「基礎OR／OR演習」
第6回

- 中間試験日程
  2017／11／14（火）
- 時間：　13：00～14：30（3限）
- 場所56－103教室
- １１／１４の4限：招聘講師による特別講義（出席をとります）
- 所健一先生（電力中央研究所、博士（工学））

主なポイント

- ①線形計画法：鉄鋼電力労働力の問題
  - 単体法計算、双対最適解
  - 最適性、感度分析、基底形式、基底解など用語
- ②双対問題：第3-4回配布資料（生産計画、栄養問題）
  - グラフによる表示、相補スラック条件の説明
- ③DEA：学部評価モデル
  - 入力ー出力表からLP定式化、意味付け
- ④⑤ネットワーク最適化
  - 最短路
  - 輸送問題
- ⑥⑦整数計画法、組合せ最適化
  - 分枝限定法
  - 動的計画法

整数計画問題(IP)
(Integer Programming Problem)

- 整数計画問題　＝
  線形計画問題　＋　変数の整数条件
- 多くの、重要な、組み合わせ「最適化問題」(Combinatorial Optimization Problem)が整数計画問題として表現可能
- 一般に、線形計画問題より解きにくい
- 解析的には簡単そうに見えるが、難しい問題

問題例1：ナップサック問題（と等価）
エベレスト登山

長年の夢であったエベレスト登山に挑戦しているAさんは、いよいよ最後のテントから頂上をめざすことになった。空気が薄いこともあって、リュックに詰めるものの重量はW kgに抑えなければならないが、頂上まで持ってあがりたいものは多い。

持ってゆくものの候補$j=1,\ldots,n$とその重量a_jと候補jを持っていったときの「望ましさ」c_jは既知である。各候補はリュックに入れるか入れないかという選択肢しかない（量の調整が不可能である）としたとき、重量制限の中で、どの品物をつめたら望ましさを最大にできるか。

ナップサック問題（と等価） エベレスト登山

- 問題データ： 候補の品目（1,2,…,n)
  ナップザックの重量制限W(kg)；
  各品目の重量a_j(kg)；
  各品目の望ましさ（「価値」）c_j
- 変数（＝列）の定義： x_j＝品目jをつめるとき1
  さもなくば0
- 目的関数（総価値最大）：
  最大化　$z = \Sigma_j \; c_j x_j$
- 制約条件（容量制約を満足）：
  制約　$\Sigma_j \; a_j x_j \leqq W$

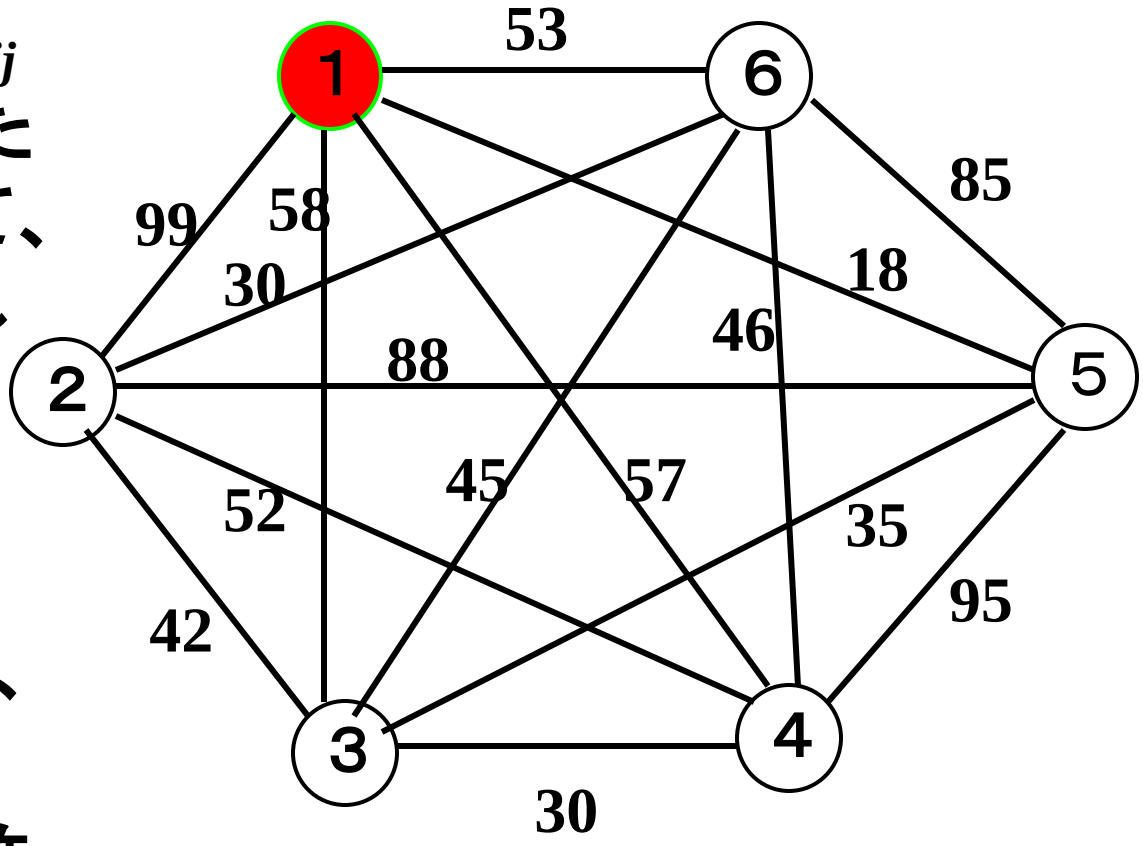
ナップサック問題
(Knapsack　Problem)

- 選択肢がとるかとらないかに限られている場合、0−1ナップサック問題と呼ばれる
- この他にも、整数値ナップサック問題（変数が非負整数値）、多重選択条件付きナップサック問題等、様々なバリエーションがある。

問題例2：巡回セールスマン問題

Traveling Salesman Problem（TSP)

- n個の点1,2,…,nと、任意の点間の「コスト」c_{ij}が所与であるとき、（たとえば）点1を出発して、各点を1回ずつ訪問して、点1に戻る最小コストの巡回路（Hamilton tour）を求めよ。（枝がなければ、$c_{ij}=\infty$）
- $c_{ij}=(\neq)c_{ji}$のとき　対称（非対称）型TSP



(非対称)巡回セールスマン問題
定式化

- 問題データ: 「点」$(1,2,\ldots,n)$
  点間の「コスト」行列$C=[c_{ij}]$
- 変数(＝列)の定義: x_{ij}＝「枝」ij を「通る」とき 1、
  さもなくば0
- 目的関数(総コスト最小):
  最小化　$z = \Sigma_j \Sigma_i c_{ij} x_{ij}$
- 制約条件(各配送先を1回ずつ訪問):
  制約　$\Sigma_j \ x_{ij}=1 \quad \forall i$ （または、$i=1,2,\ldots,n$）
  $\Sigma_i \ x_{ij}=1 \quad \forall j$ （または、$j=1,2,\ldots,n$）
  「部分巡回路ができない」(部分巡回路除去)

部分巡回路除去制約

- 「部分巡回路ができない」(部分巡回路除去)
- **部分集合Sの中の枝は高々$|S|-1$本**(ただし、$|S|$はSを構成する点の数)

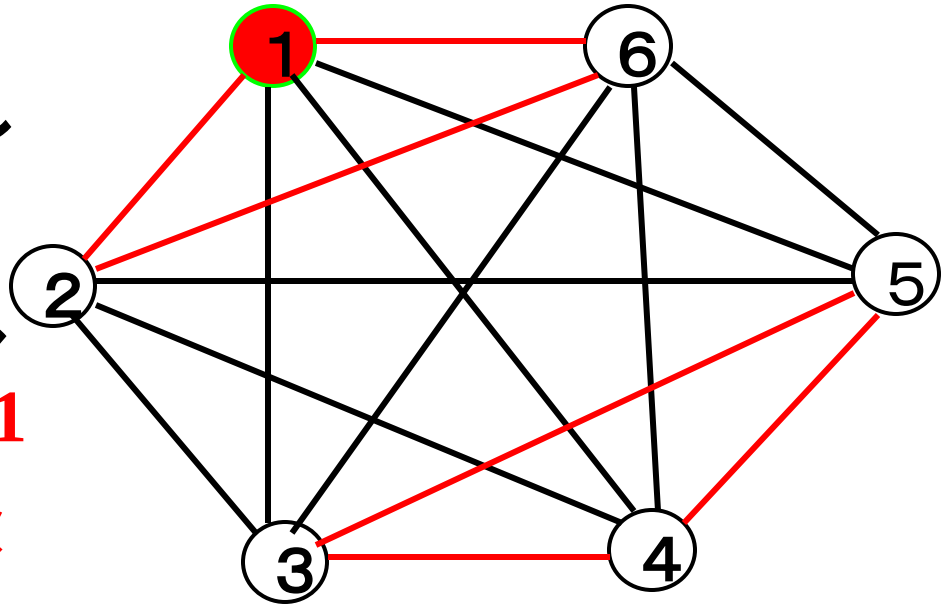
$$\Sigma_{i\in S}\Sigma_{j\in S}\ x_{ij} \leqq |S|-1,$$
$$\forall S\subset N(\text{点集合}),\ 2\leqq |S|\leqq |N|-1$$

- または、点を2つの部分集合に分けたとき、必ず、片方からもう一方に行けなくてはいけない

$$\Sigma_{i\in S}\ \Sigma_{j\in N\setminus S}\ x_{ij} \geqq 1,$$
$$\forall S\subset N,\ S\neq\varphi(\text{空集合}),\ S\neq N$$

巡回セールスマン問題における部分巡回路

- 各点に入る辺の数と各点から出る辺の数が1つでも、部分巡回路（subtour）できてしまう
- 点集合S={3,4,5}と定めると、
- $\Sigma_{i \in S} \Sigma_{j \in S} \ x_{ij} \ \leqq \ |S| - 1$
- この制約は部分巡回路を除去する
- 切除平面法による多面体的アプローチ（厳密解を求める）：数学的に高度



問題例3：勤務シフトスケジューリング

- 各勤務シフトに対する作業者を決めたい
- 月曜から金曜まで勤務し土日を休日とするシフト、火曜から土曜まで勤務し日月を休日とするシフト、．．．、日曜から木曜まで勤務し金土を休日とするシフトがある
- 各曜日に最低限必要な作業者数は既知月17名、火13名、水15名、木19名、金16名、土11名、日7名
- 各シフトに何名を割り当てれば総人数を最小化できるか

勤務シフトスケジューリング問題 定式化

- 問題データ:
  勤務シフト($j=1,2,\ldots,n$)、各曜日 ($i=1,2,\ldots,m$); 勤務シフト jが曜日iの勤務を含むとき$a_{ij}=1$(さもなくば0)とする「勤務シフト／曜日」行列$A=[a_{ij}]$, 勤務jを行うコストc_j, 各曜日の 必要な作業者b_i
- 変数(＝列)の定義: x_j＝勤務シフトj の人数(整数)
- 目的関数(総コスト最小):
  最小化 $z = \Sigma_{j=1,\ldots,n} c_j x_j$
- 制約条件(各曜日の必要人数を充足):
  制約 $\Sigma_{j=1,\ldots,n} a_{ij} x_j \geqq b_i \quad \forall i$ (または$i=1,2,\ldots,m$)

勤務シフトスケジューリング：excelデータ

勤務シフトスケジューリング											
										総人数	
	1	1	1	1	1	1	1				
	月から金	火から土	水から日	木から月	金から火	土から水	日から木	稼動人数		必要人数	
月	1	0	0	1	1	1	1		>=	17	月
火	1	1	0	0	1	1	1		>=	13	火
水	1	1	1	0	0	1	1		>=	15	水
木	1	1	1	1	0	0	1		>=	19	木
金	1	1	1	1	1	0	0		>=	16	金
土	0	1	1	1	1	1	0		>=	11	土
日	0	0	1	1	1	1	1		>=	7	日

エアラインのクルースケジューリング
EXCELソルバーシートの一例

クルースケジューリング											
										総人数	
	1	1	1	1	1	1	1				
	月から金	火から土	水から日	木から月	金から火	土から水	日から木	稼動人数		必要人数	
月	1	0	0	1	1	1	1		>=	17	月
火	1	1	0	0	1	1	1		>=	13	火
水	1	1	1	0	0	1	1		>=	15	水
木	1	1	1	1	0	0	1		>=	19	木
金	1	1	1	1	1	0	0		>=	16	金
土	0	1	1	1	1	1	0		>=	11	土
日	0	0	1	1	1	1	1		>=	7	日

c

x

A

b

写真提供:シンガポール航空

問題例3’:コンサート会場の警備

- 問題データ: 配置候補地点(1,2,…,n)、警備すべきブロック(1,2,…,m);ブロック i が候補地点 j から監視可能ならa_{ij}＝1(さもなくば0)とする「監視可能」行列$A=[a_{ij}]$
- 変数(＝列)の定義: x_j＝警備員を地点 j に配置するとき1、さもなくば0
- 目的関数(警備員を配置する地点数最小):
  最小化　$z = \Sigma_j x_j$
- 制約条件(各ブロックは監視可能):
  制約　$\Sigma_j a_{ij}x_j \geqq 1 \quad \forall i$ (または、$i=1,2,\ldots,m$)

問題例3’’：子会社の再編成

- 問題データ：　子会社(1,2,…,m), および子会社のグループ(1,2,…,n)；子会社iがグループjに含まれているならa_{ij}＝1（さもなくば0）とする「子会社組み合わせ」行列$A=[a_{ij}]$、組み合わせjのコストc_j
- 変数（＝列）の定義：　x_j＝子会社の組み合わせjを採用するとき1、さもなくば0
- 目的関数（総費用最小）：

  最小化　$z = \Sigma_j \; c_j x_j$
- 制約条件（各子会社はどこか1グループに加わる）：

  制約　$\Sigma_j \; a_{ij} x_j = 1 \quad \forall i$　（または、i＝1,2,…,m）

集合被覆問題
Set Covering Problem

- 集合$M=\{1,2,\ldots,m\}$とその部分集合の集まり（族）$P_1, P_2, \ldots, P_n$が所与
- P_jの集まりからなる集合族$\{P_j, j\in K\}$は、$\cup_{j\in K} P_j = M$を満たすとき、「被覆」と呼ぶ。
- P_jのコストc_jが所与のとき、最小費用の被覆を求める問題を「集合被覆問題」と呼ぶ。
- 被覆が以下を満たすとき、「分割」と呼ぶ:

  $P_j \cap P_k = \varphi$（空集合）; $j,k\in K,\ j\neq k$

- $P_1, P_2, \ldots, P_n$が与えられていないとき、これらを作り出さなければならない（何曜日に勤務するか、どの区間を飛ぶか、どの子会社を組み合わせるか…）

集合分割問題
Set Partitioning Problem

- 集合$M=\{1,2,\ldots,m\}$とその部分集合の集まり（族）$P_1, P_2, \ldots, P_n$が所与
- P_jの集まりからなる集合族$\{P_j, j\in K\}$は、
  $\cup_{j\in K} P_j = M$、かつ、
  $P_j \cap P_k = \varphi$（空集合）； $j,k\in K,\ j\neq k$
  を満たすとき、「分割」と呼ぶ。
- P_jのコストc_jが所与のとき、最小費用の分割を求める問題を「集合分割問題」と呼ぶ。
- $P_1, P_2, \ldots, P_n$が与えられていないとき、これらを作り出さなければならない（何曜日に勤務するか、どの区間を飛ぶか、どの子会社を組み合わせるか…）

集合被覆／分割問題の特徴

Set Covering／Partitioning Problem

- 「費用」最小化の問題
- 制約条件左辺の係数行列Aの要素は0または1（0－1係数行列）
- 右辺定数は1のベクトル
- 左開きの不等式　$Ax \geqq 1$　（集合被覆）
  （「少なくとも1回被覆」）

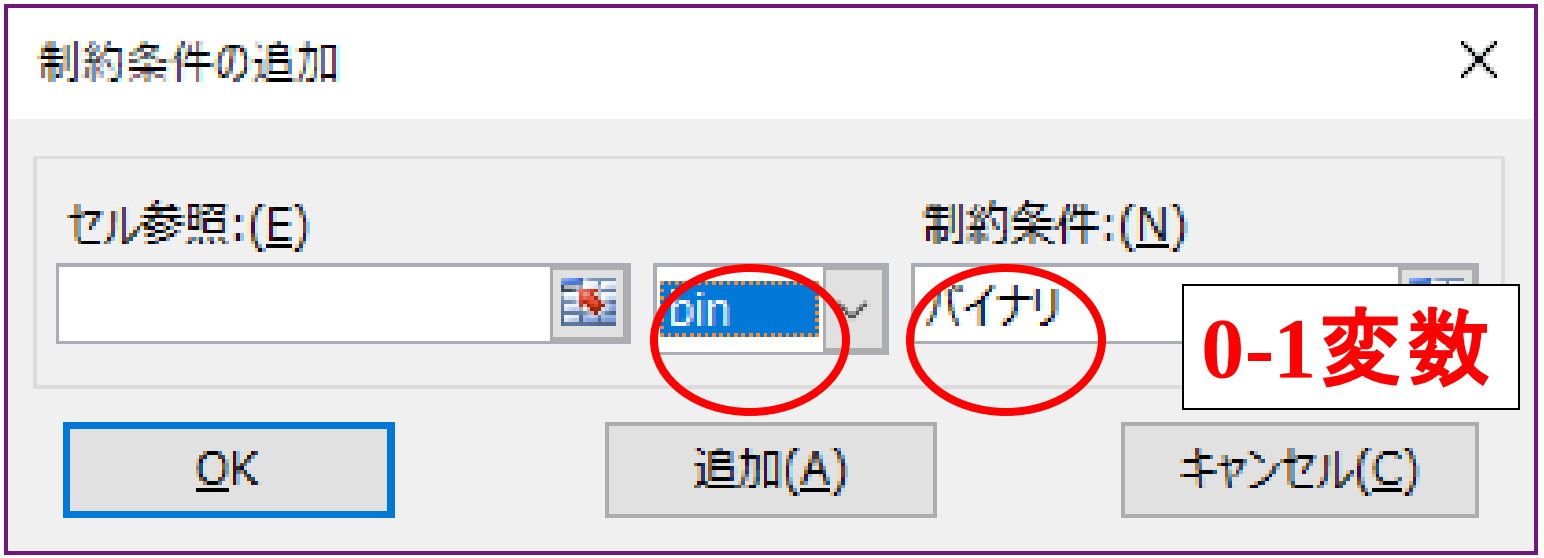
  等号　$Ax = 1$　（集合分割）
  （「ちょうど1回被覆」）
- 変数は0-1

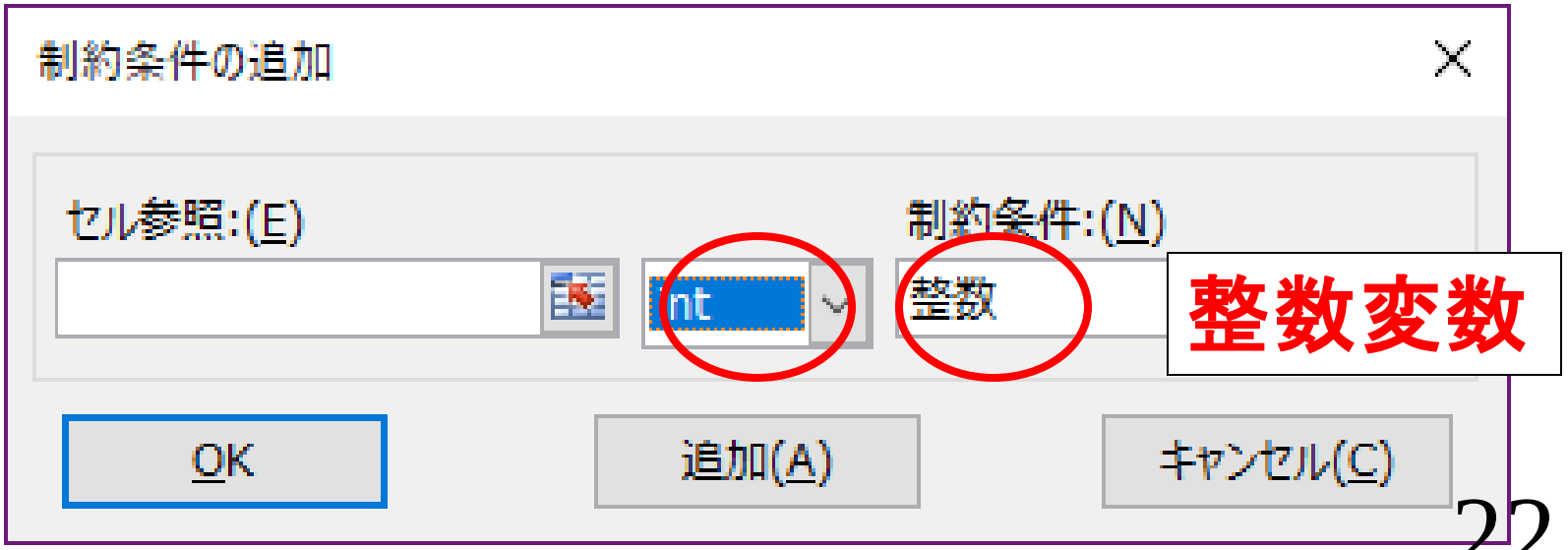
整数計画問題をソルバーで解くには

- 問題の設定は基本的には、線形計画問題の場合と同じ
- 「制約条件」の入力において、０－１条件や整数条件をかけたい変数に対して、（等号不等号の位置に）以下を設定：
  - ０－１変数の場合は、「データ」と設定
  - （一般の）整数変数の場合は、「区間」と設定

変数の０－１制約、整数制約指定

EXCELソルバーでは制約条件の一部として指定





整数計画問題(IP)と その線形緩和問題(LP)との関係

- 線形緩和問題の最適解を、四捨五入、切上げ、切下げなどにより整数値に丸めても、対応する整数計画問題の最適解が得られるとは限らない
- 線形計画問題の最適（目的関数）値は、常に対応する整数計画問題（最大化問題を想定）の最適値の上界値を与える（最大化問題の線形緩和問題は、元の問題の最適値の上界を与える）
- もし線形緩和問題の最適解がたまたま整数値をとるならば、その解は対応する整数計画問題の最適解でもあり、同時に両者の最適値が一致する

数理計画問題の解法（アルゴリズム）

- 「定式化」された問題は、適切な解法で解く
- パッケージを利用して解くか、自分で実装
- 解法の種類（数理計画法）
  - 厳密解法（最適を保証する解を求める解法）
  - 近似解法（最適は保証しないが、良い解を素早く求める解法）
- 整数計画：解析的には簡単そうに見えるが、難しい問題（最適解になりうる候補を列挙する手間）
- 非線形計画、大規模最適化、確率計画など

近似解法

巡回セールスマン問題（TSP)
Traveling Salesman Problem

- n個の点1,2,…,nと、任意の点間の「コスト」c_{ij}が所与であるとき、（たとえば）点1を出発して、各点を1回ずつ訪問して、点1に戻る最小コストの巡回路（Hamilton tour）を求めよ。（枝がなければ、$c_{ij}=\infty$）
- $c_{ij}=(\neq)c_{ji}$のとき　対称（非対称）型TSP

近似解法の種類

- **構築型 (construction-type)**
  - （TSPの場合）適当な部分巡回路、または、巡回路がない状態から順次巡回路を構築していく方法
  - （TSPの場合）挿入法（Nearest Insertion, Furthest Insertion, 他）、空間充填曲線法等
- **改善型 (improvement-type)**
  - (TSPの場合）すでに出来上がっている巡回路を改善する方法
  - （TSPの場合）広義の局所探索（近傍探索）

Nearest Neighbor法

- 適当な初期点から始める
- 現在いる点から、まだ訪問していない点の中でもっとも近い点に移動する
- すべての点を訪問し終えたら初期点に戻る

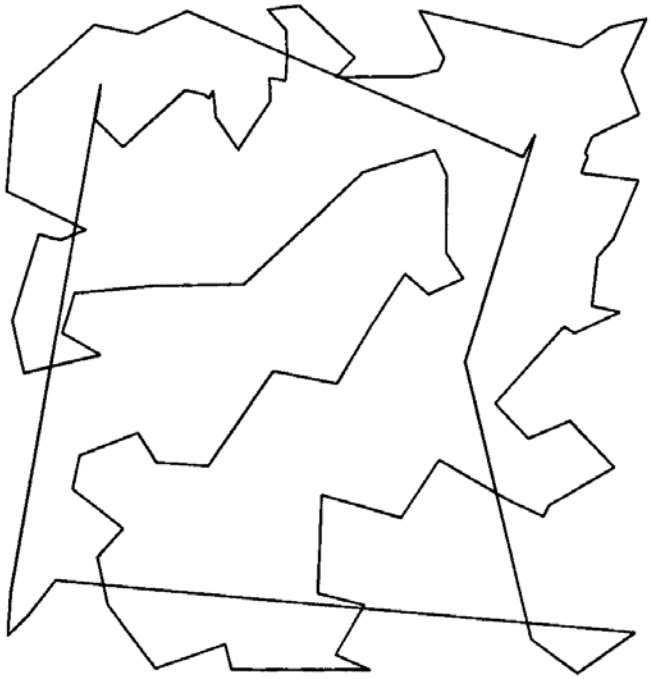
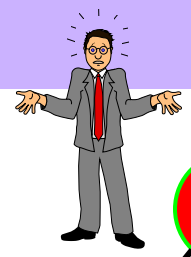


Figure 6.1 A nearest neighbor tour for rd100

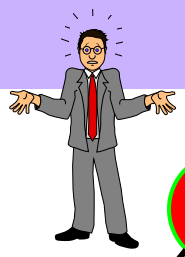
Nearest Neighbor法



距離(費用)行列

点1	99	58	57	18	53
	点2	42	52	88	30
		点3	30	35	45
			点4	95	46
				点5	85
					点6

Nearest　Neighbor法



距離（費用）行列

点１	99	58	57	18	53
	点２	42	52	88	30
		点３	30	35	45
			点４	95	46
				点５	85
					点６

Nearest Neighbor法

距離(費用)行列

点1	99	58	57	18	53
	点2	42	52	88	30
		点3	30	35	45
			点4	95	46
				点5	85
					点6

Nearest Neighbor法

距離(費用)行列

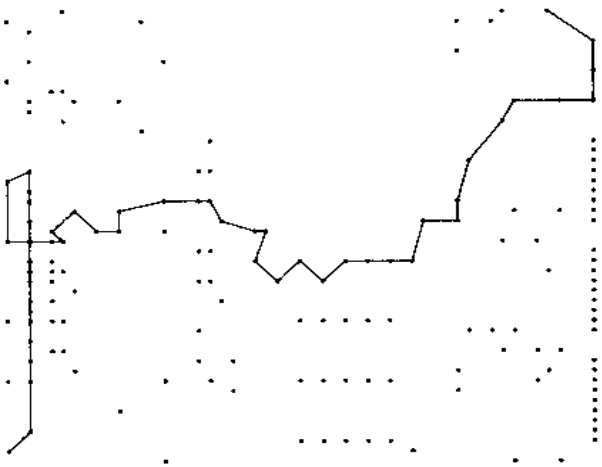
点1	99	58	57	18	53
	点2	42	52	88	30
		点3	30	35	45
			点4	95	46
				点5	85
					点6

Nearest Neighbor法

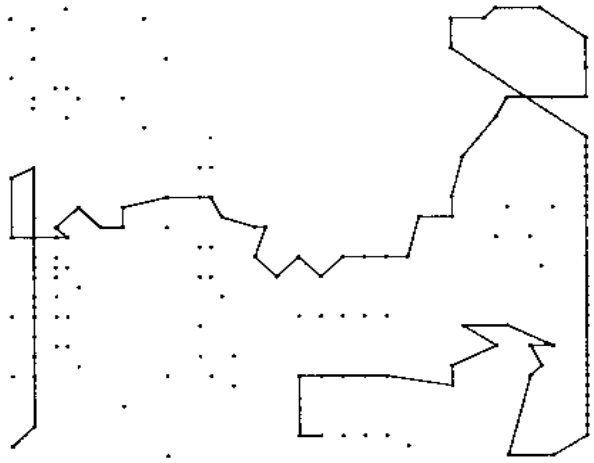
距離(費用)行列

点1	99	58	57	18	53
	点2	42	52	88	30
		点3	30	35	45
			点4	95	46
				点5	85
					点6

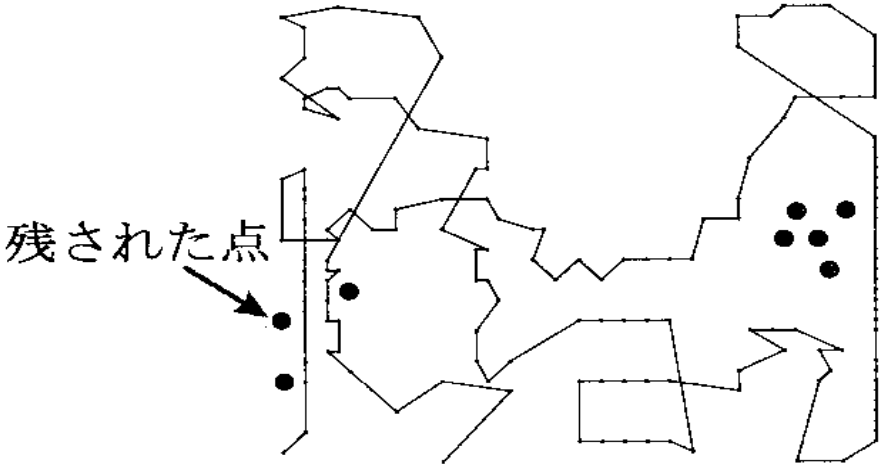
Nearest Neighborのスナップショット



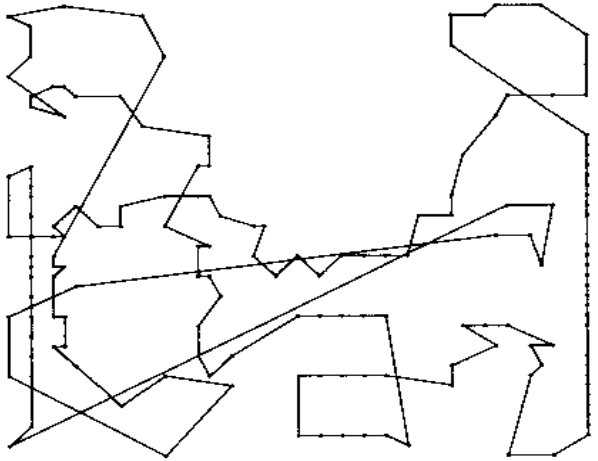
50イテレーション



100イテレーション

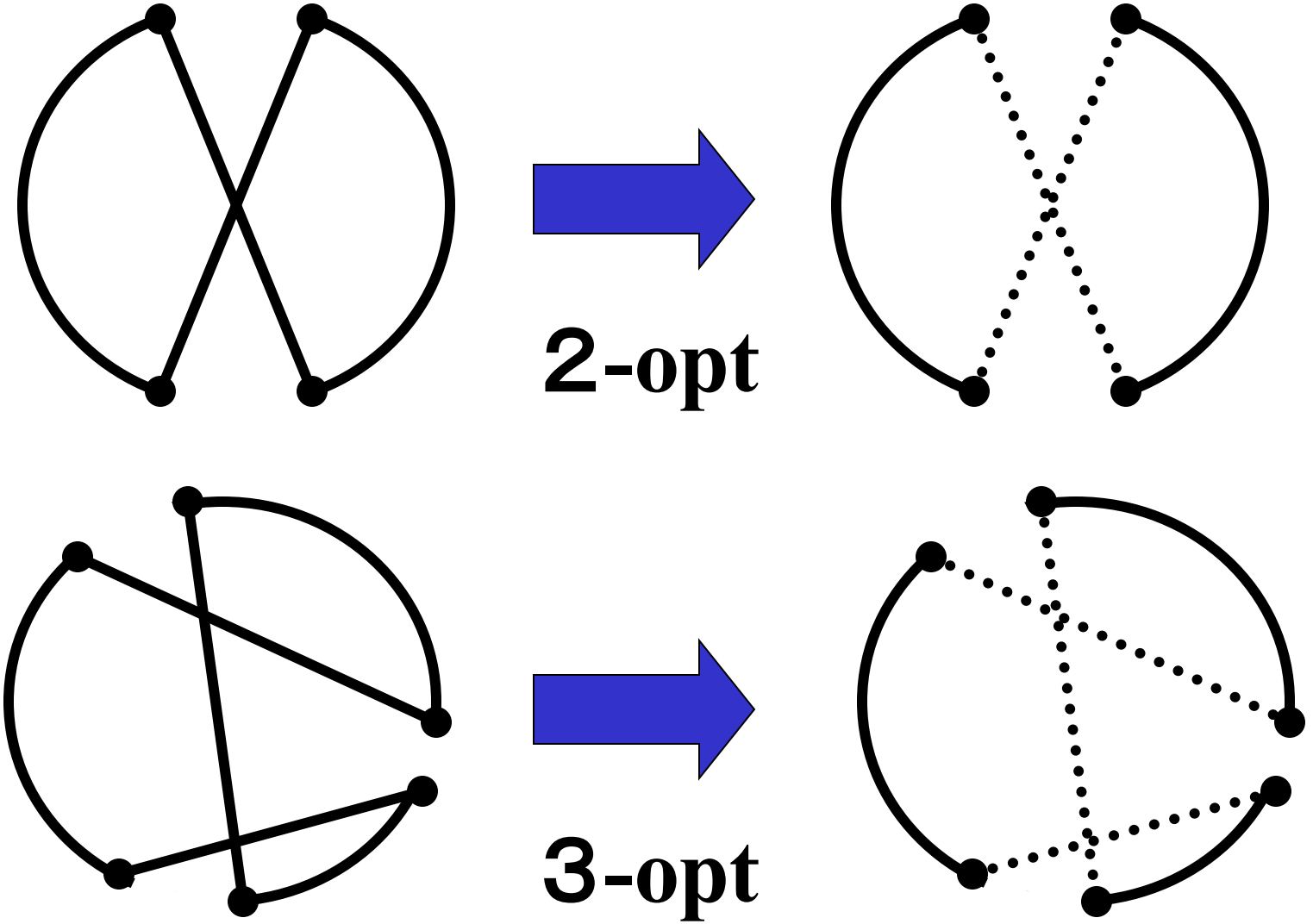


150イテレーション

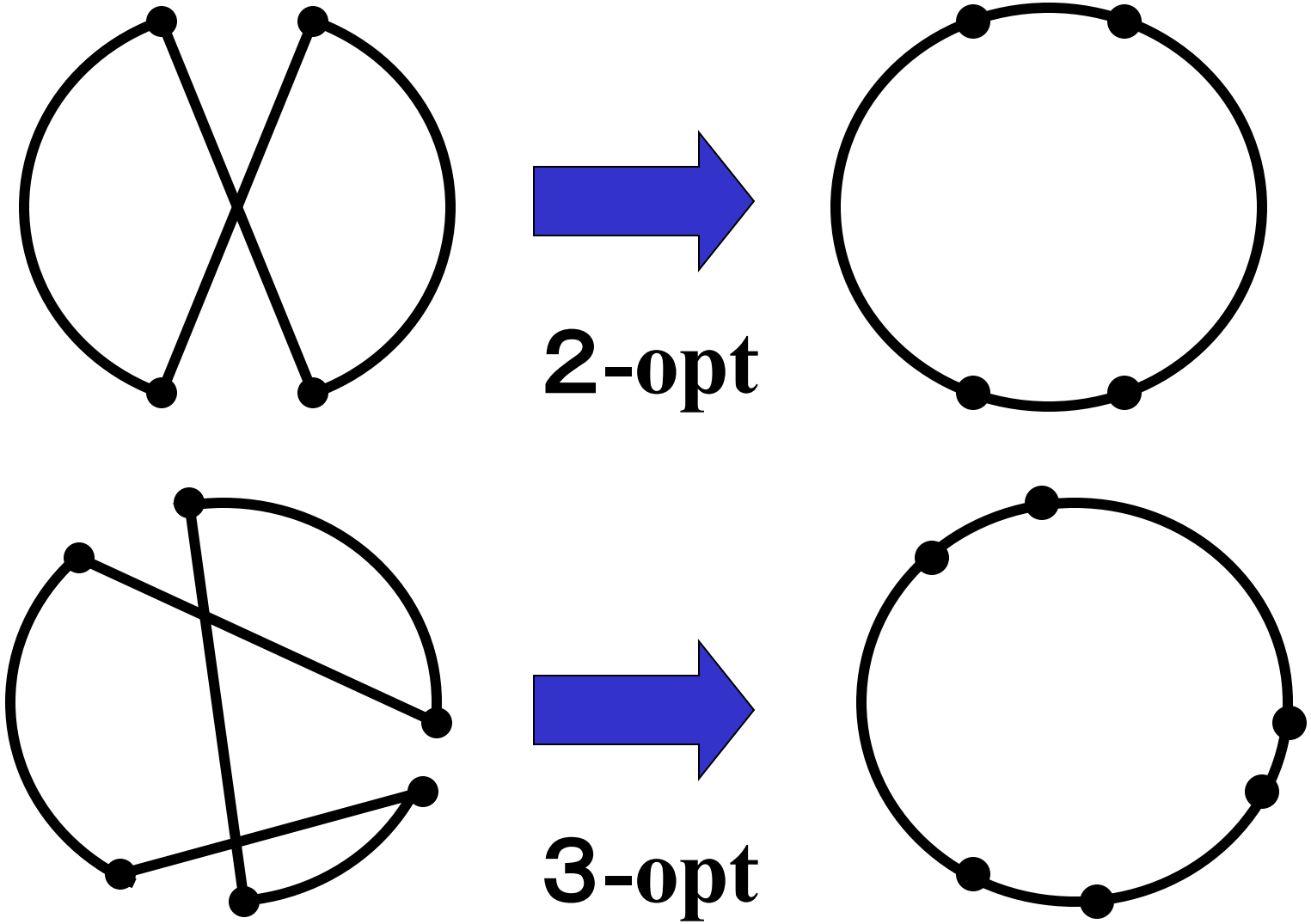


出来上がり

局所探索(2-opt, 3-opt)による巡回路の改善



局所探索(2-opt, 3-opt)による巡回路の改善



厳密解法

組合せ最適化の厳密解法

- 列挙法(enumerative algorithm)
- いかに「すべての」解を『列挙』するか
- すべての順列／組合せの列挙（Cでプログラムを書けるはず）: 計算時間が膨大!!
- すべての解を列挙することは、多くの実際問題において不可能 → 効率的な列挙法
  - 分枝限定法(branch and bound algorithm)
  - バックトラック法(backtrack algorithm)
- 動的計画法(dynamic programming)
- 元問題を部分問題に分解

ナップサック問題（Knapsack Problem）KP

（KP）最大化　$z = \Sigma^{n}_{j=1} c_j x_j$

制約　$\Sigma^{n}_{j=1} a_j x_j \leqq b$

$x_j \in \{0,1\}, \ \forall j$

より正確には，0-1 KP

0-1条件の無いナップサック問題はGeneral KP（GKP）.

例題

(KP)　最大化　$z = 7x_1 + 8x_2 + 3x_3$

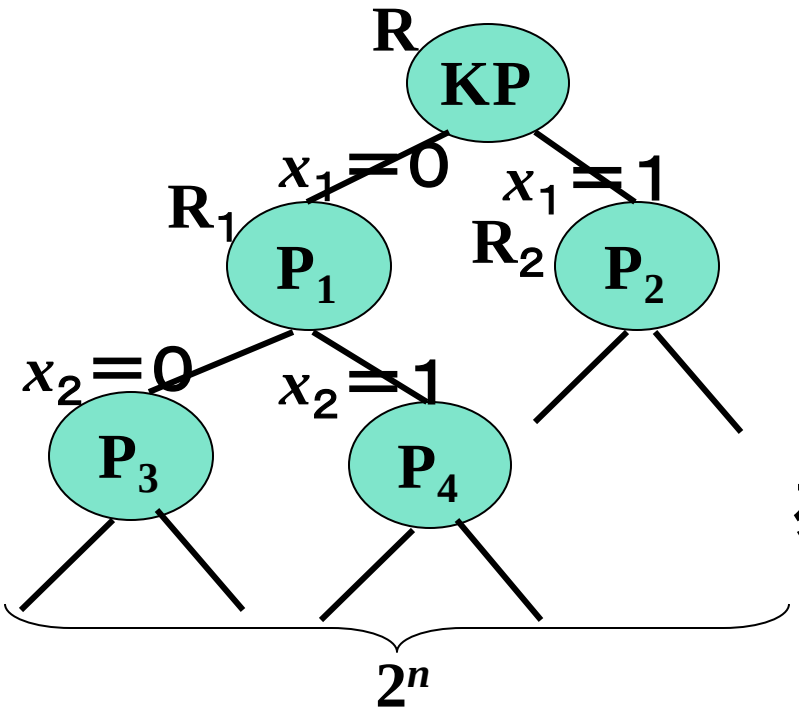
制約　$3x_1 + 4x_2 + 2x_3 \leqq 6$

$x_j \in \{0,1\}, \ \forall j$

分枝操作（branching operation）

ある問題（「親問題」）の実行可能領域をいくつかに分割して，
複数の「子問題」（subproblem）を生成する操作．

→分枝変数（branching variable）：分割に用いた変数



R=（KP）の可能領域
R_1＝（P_1）の可能領域
R_2＝（P_2）の可能領域

$R_1 \cup R_2 = R$
$R_1 \cap R_2 = \varphi$

列挙木（enumeration tree）

限定操作（bounding operation）

<最大化の場合>

ある子問題（P_k）に分枝操作を施す必要があるか否かを判定する，
または，ある子問題（P_k）の緩和問題（P'_k）から得られる上界z'_kが，
手元にある最良解の値z^0より大きいか否かを判定する操作．

上界値、下界値
upper/lower bounds

- z^*＝元の問題の最適目的関数値
- 最大化で考えると:
  上界値: $z^* \leqq z$ が保証されている値z
  （「緩和問題」の解は、z^*の上界を与える）
  下界値: $z^* \geqq z$ が保証されている値z
  （任意の可能解は、z^*の下界を与える）
- 上界値(下界値)は小さい(大きい)ほどよい
- 問題が最大化か、最小化かによって、上界値と下界値の役割が逆転することに注意。

緩和問題による「上界値」算出
（最大化問題を想定）

- **緩和問題**（relaxation）＝
  元の問題の条件を緩和した問題
- 代表的な緩和の方法
  （１）整数条件を緩和
  →線形緩和、連続緩和
  （２）制約条件を除去
  →ラグランジュ緩和
- 緩和問題は、原則として、元の問題より解きやすいことが望ましい

分枝限定法の考え方(1)
Branch and Bound Method

- 分枝(branching)===制限(restriction)
  変数値を制限(固定)する
  ===> 子問題(部分問題, subproblem)
- 限定(bounding)===緩和(relaxation)
  変数値を緩和する
  ===> 緩和問題(relaxation problem)
- 「制限」と「緩和」は、対立する概念

分枝限定法の考え方(2)
Branch and Bound Method

- 分枝操作により、変数値を固定し、元の問題を子問題に「場合分け」して考える
- 子問題そのものは依然として整数計画問題であるためLPに比べて解きにくい
- そこで、より解きやすい子問題の線形緩和問題を解き，子問題から手許にある最良解(暫定解)より良い解が得られるかを判定する(限定操作)

０－１ナップザック問題に対する分枝限定法

1)上界値の算出方法： **線形緩和（LP）**

2)下界値（暫定値）の算出方法： LP最適解の**切り下げ**

3)**分枝変数**の決定： LP最適解で**小数値をとる変数**（小数値をとる変数は高々1つしかない）

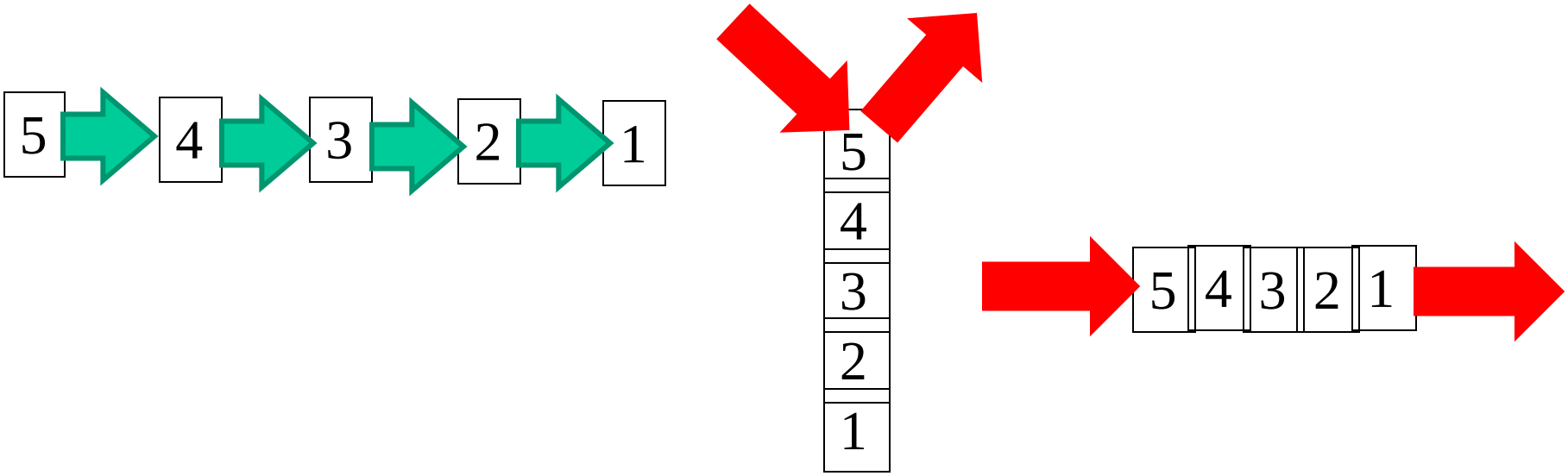
4)**分枝頂点**（分枝子問題）の選択：

深さ優先規則（問題番号大きいもの）

幅優先規則（問題番号小さいもの）

子問題選択：スタックとキュー

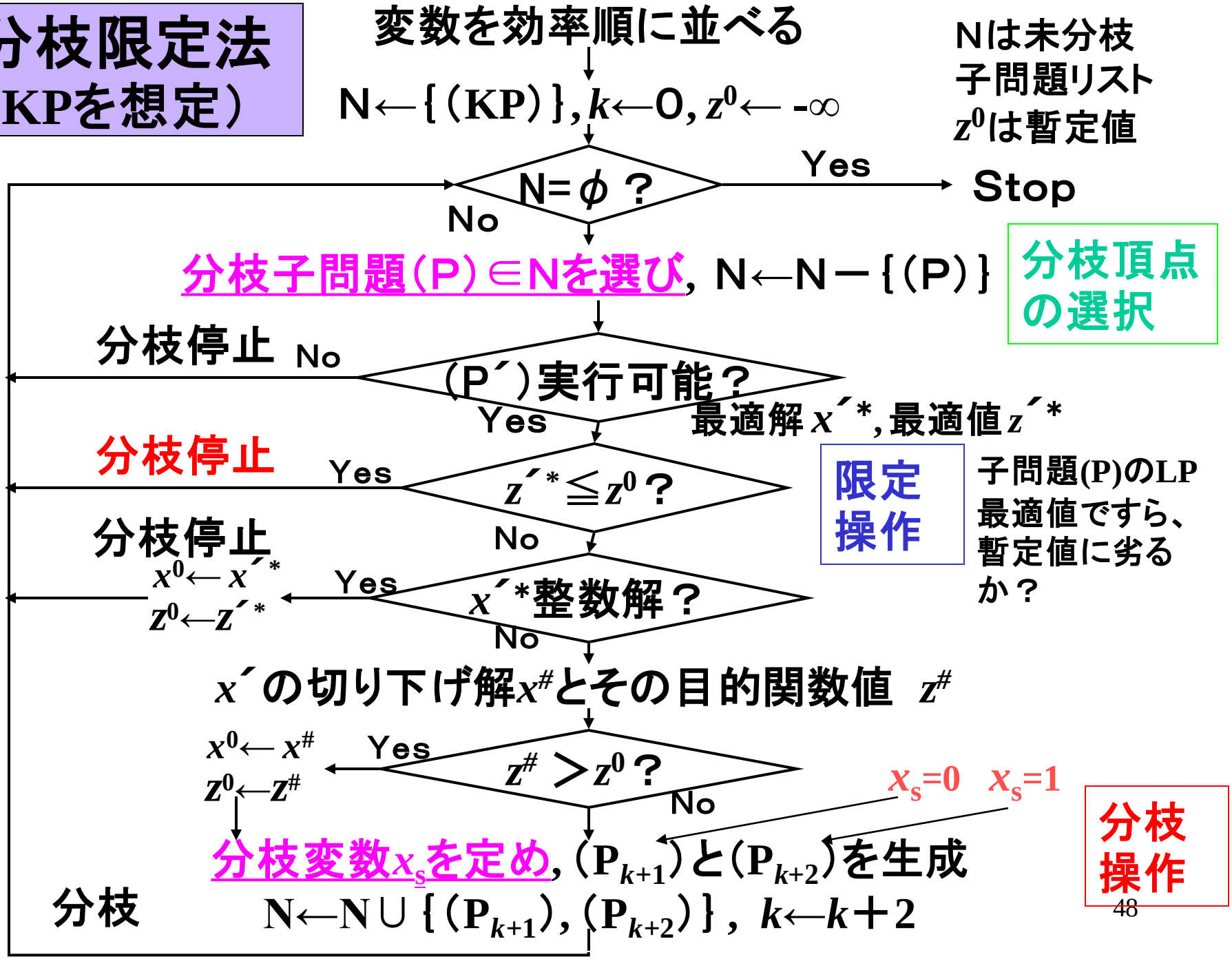
- 子問題1，2，3，4，5が番号順に生成
- スタック（先入後出）：奥行優先
  - 子問題を下から積み重ね、上から選択
- キュー（先入先出）：幅優先
  - 子問題を待ち行列の形で保持



分枝限定法の基本用語

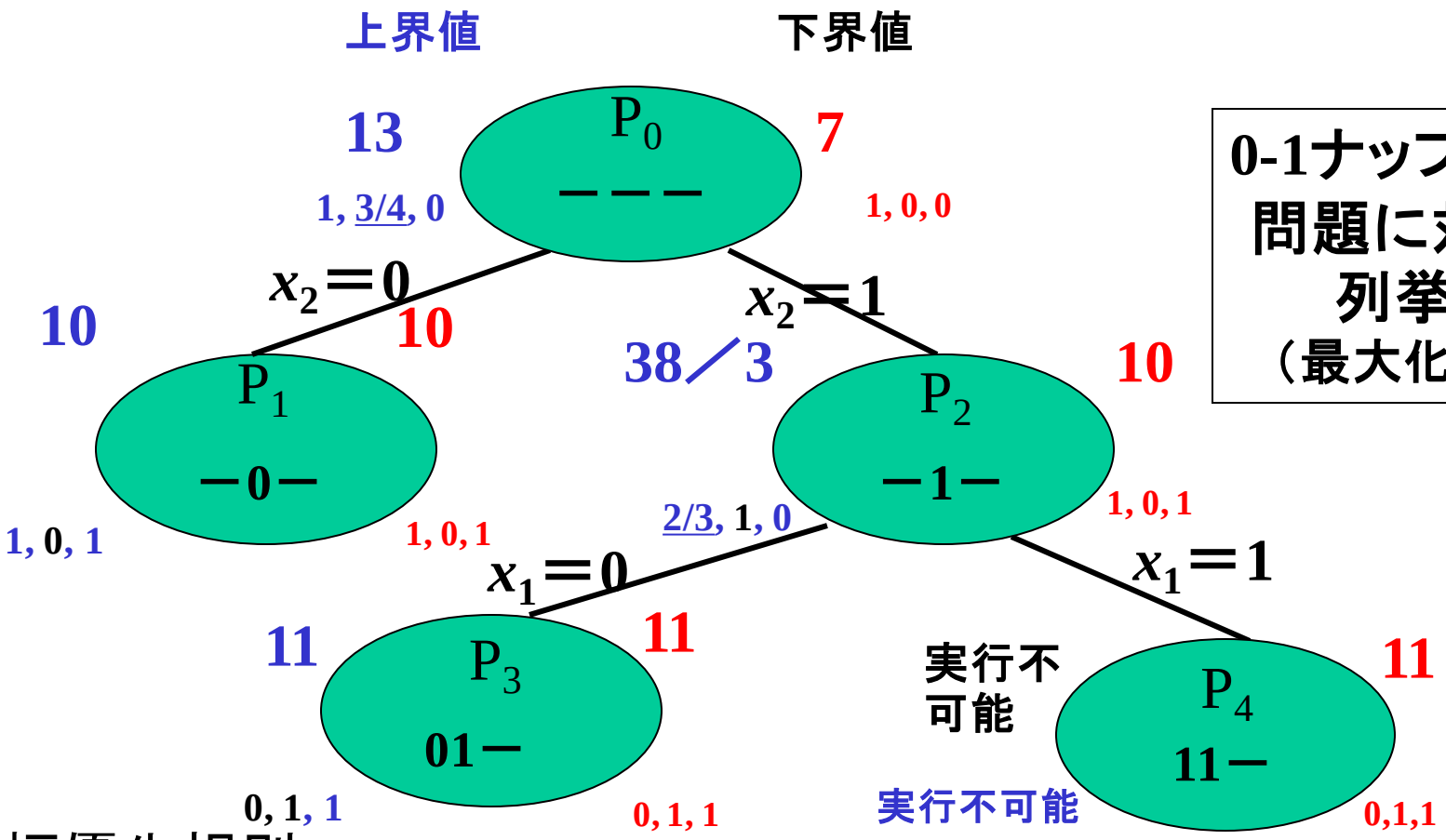
- 暫定(目的関数)値: これまでに見つかっている最良解の目的関数値
- 暫定解: これまでに見つかっている最良解
- 下界値、上界値
- 分枝変数: 分枝の対象となる変数
- 分枝頂点: 分枝の対象となる頂点(子問題)
- 未分枝頂点: まだ探索が終わっていない頂点(子問題)

分枝限定法（KPを想定）


変数を効率順に並べる
N←{(KP)}, k←0, z^0← -∞
Nは未分枝子問題リスト
z^0は暫定値
N=φ？
Yes
Stop
No
分枝子問題(P)∈Nを選び, N←N－{(P)}
分枝頂点の選択
分枝停止
No
(P´)実行可能？
Yes
最適解x´*, 最適値z´*
分枝停止
Yes
z´*≦z^0？
限定操作
子問題(P)のLP最適値ですら、暫定値に劣るか？
分枝停止
No
x^0← x´*
z^0←z´*
Yes
x´*整数解？
No
x´の切り下げ解x^#とその目的関数値 z^#
x^0← x^#
z^0←z^#
Yes
z^# ＞z^0？
No
x_s=0 x_s=1
分枝変数x_sを定め, (P_{k+1})と(P_{k+2})を生成
分枝
N←N∪{(P_{k+1}), (P_{k+2})}, k←k＋2
分枝操作

最大化　$z = 7x_1 + 8x_2 + 3x_3$

制約　$3x_1 + 4x_2 + 2x_3 \leqq 6,\ x_j \in \{0,1\},\ \forall j$



0-1ナップザック問題に対する列挙木（最大化問題）

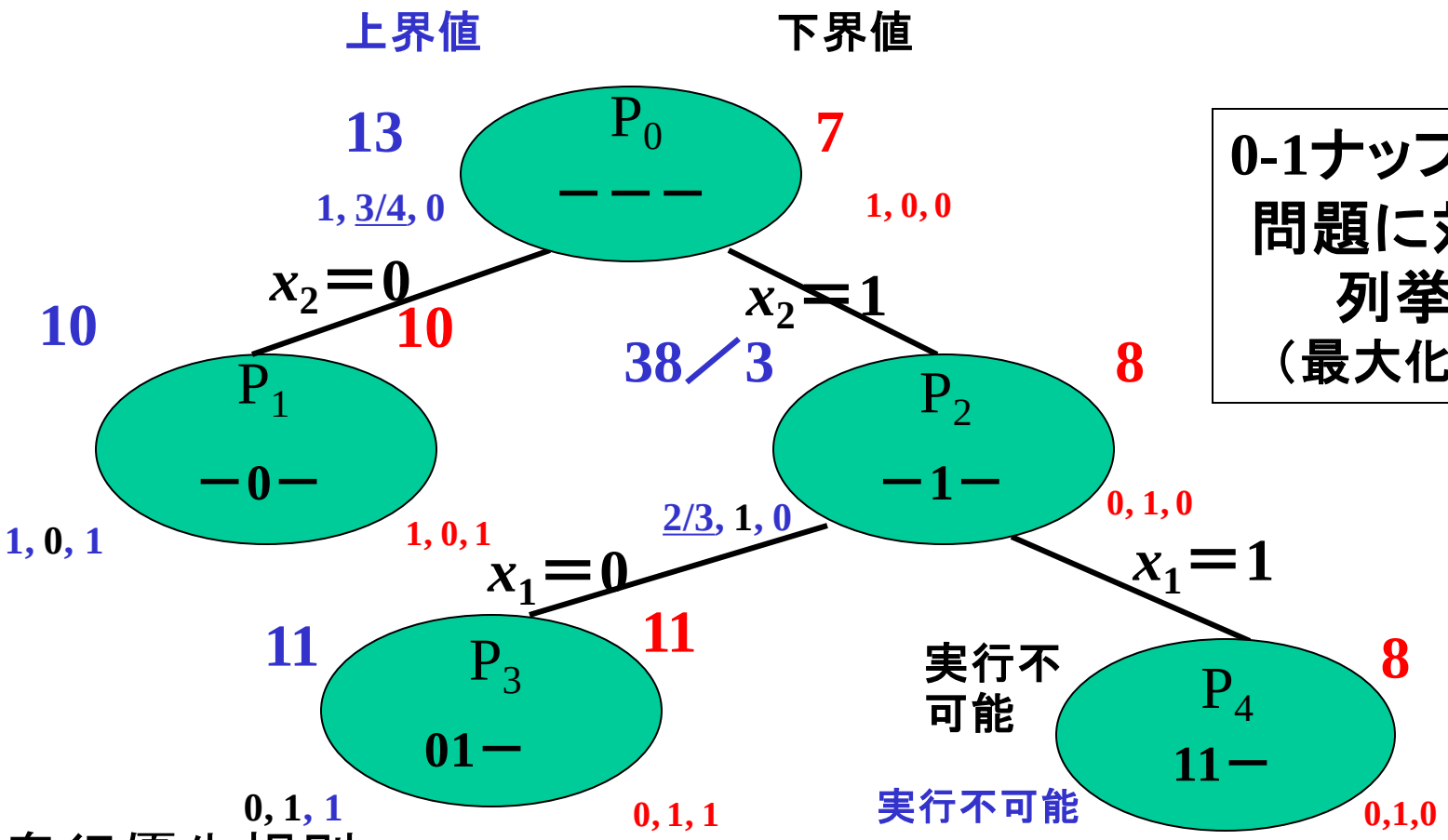
幅優先規則

N=　$\{(KP)\} \rightarrow \{P_1, P_2\} \rightarrow \{P_2\} \rightarrow \{P_3, P_4\} \rightarrow \{P_4\}$

解いた問題：　$(KP) \rightarrow P_1 \quad \rightarrow P_2 \quad \rightarrow P_3 \quad \rightarrow P_4$

最大化　$z = 7x_1 + 8x_2 + 3x_3$

制約　$3x_1 + 4x_2 + 2x_3 \leqq 6,\ x_j \in \{0,1\},\ \forall j$



0-1ナップザック問題に対する列挙木（最大化問題）

奥行優先規則

N=　$\{(KP)\} \rightarrow \{P_1, P_2\} \rightarrow \{P_1, P_3, P_4\} \rightarrow \{P_1, P_3\} \rightarrow \{P_1\}$

解いた問題：　$(KP) \rightarrow P_2 \rightarrow P_4 \rightarrow P_3 \rightarrow P_1$

分枝限定法設計のポイント

- 分枝頂点の選択
  - 下界値優先則
  - 奥行優先則
  - 幅優先則
- 分枝変数の選択
  - 分枝変数の値を制限（固定）することによってなるべく目的関数値が悪化するような変数を選択

ナップサック問題に対する動的計画法

（KP）最大化　$z=\Sigma^{n}_{j=1}\ c_j\,x_j$

制約　$\Sigma^{n}_{j=1}\ a_j\,x_j \leqq b$

$x_j\in\{0,1\},\ \forall\ j$

$f_k(\lambda)=\max\ \{\Sigma^{k}_{j=1}\ c_j\,x_j \mid \Sigma^{k}_{j=1}\ a_j\,x_j \leqq \lambda,$

$x_j\in\{0,1\},\ \ j=1,\ldots,k\}$

$f_k(\lambda)$: n個の品物のうち、$j=1,\ldots,k$ 個までの 品物を容量λのナップサックに入れる問題の最適目的関数値

最終的に $f_n(b)$を求めればよい

漸化式　$f_k(\lambda)=\max\{\ f_{k-1}(\lambda),\ c_k+f_{k-1}(\lambda-a_k)\}$

$=\max\{$品物kを入れない利益，品物kを入れる利益$\}$

- $f_k(\lambda)$を定義するナップサック問題の解においては、品物kが入るかまたは入らないということを考える
- 品物kが入らない場合：容量λのナップサックに$j=1,\ldots,k$-1個までの品物を入れるときの利益
- 品物kが入る場合：品物kを入れて、残り容量$\lambda-a_k$に$j=1,\ldots,k$-1個までの品物を入れるときの利益

ナップサック問題：例題

(KP) 最大化 $z = 7x_1 + 8x_2 + 3x_3$

制約 $3x_1 + 4x_2 + 2x_3 \leqq 6$

$x_j \in \{0,1\}, \ \forall j$

最適解を求めよ。ただし、記号を以下のように定義する。

$f_1(\lambda)=\max\{ 7x_1 \mid 3x_1 \leqq \lambda, \ x_j \in \{0,1\}, \ j=1\}$

$f_2(\lambda)=\max\{ 7x_1 + 8x_2 \mid 3x_1 + 4x_2 \leqq \lambda, \ x_j \in \{0,1\}, \ j=1,2\}$

$f_3(\lambda)=\max\{ 7x_1 + 8x_2 + 3x_3 \mid 3x_1 + 4x_2 + 2x_3 \leqq \lambda, \ x_j \in \{0,1\}, \ j=1,2,3\}$

$f_1(\lambda)$ の結果から$f_2(\lambda)$を求め、$f_2(\lambda)$の結果から$f_3(\lambda)$

$f_k(\lambda)$を計算する場合$f_{k-1}(\lambda)$は既に計算済

初期設定：$f_k(0)=0$ ($\lambda=0$のとき), $k=0,1,2,3$, $f_0(\lambda)=0$ ($\lambda>0$のとき),

f_k(負の数)$=-\infty$ ($\lambda<0$のとき) , $k=0,1,2,3$

実行不可能（容量オーバー）な場合を除外するため

ナップサック問題：解答例

(KP) 最大化　$z = 7x_1 + 8x_2 + 3x_3$

制約　$3x_1 + 4x_2 + 2x_3 \leqq 6,\ x_j \in \{0,1\},\ \forall j$

$f_k(\lambda) = \max\{ f_{k-1}(\lambda),\ c_k + f_{k-1}(\lambda - a_k) \}$

$k=1$のとき

$f_1(1) = \max\{\quad ,\quad \} = \max\{\ ,\ \} = \quad ,$

$f_1(2) = \max\{\quad ,\quad \} = \max\{\ ,\ \} = \quad ,$

$f_1(3) = \max\{\quad ,\quad \} = \max\{\ ,\ \} = \quad ,$

$f_1(4) = \max\{\quad ,\quad \} = \max\{\ ,\ \} = \quad ,$

$f_1(5) = \max\{\quad ,\quad \} = \max\{\ ,\ \} = \quad ,$

$f_1(6) = \max\{\quad ,\quad \} = \max\{\ ,\ \} = \quad ,$

最終的に、$f_n(b)$を求めたい

次の順に計算

$f_1(1) \Rightarrow f_1(2) \Rightarrow \cdots \Rightarrow f_1(b)$

$f_2(1) \Rightarrow f_2(2) \Rightarrow \cdots \Rightarrow f_2(b)$

$\cdots$

$f_n(1) \Rightarrow f_n(2) \Rightarrow \cdots \Rightarrow f_n(b)$

$k=2$のとき

$f_2(1) = \max\{\quad ,\quad \} = \quad ,$

$f_2(2) = \max\{\quad ,\quad \} = \quad ,$

$f_2(3) = \max\{\quad ,\quad \} = \quad ,$

$f_2(4) = \max\{\quad ,\quad \} = \quad ,$

$f_2(5) = \max\{\quad ,\quad \} = \quad ,$

$f_2(6) = \max\{\quad ,\quad \} = \quad ,$

$k=3$のとき

$f_3(1) = \max\{\quad ,\quad \} = \quad ,$

$f_3(2) = \max\{\quad ,\quad \} = \quad ,$

$f_3(3) = \max\{\quad ,\quad \} = \quad ,$

$f_3(4) = \max\{\quad ,\quad \} = \quad ,$

$f_3(5) = \max\{\quad ,\quad \} = \quad ,$

$f_3(6) = \max\{\quad ,\quad \} = \quad ,$

k λ	1	2	3	4	5	6
$f_1(\lambda)$						
x_1						
$f_2(\lambda)$						
x_2						
$f_3(\lambda)$						
x_3						

施設配置問題

関東地方を中心に営業を行ってきた輸入品販売業のK社では、関西地方に活動を拡大するため、京阪神地方に倉庫の賃借を行う計画を立てている。賃借の候補となる倉庫はmカ所にあって、第i地点の倉庫W_i (i=1, ･･･,m)の月間処理能力はa_i(トン／月)で、その経費(賃借料や維持費など毎月の固定費)はd_i(千円／月)である。また、関西一円に広がる消費地D_j(j=1, ･･･,n)での輸入品の需要量b_j(トン／月)と、W_iからD_jへのトン当たり輸送費c_{ij}(千円)が与えられたとして、すべての需要を満たし、毎月の総費用(倉庫経費＋輸送費)を最小にする倉庫配置と輸送計画を求めたい。この問題を数理計画問題として定式化せよ。

施設配置問題

- 定数：d_i＝倉庫iの経費（固定費），a_i＝倉庫iの容量
  c_{ij}＝倉庫iからjへの輸送単価，b_j＝需要地jの需要量
- 変数：y_i ＝ 倉庫iを借りるとき1，さもなくば0
  x_{ij} ＝ 倉庫iから需要地jへの輸送量（$x_{ij} \geqq 0$）
- 目的関数（総費用最小）：
  最小化 $z = \Sigma_i d_i y_i + \Sigma_i \Sigma_j c_{ij} x_{ij}$
- 制約条件：
  制約 $\Sigma_{j=1,\ldots,n} x_{ij} \leqq a_i y_i \quad i=1,2,\ldots,m$
  $\Sigma_{i=1,\ldots,m} x_{ij} \geqq b_j \quad j=1,2,\ldots,n$
  $x_{ij} \geqq 0,\ y_i = 0 \text{ or } 1$

「基礎OR」（前半）のまとめ

- 線形計画問題
  - 定式化とソルバーによる解の算出、結果の読み方
  - 単体法：（可能）基底解、有限収束、退化、巡回、初期可能基底解の求め方
  - 双対問題、双対定理、相補性定理
- 包絡分析法（DEA）
- ネットワーク計画問題
  - 最短路、最大流、最小費用流、最小木
  - 輸送問題、飛び石法
- 整数計画問題
  - 定式化とソルバーによる解の算出
  - 分枝限定法

オペレーションズ・リサーチによる問題解決

現実問題

ロジスティクス
（輸配送、在庫立地等）

スケジューリング
（順序づけ、資源配分、鉄道等）

生産
（循環型社会の生産)

数理モデル

最適化モデル
（数理計画モデル）

とくに
組合せ最適化
（巡回セールスマン、ナップザック、施設配置、」…）
シミュレーション

数理解法

最適解法
（分枝限定法、切除平面法、列生成法、ラグランジュ緩和法）
メタ解法
（タブー探索、アニーリング、…）
離散型シミュレーション

数理モデル：応用問題

- （数理）モデルを使って．．．

  システムの「構造」「本質」を解明

  システムを最適に設計する

  システムを最適に動かす

解法の高度化＋新たなモデル

- （数理）モデルに対して．．．

  解法の高度化、効率化

  新たなモデルの提案

数理計画法についてもう少し知りたい人は

- 確率計画法 (椎名孝之)

応用最適化シリーズ 5, 朝倉書店

A5／180ページ／ISBN978-4-254-11790-5

不確実要素を直接モデルに組み入れた本最適化手法について，理論から適用までを平易に解説した初の成書。〔内容〕一般定式化／確率的制約問題／多段階確率計画問題／モンテカルロ法を用いた確率計画法／リスクを考慮した確率計画法／他