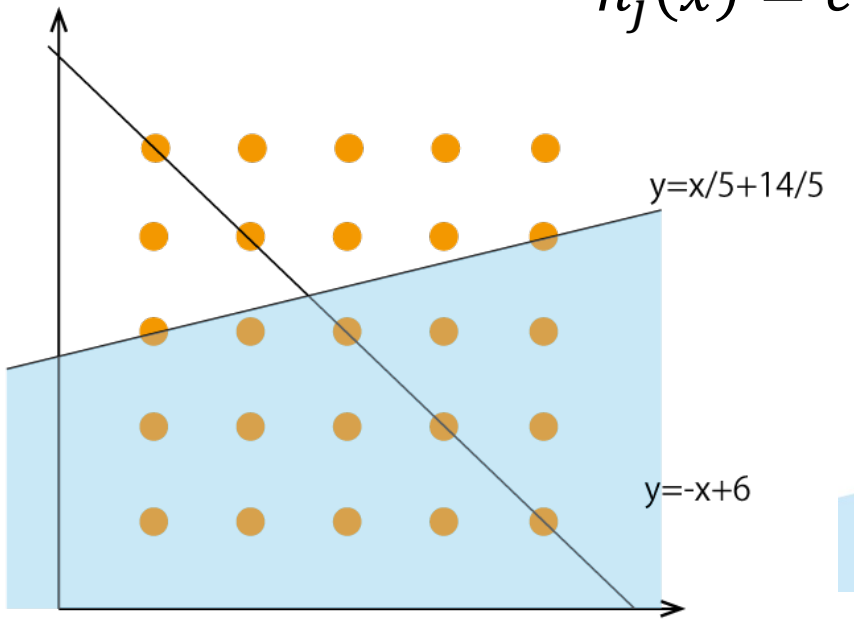
Minimize　$f(x)$　目的関数(最大化の場合は−f(x))

s.t.　$x \in S$　制約条件(拘束条件)

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

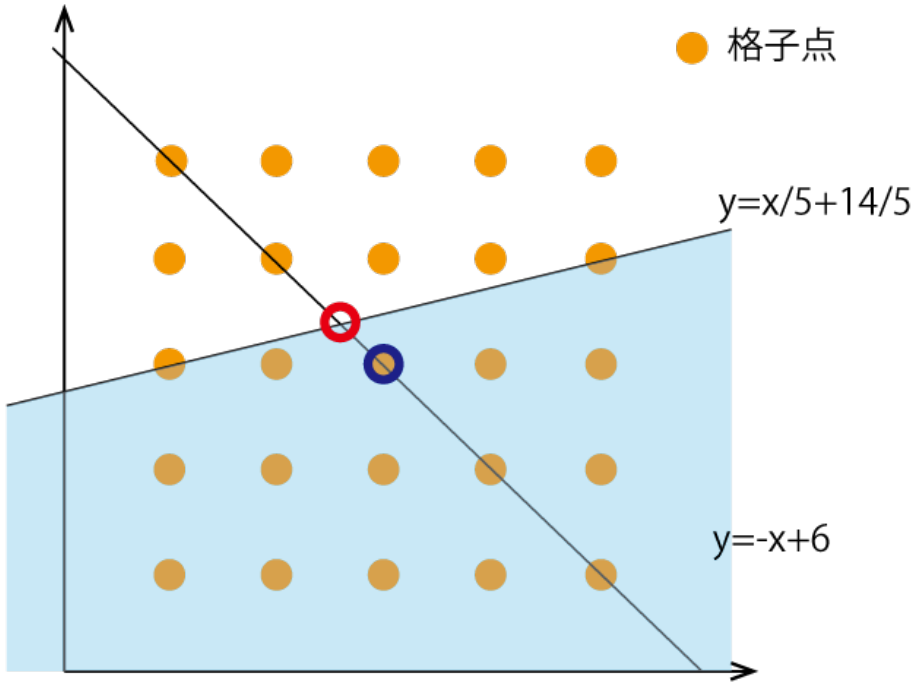
Max　$f(x) = -x + 6$

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

最適化問題とは

最適化問題は，変数が連続か離散かによって区別される



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

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

最適化問題の解

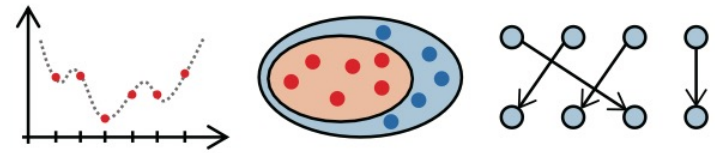
- ... xが連続変数($x \in \mathbb{R}$)のとき
- ... xが離散変数($x \in \mathbb{Z}$)のとき

- 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

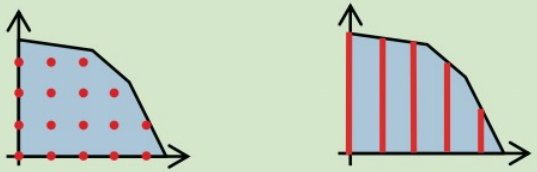
離散最適化と連続最適化

最適化問題は，変数が連続か離散かによって区別される

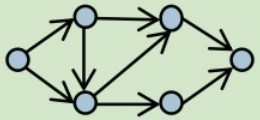
離散最適化（組合せ最適化）



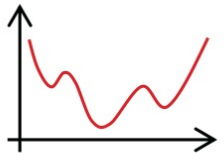
整数計画問題　混合整数計画問題



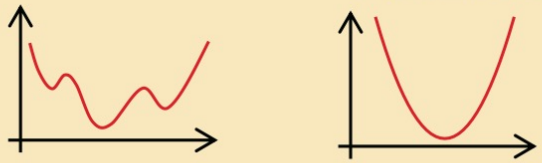
ネットワーク最適化問題



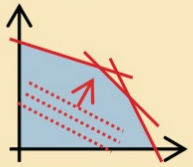
連続最適化



非線形計画問題　2次計画問題



線形計画問題



梅谷(2020)より引用

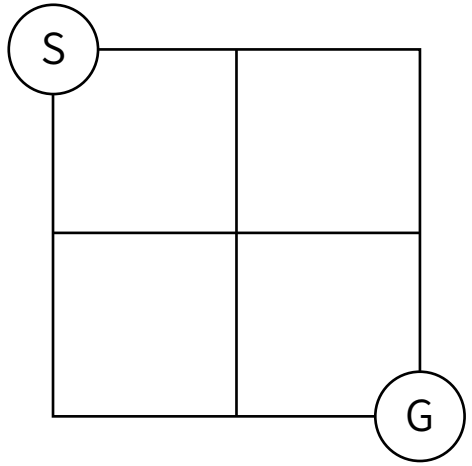
効率的な最適化アルゴリズムの必要性

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

アルゴリズムの計算量

計算の複雑さ(＝計算量)は２つの観点から評価される：

▶ 時間計算量 … 四則演算などの基本操作の実行回数

▶ 空間情報量 … 必要なメモリ量

オーダー記法 $O\left(\varphi(N)\right)$ を用いてこれらを評価

規模 N の問題をあるアルゴリズムによって $\varphi(N)$ の計算手間によって必ず解くことができるとき，そのアルゴリズムの計算量は $O\left(\varphi(N)\right)$ であるという．

$\varphi(N)$ の関数系によってアルゴリズムの分類がある：

▶ 多項式(Polynomial)時間アルゴリズム　　複雑性クラス　$\mathcal{P}$

$\varphi(N)$ がある定数 k を用いて N^k と書ける場合

▶ 指数時間アルゴリズム　　複雑性クラス　$\mathcal{NP}$

$\varphi(N)$ が2^Nや$N!$のように多項式函数では抑えられない場合

参考：オーダー記法

関数の極限における値の変化を大まかに評価するための記法

- 影響力の大きい（次数の大きい）項以外は無視する
- 定数倍の差は無視する

漸近的上界[ビッグ・オー記法：O]

十分大きな n について，定数倍を無視すれば計算量 $T(n)$ は $f(n)$ が上限となる

$$T(n) = O(f(n)) \quad \leftrightarrow$$

ある実数 c と自然数 n_0 が存在して，
全ての $n \geq n_0$ に対して $T(n) \leq c \cdot f(n)$ が成り立つ

例. $T(n) = 2n^2 + 3n + 1$ のとき，$f(n) = n^2$ $(2n^2 + 3n + 1 = O(n^2))$

※$f(n)$ は上限であるため，$f(n) = n^3$ でも成り立つ

→アルゴリズムの計算量の評価に用いられる
あるアルゴリズムについて n 個の入力データに対して必要な計算量のオーダーを示す

cf. 漸近的下界[ビッグ・オメガ記法：Ω]

十分大きな n について，定数倍を無視すれば計算量 $T(n)$ は $g(n)$ が下限となる

$$T(n) = O(g(n)) \quad \leftrightarrow$$

ある実数 c と自然数 n_0 が存在して，
全ての $n \geq n_0$ に対して $T(n) \geq c \cdot f(n)$ が成り立つ

組合せ最適化問題の類型と複雑性クラス

複雑性クラス	$\mathcal{P}$	$\mathcal{NP}$ 完全	$\mathcal{NP}$ 困難
最適化問題	線形最適化問題	充足可能性問題(3-SAT)	整数最適化問題
	最短路問題	巡回セールスマン問題 (決定問題)	巡回セールスマン問題
	最大流問題	最大安定集合問題 (決定問題)	最大安定集合問題
	最小費用流問題	ナップサック問題 (決定問題)	ナップサック問題
	最小全域木問題	ビンパッキング問題 (決定問題)	ビンパッキング問題
	割当問題	最大クリーク問題 (決定問題)	最大クリーク問題
	充足可能性問題(2-SAT)	最小頂点被覆問題 (決定問題)	最小頂点被覆問題
	最大マッチング問題		最小極大マッチング問題
	最大重みマッチング問題		
	最大(最小)重み 最大マッチング問題		

決定問題▶最適値ではなく目的関数がある値と比較して大きいかどうかをYes/Noで返す問題

複雑な問題に対しては近似解法やメタヒューリスティクス解法が開発されてきている

2. 最適化問題の代表的解法

【目次】

2.1　ネットワーク最適問題ー最短経路問題
2.2　線形計画問題ー単体法
2.3　双対問題
2.4　多目的最適化ーパレートフロンティア

ネットワーク最適化問題の類型

- 最短路問題(shortest path)
- 最小木問題(minimum spanning tree)
- 最大流問題(maximum flow)
- 最小費用流問題(minimum cost flow)
- 割当問題(assignment)/マッチング問題
- 多品種流問題(multi-commodity flow)

最短経路問題

１始点最短路問題の代表的解法

- 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

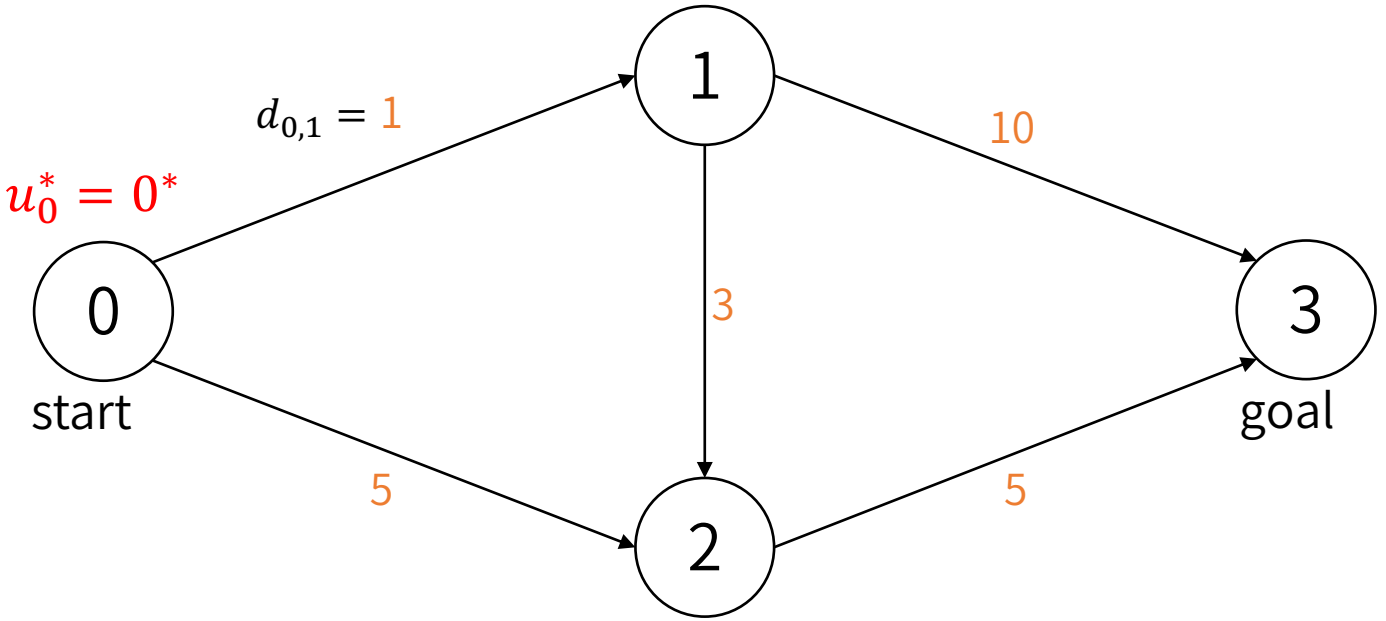
Dijkstra法

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を始点に持つ全リンクl_{ij}に対し，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．
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．

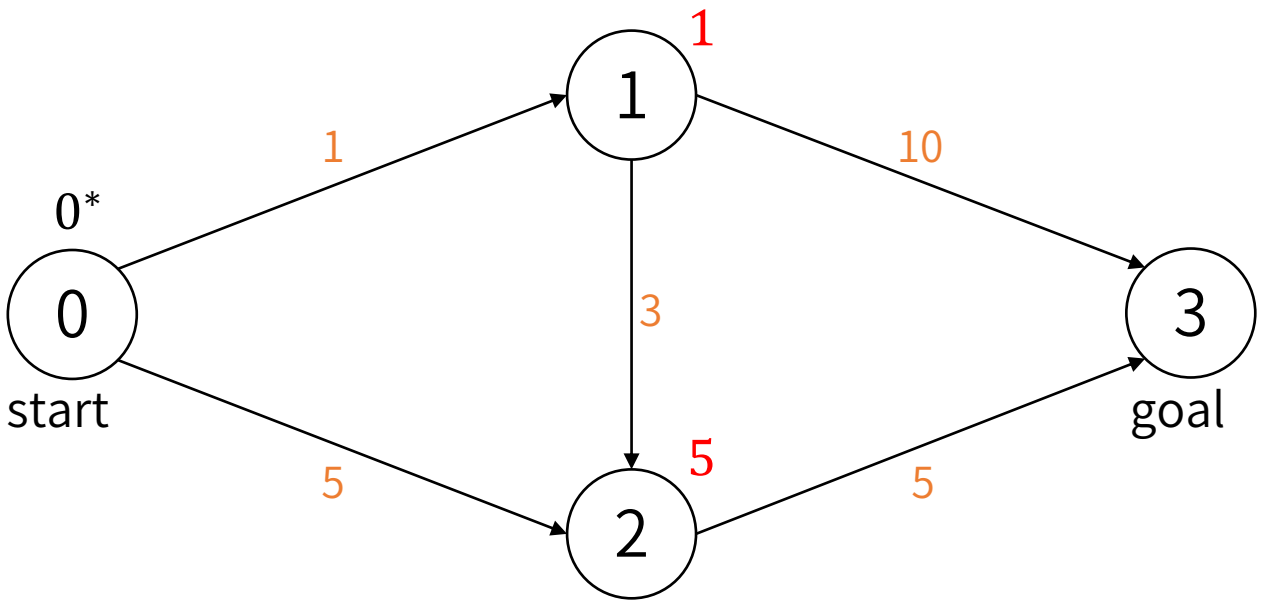
$i = 0$



Dijkstra法

3. ノードiを始点に持つ全リンクl_{ij}に対し，終点ノードjが永久ラベルを持っていないとき，
   i. (条件1)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. (条件2-1)ノードjが一時ラベルu_jを持っており，(条件2-2)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．
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．

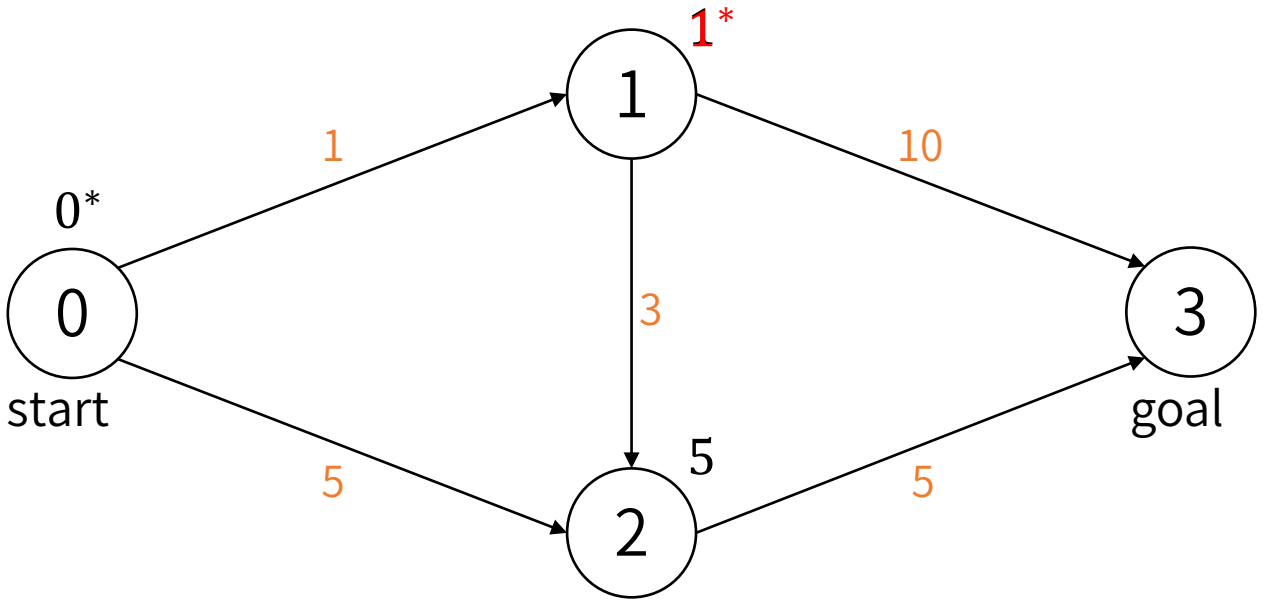
$i = 0$



Dijkstra法

3. ノードiを始点に持つ全リンクl_{ij}に対し，終点ノードjが永久ラベルを持っていないとき，
   i. (条件1)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. (条件2-1)ノードjが一時ラベルu_jを持っており，(条件2-2)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．
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．

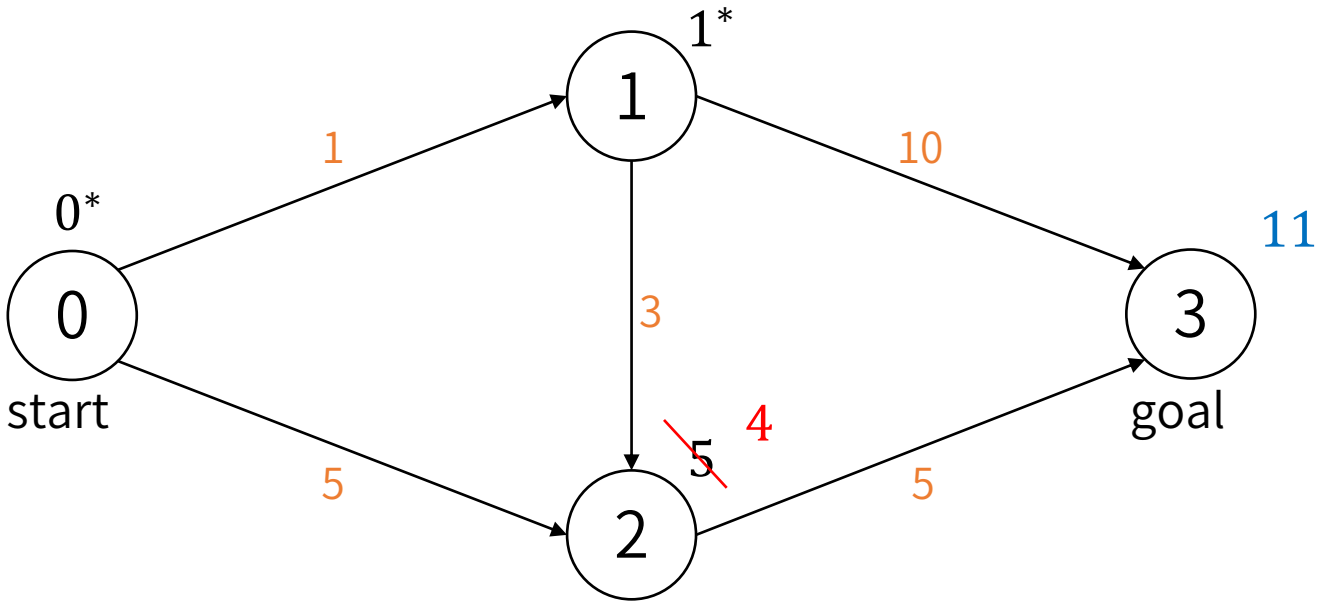
$i = 0$



Dijkstra法

3. ノードiを始点に持つ全リンクl_{ij}に対し，終点ノードjが永久ラベルを持っていないとき，
   i. (条件1)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. (条件2-1)ノードjが一時ラベルu_jを持っており，(条件2-2)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．
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．

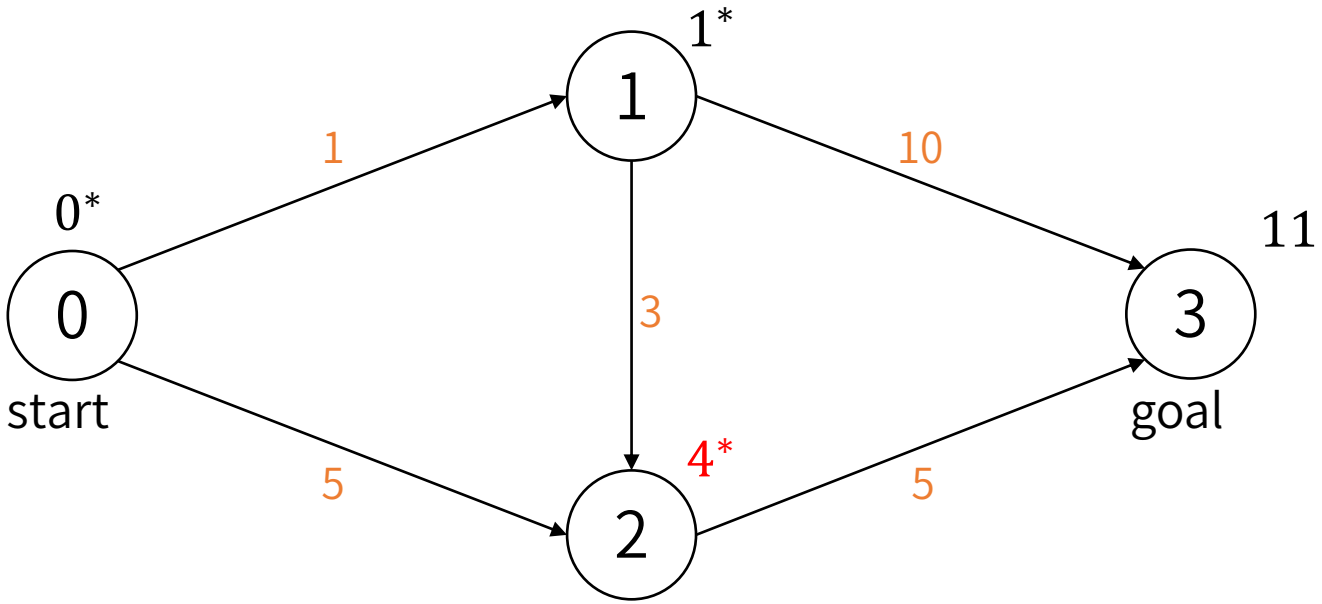
$i = 1$



Dijkstra法

3. ノードiを始点に持つ全リンクl_{ij}に対し，終点ノードjが永久ラベルを持っていないとき，
   i. (条件1)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. (条件2-1)ノードjが一時ラベルu_jを持っており，(条件2-2)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．
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．

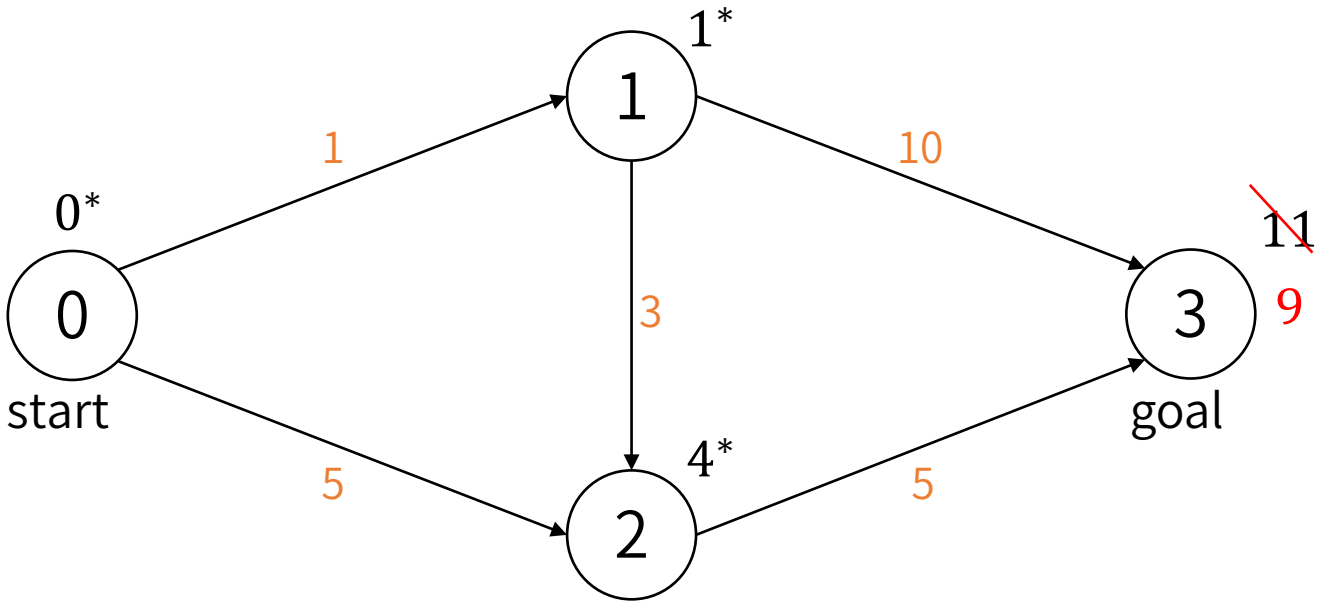
$i = 2$



Dijkstra法

3. ノードiを始点に持つ全リンクl_{ij}に対し，終点ノードjが永久ラベルを持っていないとき，
   i. (条件1)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. (条件2-1)ノードjが一時ラベルu_jを持っており，(条件2-2)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．
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．

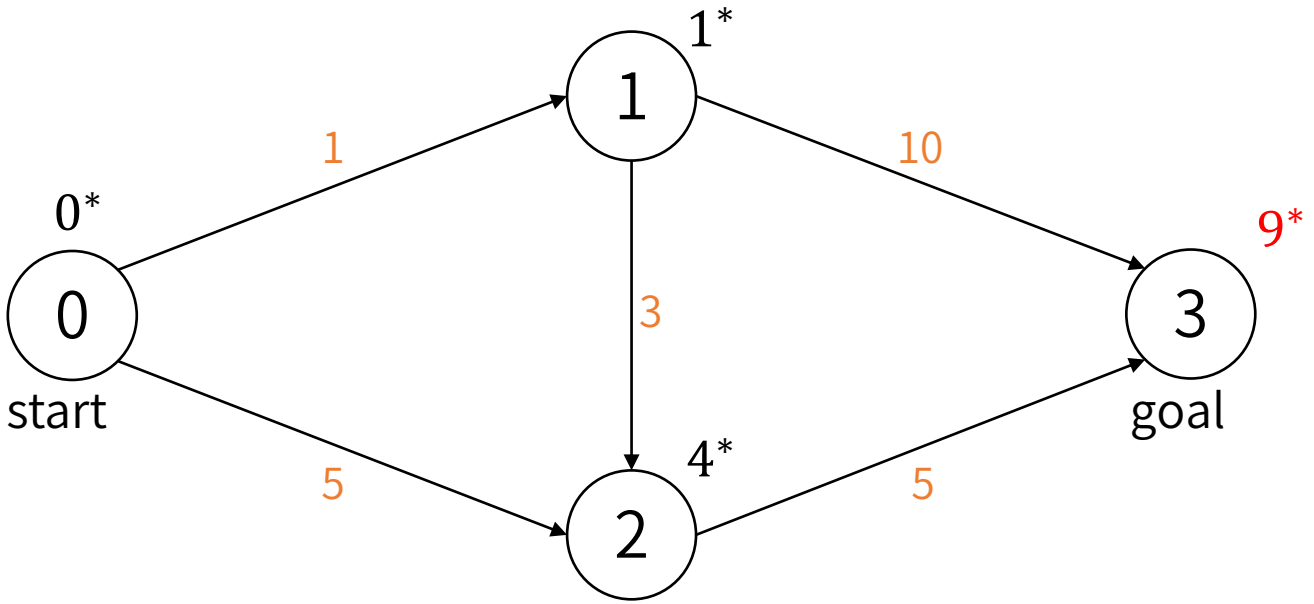
$i = 2$



Dijkstra法

3. ノードiを始点に持つ全リンクl_{ij}に対し，終点ノードjが永久ラベルを持っていないとき，
   i. (条件1)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. (条件2-1)ノードjが一時ラベルu_jを持っており，(条件2-2)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．
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．

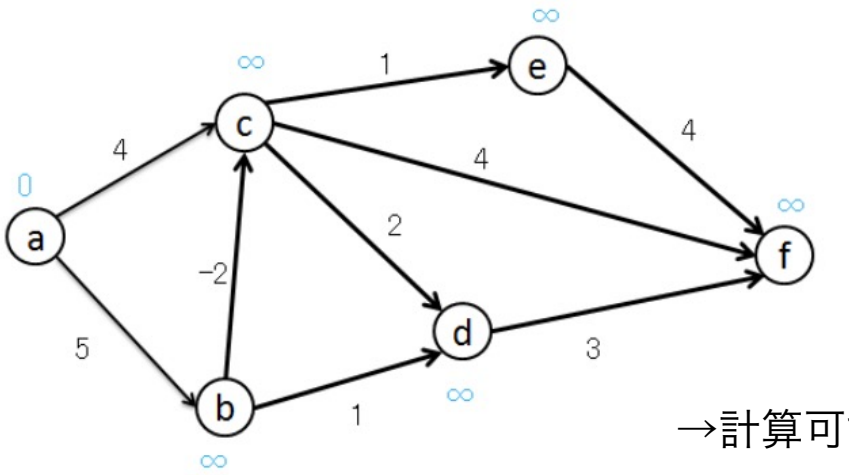
$i = 3$



Bellman-Ford法

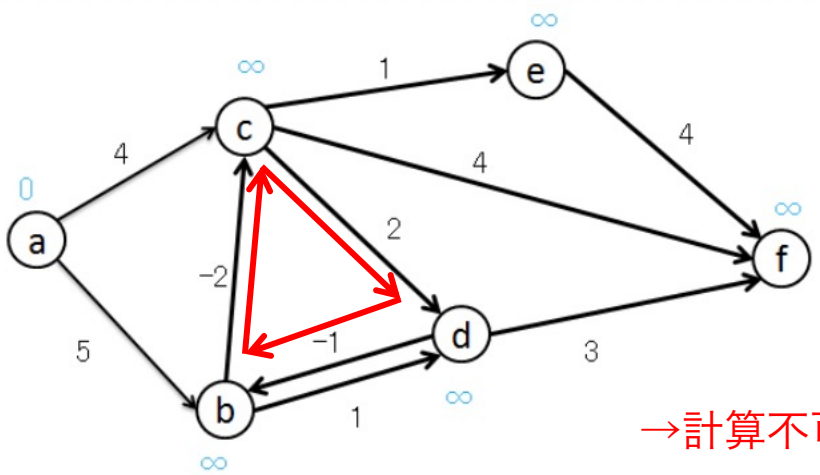
- 各頂点に与えられたコストを，より小さい値で置き換えていくアルゴリズム.
- Dijkstra法・A*アルゴリズムよりも計算量が多いが,
- コストが負の値の辺が含まれていても計算可能（Dijkstra法，A*アルゴリズムでは不可能）.
- ただし，グラフに負閉路 (negative cycle、辺の重みの総和が負になる閉路) が含まれるとき，その閉路を通過することでいくらでも重みを小さくできるので最短経路は決まらない.

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

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，ゴールまでの予想コスト$h^*(n)$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．
  ※ 実際の経路距離よりも小さくなくてはならないことに注意
  - ✓ マンハッタン距離$h(n) = x(n) + y(n)$
  - ✓ ユークリッド距離$h(n) = \sqrt{\{x(n)\}^2 + \{y(n)\}^2}$

ハンズオン①

- Dijkstra法をcolabで実装したい
- ChatGTPを使ってやってみる

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，混合整数計画問題(MIP)と呼ぶ

等式標準形

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

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

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

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

- 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とよい

線形計画問題(LP)

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)

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

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

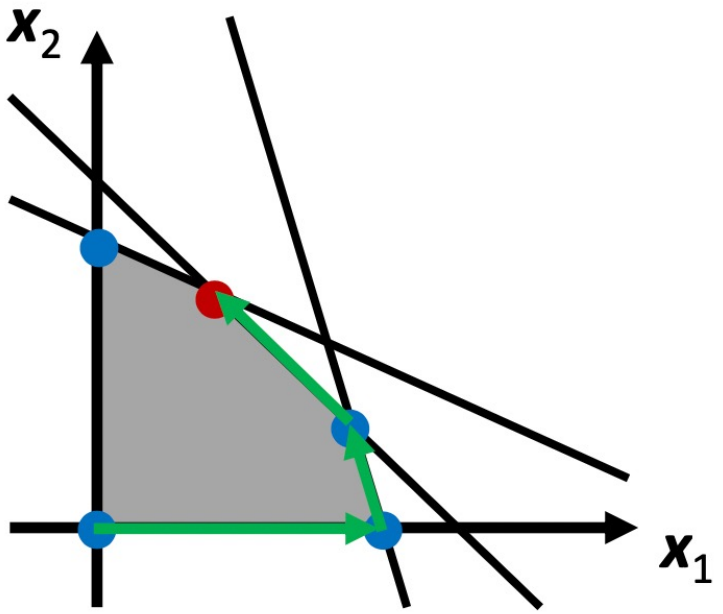
$$\begin{aligned} \text{Min} \quad & -2x_1 - (z_1 - z_2) \\ \text{s.t.} \quad & \\ & x_1 - (z_1 - z_2) + s_2 = 1 \\ & x_1, s_1, s_2, z_1, z_2 \geq 0 \end{aligned}$$

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 → $x_2 = z_1 - z_2$

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)から出発して，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



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)

例題) バスタ計画を単純化して評価する．
予算制約下で 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．

	駅前商業ビル 新規利用者数 x_1	鉄道駅 新規利用者数 x_2	予算
新モビリティ 導入費用 y_1	2	1	70(/year)
駅施設・広場 改修費用 y_2	3	4	180(/year)
利益率	￥6 (/unit)	￥4 (/unit)	

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

等式標準形

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

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

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

- 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

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

- 基底変数と目的関数を
  非基底変数(x_1, x_2)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

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

- $x_1 = x_2 = 0$ を代入し，以下の初期解を得る
  $(x_1, x_2, s_1, s_2) = (0,0,70,180)$
- 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

$$\begin{aligned} \text{Min} \quad & u = -6x_1 - 4x_2 \\ \text{s.t.} \quad & s_1 = 70 - 2x_1 - x_2 \\ & s_2 = 180 - 3x_1 - 4x_2 \end{aligned} \quad \Rightarrow \quad \begin{aligned} & x_1 \leq 35 \\ & x_1 \leq 60 \end{aligned}$$

初期解の非基底変数うちの1つを固定する
▶ $x_2 = 0$ とする

x_1 のとりうる最大値を考えると，
$s_1, s_2 \geq 0$ (非負制約)から，
$x_1 = 35$ が得られる．

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を基底変数とする
- 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

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

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

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

変換後も x_2 の係数が負なので，
x_2 の値を $x_2 = 0$ から増加させることで目的関数値をさらに減少させられると分かる

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

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

ヒント

- x_2 をどこまで大きくできるかが気になっている
- 先ほど得られた解は $(x_1, x_2, s_1, s_2) = (35,0,0,75)$

Step 5 非基底変数と基底変数を入れ替える

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

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

$s_1 = 0$ を固定し，$x_1, s_2 \geq 0$から，

$x_2 \leq 70$

$x_2 \leq 30$

$x_2 = 30$，
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(シンプレックス法)

例題)

バスタ計画を単純化して評価する．
予算制約下で x_1, x_2 を求める収益最大化問題を定式化して，単体法で解け．

	駅前商業ビル 新規利用者数 x_1	公共交通 新規利用者数 x_2	予算
新モビリティ 導入費用 y_1	2	1	70(/year)
駅施設/広場 改修費用 y_2	3	4	180(/year)
施設の利益率	￥6 (/unit)	￥4 (/unit)	

最適解 $(x_1, x_2, s_1, s_2) = (20,30,0,0)$ は得られたが，，，この問題は単純化しすぎている
実際にはどのような難しさがあるか？

・投資効果に対して予算制約(y_1, y_2や$y_1 + y_2$:総事業費)も変動する
　　▶パレートフロンティアの分析・ B/Cの評価

・新モビリティや駅施設/広場の改修が新規利用者を誘引する効果を定量化しなくてはならない
　　▶アクティビティモデルによる道路/歩道環境やアクセシビリティのパラメータの推定

生産計画との比較

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)	

バスタ計画を単純化して評価する．
予算制約下で x_1, x_2 を求める収益最大化問題を定式化して，単体法で解け．

	駅前商業ビル 新規利用者数 x_1	公共交通 新規利用者数 x_2	予算
新モビリティ 導入費用 y_1	2	1	70(/year)
駅施設/広場 改修費用 y_2	3	4	180(/year)
施設の利益率	￥6 (/unit)	￥4 (/unit)	

パレートフロンティア

- パレート解の集合を
  パレートフロンティアと呼ぶ
  - 目的関数同士がトレードオフ関係を持つ多目的最適化においては解を一意に定めることができない
  - それ以上パレート改善(他の目的関数を悪化させずに，ある目的関数を改善すること)ができない解を目的関数空間上にプロットして意思決定の助けにする

(例1)平均滞在時間/総期待効用と歩道幅員拡幅面積のトレードオフ関係を評価(大山&羽藤, 2017)

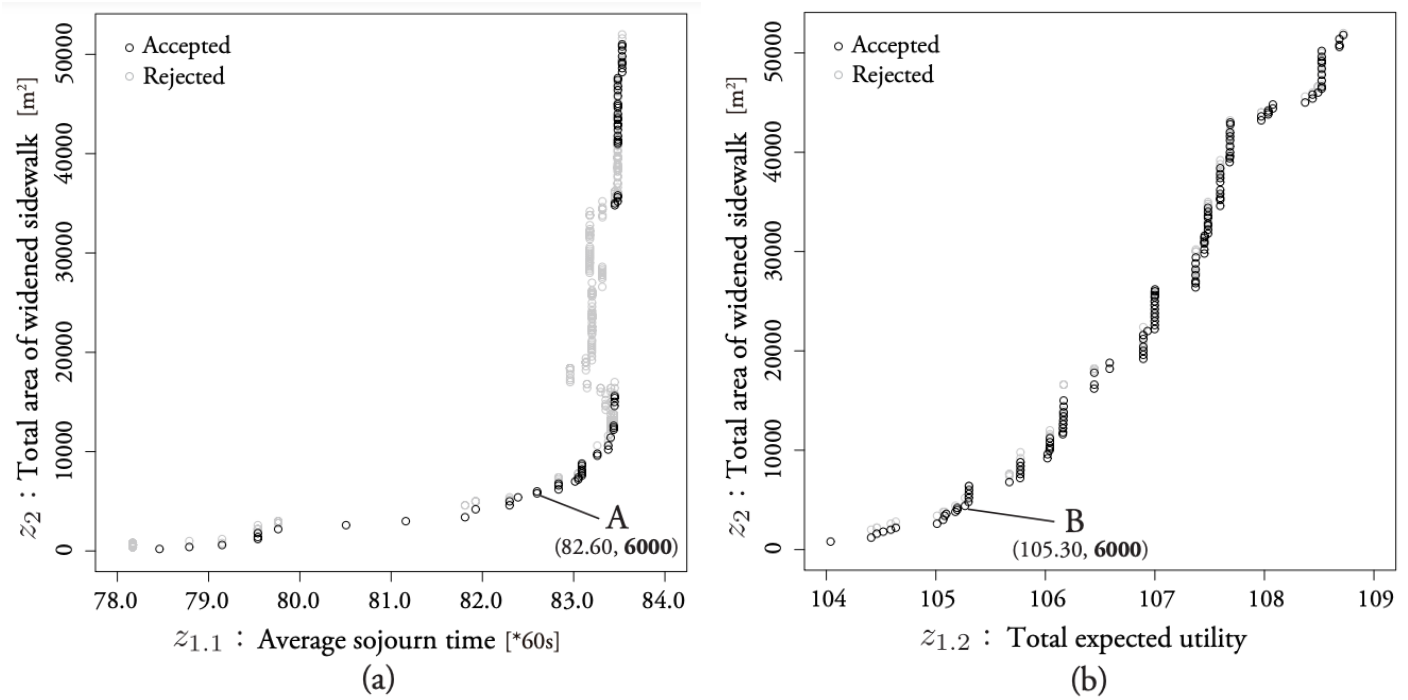
(例2) 復興自治体の補助金政策において補助金額を抑えつつ流出人口を最小化(小関&羽藤,2022)

大山雄己, & 羽藤英二. (2017). 多目的最適化に基づく歩行者の活動ネットワークデザイン. *都市計画論文集*, *52*(3), 810-817.

歩行者の活動ネットワークデザイン
における歩道幅員面積の効果の検証

$$v_t(a) = \underbrace{\theta_1 tt_a}_{\text{travel time}} + \underbrace{(\theta_2 x_a^w + \theta_3 x_a^s)\left(\frac{l_a}{L}\right)}_{\text{utility of moving sidewalk and shopping street}} + \underbrace{\theta_4 \int_{t\tau}^{(t+1)\tau} (x_a^c + x_a^d \omega) d\omega}_{\text{utility of staying}}. \quad (13)$$

リンク a についての効用関数の設定：
リンク長 l_a，歩道幅員 x_a^w，商店街ダミー x_a^s



パレートフロンティア

考えるべきトレードオフ関係は？

B4テーマ一覧(仮)

- 避難経路計画と最適制御
- 浜通り地域の最適NW設計
- 観光回遊空間の設計
- 土地所有と空間形成
- 広域的社会移動経路

cf. 楕円体法・内点法

- 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．

- 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

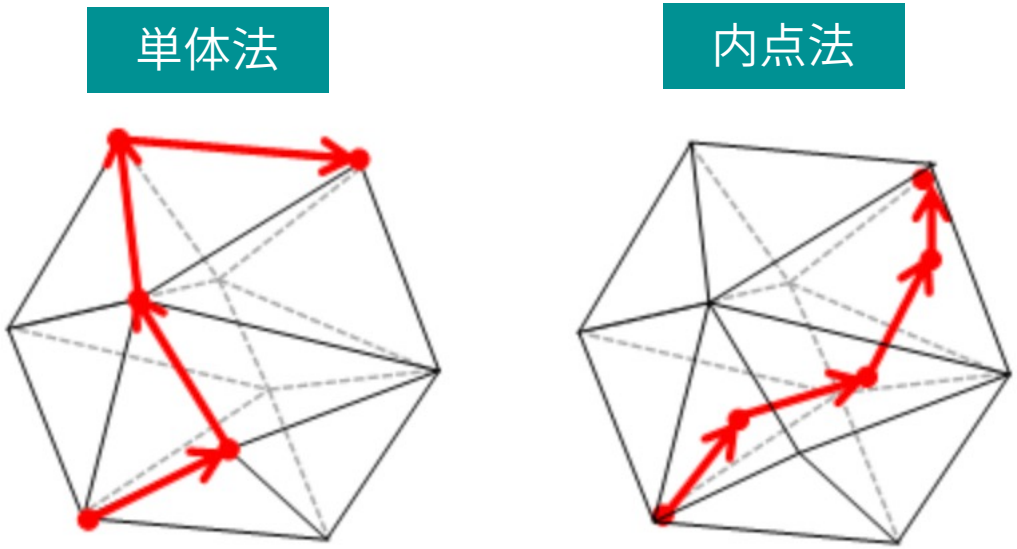
Khachiyan(1979): **楕円体法**

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



双対問題

最大化LPの目的関数値の上界を最小化する問題が双対問題である

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

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

主問題(P)

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

目的関数値が一致！

双対問題(D)

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

・弱双対定理

問題(P)と(D)が実行可能であるとし，$x \in X, y \in Y$ とする．次の不等式が常に成り立つ．

$$\boldsymbol{c}^{\mathrm{T}}\boldsymbol{x} \leq \boldsymbol{y}^{\mathrm{T}}\boldsymbol{b}$$

・強双対定理

問題(P)が最適解を持つための必要十分条件は問題(D)が最適解を持つこと

ハンズオン②

- 数理最適化パッケージpulpを使って，
  例題を解くコードをcolabで実装したい
- これもまたChatGTPを使ってやってみる

- 双対問題も同じようにpulpに入れて
  解が一致するか確認してみましょう

参考資料

- 穴井宏和, & 斉藤努. (2015). 今日から使える! 組合せ最適化: 離散問題ガイドブック. 講談社.
- 塩浦昭義 数理計画法(数理最適化)第7回 ネットワーク最適化
- 藤沢克樹, & 梅谷俊治. (2009). 応用に役立つ50の最適化問題, 朝倉書店.
- 梅谷俊治. (2020). 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.
- 水野眞治. (2019). 単体法を理解しよう!: 例題を使ったやさしい解説 (特集 はじめよう線形計画法). *オペレーションズ・リサーチ= Communications of the Operations Research Society of Japan: 経営の科学*, *64*(4), 209-217.
- 大山雄己, & 羽藤英二. (2017). 多目的最適化に基づく歩行者の活動ネットワークデザイン. *都市計画論文集*, *52*(3), 810-817.
- 並木 誠. (2008). 線形計画法 (応用最適化シリーズ). 朝倉書店.