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

3章(3.5-3.9)
数理計画とメタヒューリスティクスの融合

D1 浦田淳司

2013年7月10日

メタヒューリスティクス夏の祭典#5

担当部分の構成

3. 数理計画とメタヒューリスティクスの融合

3.1 分枝限定法

3.2 なぜ融合が必要か？

3.3 変数固定法

3.4 打ち切り分枝限定法と飛び込み法

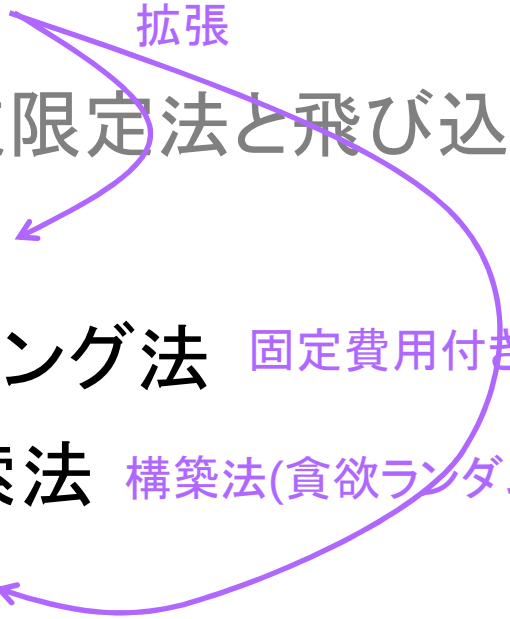
3.5 緩和固定法

3.6 容量スケーリング法

3.7 MIP近傍探索法

3.8 局所分枝法

3.9 MIP併合法



固定費用付き問題の解法

構築法(貪欲ランダム法等)の解の改善

複数解(蟻群生法など)から新しい初期解の生成

3.5 緩和固定法 (relax & fix method)

変数固定法の固定変数の集合を決定する操作で
MIP(混合整数計画, mixed integer programming)を利用

■ MIP問題に含まれる整数変数

- 自由変数：そのまま（元の問題と同じ変数）
- 固定変数：一次的に固定された変数
- 連続緩和変数：実数変数に緩和した変数

自由変数
固定変数
連続緩和変数

元の問題	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10
1回目	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10
2回目	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10
3回目	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10
最適解	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10

3.5 緩和固定法 (relax & fix method)

■ 自由変数の選択方法のポイント

・重要な変数（目的関数，制約条件に与える影響が大きい）
・解が密接な関係があり，同時に決定することが必要な変数の組合せ
・自由変数の個数が多くなりすぎないようにする

□ 例: ロットサイズ決定問題(lot sizing problem/Wagner-Whitin model)

期(月)	1	2	3	4
需要 d_t	2	4	5	1
固定費 a_t	12	20	16	8
製造変動費 v_t	3	3	3	3
在庫費 h_t	1	2	1	(1)

←既知

例:
カーシェアのポート設置有無と設置駐車マス
滞在/活動選択と滞在時間（未避難時間）
駐車場利用と駐車場利用時間（料金）

費用を最小にする生産計画{xt}を求める

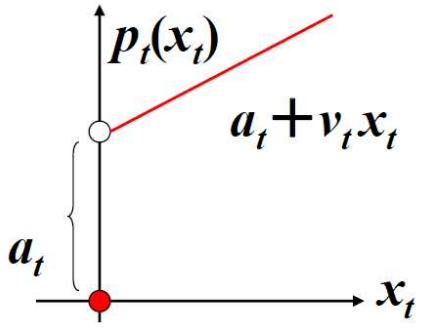
$$\min z = \sum_T (p_t(x_t) + h_t I_t)$$

$$s.t.\ I_{t-1} + x_t - d_t = I_t, \forall t \text{(流量保存)}$$

$$\text{在庫} I_t \geq 0, \text{生産量} x_t \geq 0$$

$$I_0 = 0, I_t = 0$$

製造関数p(x)は非連続



$$p_t(x_t) = \begin{cases} a_t + v_t x_t & x_t > 0 \text{のとき} \\ 0 & x_t = 0 \text{のとき} \end{cases}$$

3.5 緩和固定法 (relax & fix method)

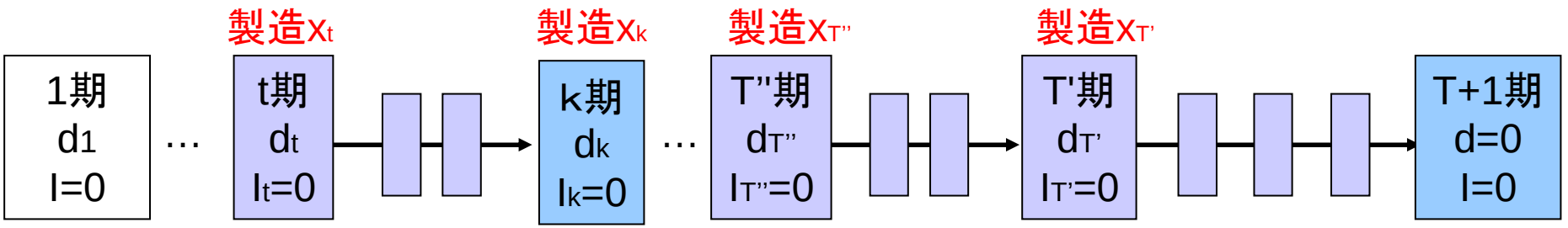
□ 例: ロットサイズ決定問題(lot sizing problem/Wagner-Whitin model)

解の性質

① $I_{t-1} \cdot x_t = 0$ 製造すると初期費用がかかるので,
在庫のみで需要をまかなう($I_{t-1}>0$, $x_t=0$)
または，在庫0(前期からの在庫管理コスト0)で製造のみを行う($I_{t-1}=0$, $x_t>0$)

② t期に製造する場合は，t-1期の在庫は0，またその先の需要も合わせて製造

1～T期で計算



近い期の変数を同時に自由変数とするほうが効率的

（緩和固定法での自由変数の決定方法）

3.6 容量スケーリング法(capacity scaling method)

固定費用付きの最適化問題の特殊構造を生かした汎用ヒューリスティックス

$$\text{minimize } vx + Fy$$ （変動費×生産数＋固定費×生産有無）

$$subject\ to$$

$$x \le Cy$$ （生産数≦生産容量×生産有無）

x=0でy=0, x>0でy=1

$$Ax + By \le b$$

$$x \in R^n, y \in \{0,1\}^n$$

$v \in R^n$： variable cost

$F \in R^n$： fixed cost

C $\in R^n$： 容量

x： variables, 実数値

y： variables, {0,1}

3.6 容量スケーリング法(capacity scaling method)

例：容量制約をもつ多品種フロー輸送ネットワーク設計問題

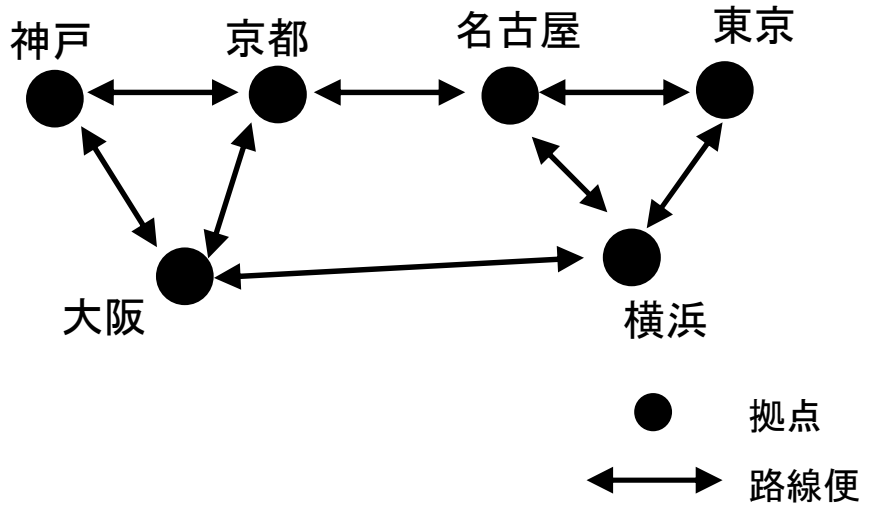
設定条件

・拠点iは所与
・品種kごとのij間の需要dkは所与
・どの拠点ij間も路線便開設可能
・ij間路線便の輸送容量Cijは所与
・ij間路線便の開設費fijは所与
・品種kの1単位あたりの輸送費用ckij

その他記号

・パスpがij間路線便を含むとき
　$\delta_{ijp}=1$ (含まないときは0)
・品種kのパスフロー量xkp(>0)
・ij間路線便の開設有無yij∈{0,1}

路線便例（輸送ネットワーク設計イメージ）



目的関数
制約条件

$$\min \sum_{(i,j)} \sum_{k} c_{ij}^{k} \sum_{p} \delta_{ijp} x_p^k + \sum_{(i,j)} f_{ij} y_{ij} \quad (1)$$

$$s.t \sum_{k} \sum_{p} \delta_{ijp} x_p^k \le C_{ij} y_{ij} \quad (2)$$

$$\sum_{p} \delta_{ijp} x_p^k \le d_k y_{ij} \quad (3)$$

$$\sum_{p} x_p^k = d^k \quad (4)$$

$$x_p^k \ge 0 \quad (5)$$

$$y_{ij} \in \{0,1\} \quad (6)$$

3.6 容量スケーリング法(capacity scaling method)

目的関数
制約条件

$$\min \sum_{(i,j)} \sum_{k} c_{ij}^{k} \sum_{p} \delta_{ijp} x_p^k + \sum_{(i,j)} f_{ij} y_{ij} \quad (1)$$

$$s.t \sum_{k} \sum_{p} \delta_{ijp} x_p^k \le C_{ij} y_{ij} \quad (2)$$

$$\sum_{p} \delta_{ijp} x_p^k \le d_k y_{ij} \quad (3)$$

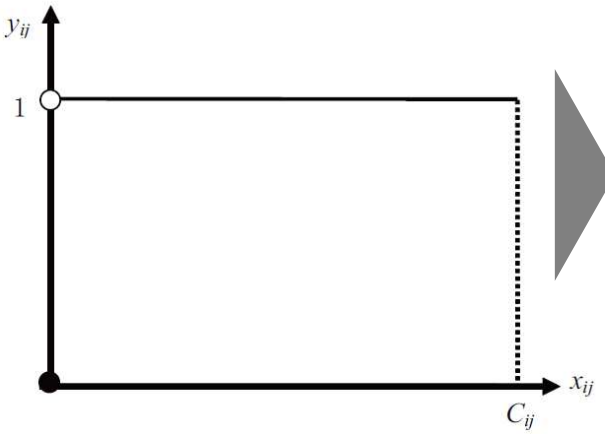
$$\sum_{p} x_p^k = d^k \quad (4)$$

$$x_p^k \ge 0 \quad (5)$$

$$y_{ij} \in \{0,1\} \quad (6)$$

(2)式に着目

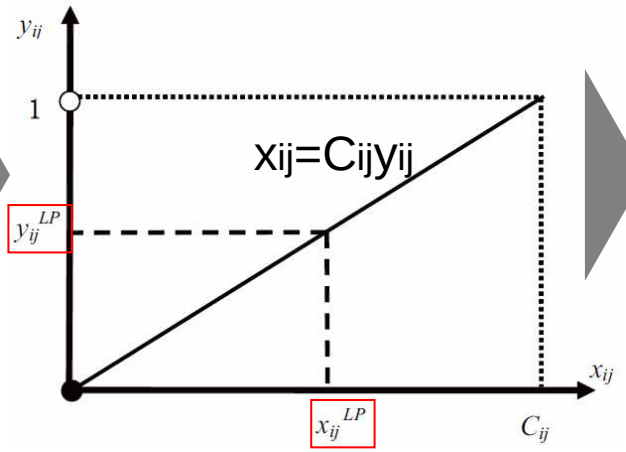
非線形のため計算し難い



x=0でy=0, x>0でy=1

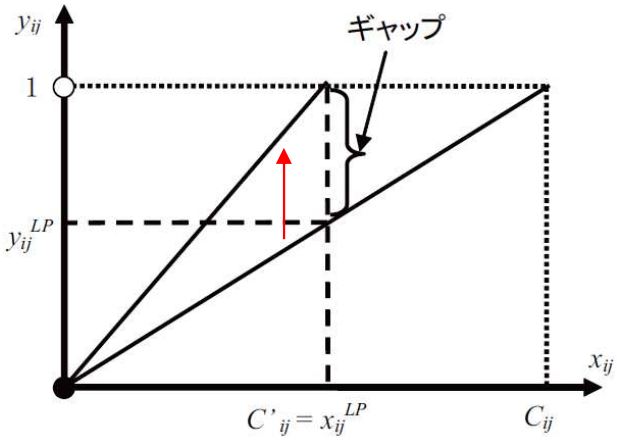
$$x_{ij} = \sum_{k} \sum_{p} \delta_{ijp} x_p^k$$

y_{ij}を線形に緩和($y_{ij} \in (0,1)$)



線形問題なので解ける
線形緩和解y_{LP}は0と1の間

y_{LP}が1になるように，
容量Cを変更



$C'_{ij}=x_{LP}$と変更して，次の計算

3.6 容量スケーリング法(capacity scaling method)

(2)式は非連続なので解けない

yを0~1の連続変数に緩和

t回目の計算

容量C(t)の元で，緩和線形問題を解く(x(t), y(t))

平滑化パラメータλを用い，容量C(t+1)に更新
C(t+1) = λC(t)y(t) + (1-λ) C(t)

理想の容量C'ではなく，少しずつ変えていく

yの範囲を次期の計算のために更新
x(t) ≦ C(t) かつ C(t+1)y(t) ≦ x(t) より
0 ≦ y(t) ≦ C(t)/C(t+1)

路線便開設変数yが0か1に収束するまで計算

$$\min \sum_{(i,j)} \sum_{k} c_{ij}^{k} \sum_{p} \delta_{ijp} x_p^k + \sum_{(i,j)} f_{ij} y_{ij} \quad (1)$$

$$s.t \sum_{k} \sum_{p} \delta_{ijp} x_p^k \le C_{ij} y_{ij} \quad (2)$$

$$\sum_{p} \delta_{ijp} x_p^k \le d_k y_{ij} \quad (3)$$

$$\sum_{p} x_p^k = d^k \quad (4)$$

$$x_p^k \ge 0 \quad (5)$$

$$y_{ij} \in \{0,1\} \quad (6)$$

3.7 MIP近傍局所探索法

- 構築法で得られた解を改善する方法として，MIP近傍局所探索法がある
- 一部の整数変数を自由変数としてMIP問題を最適化する
- 緩和固定法と同じく，自由変数の選択方法が重要になる

■ 緩和誘導近傍探索（relaxation induced neighborhood search）

- MIP近傍局所探索法を分枝限定法の各分枝子問題に適用

分枝子問題の緩和解と暫定解(current incumbent)が一致した変数
⇒ 変数を固定
⇒ 残りの変数でsub-MIP問題を探索

緩和解 $\bar{x} \in P(A,b,l,u)$，暫定解 $\tilde{x}$

$sub-MIP$で扱う自由変数$\{x\} = \left\{x \in P(A,b,l,u) \mid x_j = \tilde{x}_j, \bar{x}_j = \tilde{x}_j \wedge x_j\right\}$

$$\min cx$$

$$s.t.\ Ax \le b$$

$$cx \le (1-\lambda)c\tilde{x}$$ 暫定解の目的関数より小さくなるという制約条件を付加

$$x_j = \tilde{x}_j\ \forall j, \bar{x}_j = \tilde{x}_j$$

$$l \le x \le u$$

3.8 局所分枝法 (local branching method)

変数固定法，緩和固定法：一部の求めた解を固定して，他の解を計算

↓

局所分枝法：制約を与える

参照解 $\tilde{y}$ (実行可能解{0,1})

$(\tilde{y}_j = 1) \in \mathrm{N}_1$

$(\tilde{y}_j = 0) \in \mathrm{N}_0$

$\tilde{y}_j$と任意の解yとの距離$\Delta(y, \tilde{y})$

$$\Delta(y,\tilde{y}) = \sum_{j \in N_0, N_1} |\tilde{y}_j - y| = \sum_{j \in N_1} (1-y) + \sum_{j \in N_0} y$$

$\Delta(y, \tilde{y}) \le k$ (1) 高々k個以下の解の変更を制約⇒解の範囲を限定し，MIPソルバーで探索

※上界$\tilde{z}$以下の解を探索する制約

$$\sum f_j x_j + \sum g_j y_j \le \tilde{z}$$

3.8 局所分枝法 (local branching method)

MIPソルバーでの探索結果によって，次の探索方法を決定

1) 最適解を算出した場合

・制約(1)を付加した問題の最適解yを取得
・測深制約を追加

$$\Delta(y, \tilde{y}) \geq k+1$$

最適解を求めていない範囲

2) 実行不可能であることが証明された場合

・現在の参照解の近傍には良い解がない
・測深制約を追加

近傍を拡張

$$\Delta(y, \tilde{y}) \leq k' \ (k' > k)$$

3) 実行可能解を返した場合

・(最適解の保証はないので)得た解を参照解yとして，探索を続ける
・同じ解を探索しないように，tabu制約を付加

$$\Delta(y, \tilde{y}) \geq 1$$

4) 制限時間内に実行可能解が得られなかった場合

a) 制約を小さくして再び探索する

$$\Delta(y, \tilde{y}) \leq k' \ (k' < k)$$

b) 別の参照解を得るため，禁断制約を付加，かつ近傍を大きくして，再探索

$$\Delta(y, \tilde{y}) \geq 1$$
$$\Delta(y, \tilde{y}) \leq k' \ (k' > k)$$

3.9 MIP併合法 (MIP merging method)

複数の異なる解の保持と新しい初期解の生成（蟻群生法，GAなど）

MIPソルバーを利用

- 複数の異なる解に含まれる変数(y_j=1)を自由変数とする
- MIPソルバーで解を探索
- 現在の解と異なる解を得るためには，tabu制約を付加

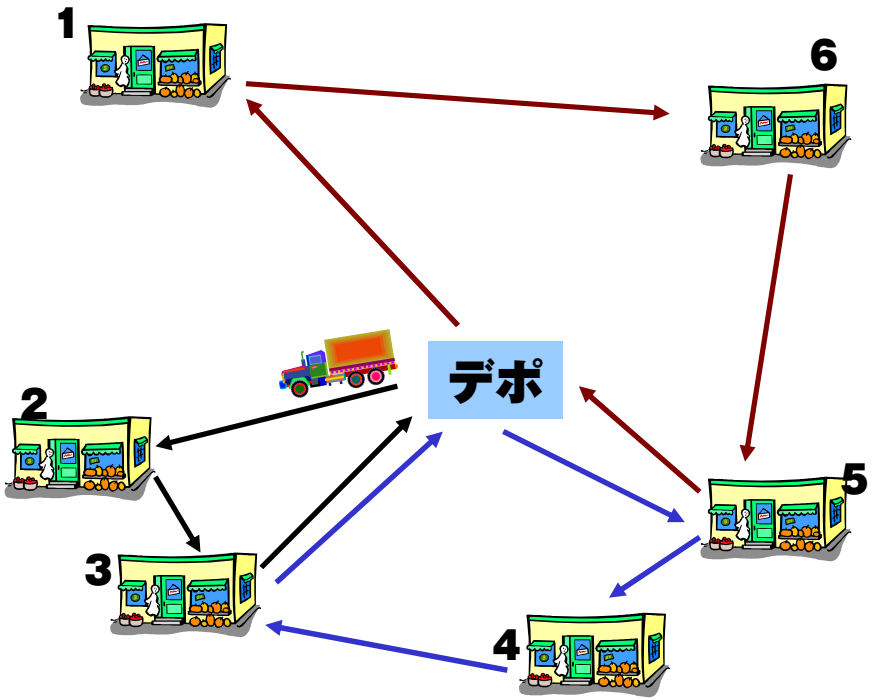
（初期解を生成して，改善法を適用）

- 遺伝的アルゴリズムにおける交叉の一般化
- 集合分割アプローチもMIP併合法の一種

3.9 MIP併合法 (MIP merging method)

集合分割アプローチ

Single-VRP問題



ルートの列挙

費用	3	8	5	4	4	5	…
店1	0	1	0	0	1	0	…
店2	1	0	0	0	0	1	…
店3	1	0	1	0	0	1	…
店4	0	0	1	0	1	1	…
店5	0	1	1	1	0	0	…
店6	0	1	0	1	0	0	…

- 全ての店を1回ずつ通過するルートの組合せを構成
- 複数解の併合だけでは新たな解が得られない場合は，対象外の解も自由変数にして探索