12月1日

●数理計画と最適化(後半)

- 金曜日　　１限　９：００～１０：００　工3号館32号講義室
  　　２限１０：１５～１１：４５　情報基盤センター
  - 担当：横井浩史（精密機械工学専攻助教授）
  - 連絡先：hyokoi@robot.t.u-tokyo.ac.jp
  - Office:工14号館826室，内線28549

- 全体のスケジュール
  - 概論　　12月1日
  - 組み合わせ問題　　12月15日，22日，
    - ナップザック問題，欲張り法，けちけち法，分岐限定法
  - グラフとネットワーク　1月12日
    - ツリー探索，最短路問題，最大流量問題
  - ゲーム理論　　1月19日．1月26日
    - 純粋戦略，混合戦略，ナッシュ均衡
  - **試験日2月2日**

    ２月２日**レポート提出期限**

本日の予定

- 講義：概要説明
- 自己紹介と今後のスケジュール
- 演習：情報基盤センターへ移動
  - メールで履修希望を送付

組み合わせ最適化問題？

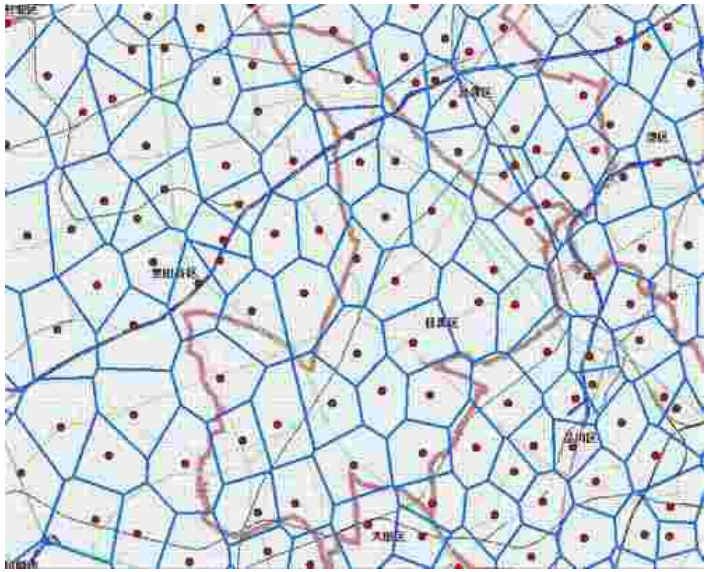
- 組合せ最適化問題は与えられた資源の組の中から，最もよく目的を達成する組み合わせを選ぶ問題.
  - 計画問題
  - 配置問題
  - 最短路問題

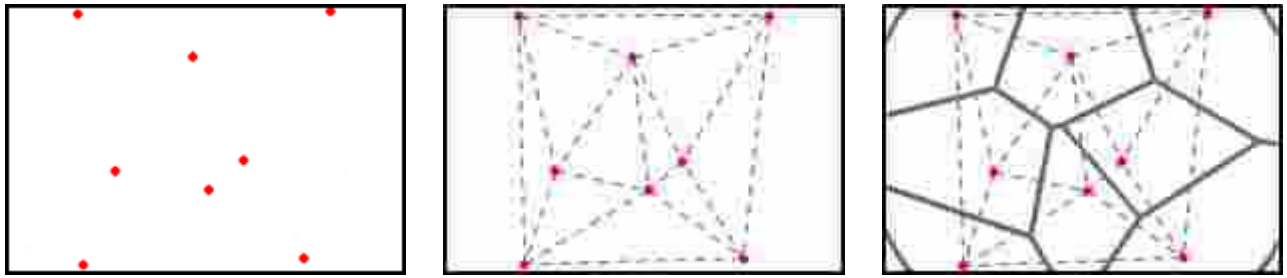
計画問題

- **離散生産・輸送計画**は、各工場での生産能力の上限と下限、各小売店での売上、各工場・小売店間の輸送コストがわかっているときに、どこの工場でどれくらい品物を作り、どこの小売店にどれだけ運べばコストがもっとも小さくなるか、という工場の生産量と輸送ルートを最適化する問題.
- **スケジューリング問題**、例えば、ある工場で、いくつか受注した仕事があり、それらの仕事をどの機械を使って誰がどの順番で行うかを、納期遅れが出ないように、残業が少なくなるように、などの目的関数を最小化するように決める問題（最適な工程表を作る問題）. その他、時間割や人の割り当てを決めたり、航空会社でのクルーの勤務表を作る問題.
- **配送計画**、物流会社などで、荷物の集積所から小売店に物品を配送、または回収するする際に、何台のトラックでどのようなルートを通って回収を行えば、トラックの台数・ドライバーの勤務時間・小売店への到着時刻などを最適化できるか、という問題.

配置問題

- 配置問題は，資源の分布を目的に応じて最適化する問題
- ボロノイ分割（ティーセン分割ともいう）とは、隣り合う母点間を結ぶ直線に垂直二等分線を引き、各母点の最近隣領域を分割したもの.
- 右図は、目黒区内にある郵便局（母点）の分布を元にボロノイ図形を描いた.



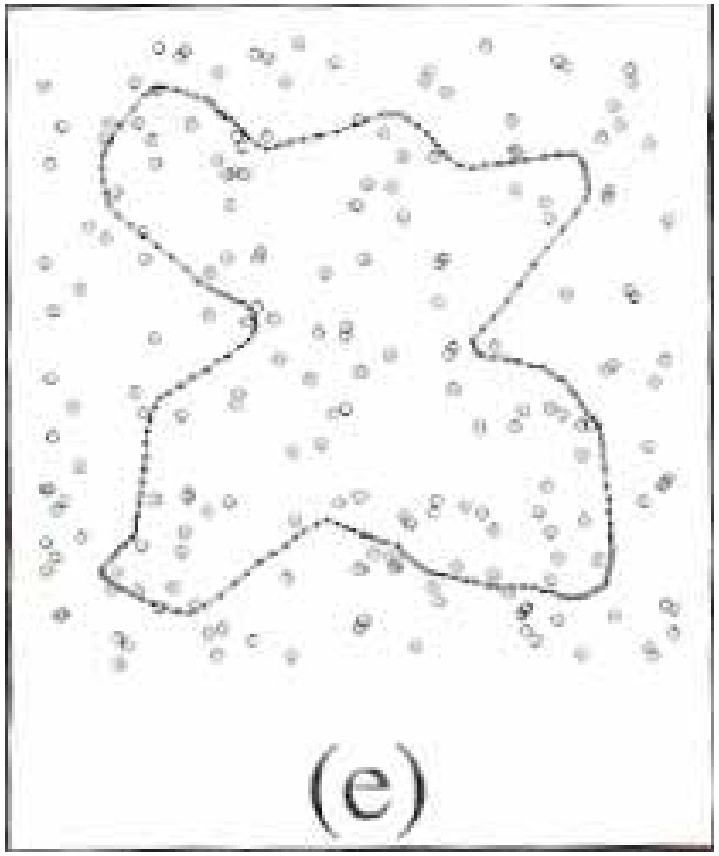


株式会社パスコ　GISシステムより

最短路問題：

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

- 多くの市と各市間の移動コストが与えられたとき、全ての市を一度だけ回って戻ってくるルートのうちコスト最小のものを求める問題
- 解法としては，（線形計画法と論理木を組み合わせた）分枝限定法や、（線形計画法と巡回群を組み合わせた）切除平面法により、約2000都市以内ならば普通のパーソナルコンピュータでも、およそ1日以内で厳密解を得られることが多い。



- その他，遺伝的アルゴリズム、タブ探索といったヒューリスティックを用いる.

自己紹介

- 研究紹介
  - 認知発達機械研究室(資料)
  - アメーバ状ロボット(資料)
  - 遊泳ロボット(資料)

ドキュメント　NHK総合　後9・00～10・15(75分)

NHKスペシャル
立花隆 最前線報告
「サイボーグ」が人類を変える」

サイボーグは悪魔の人体改造か

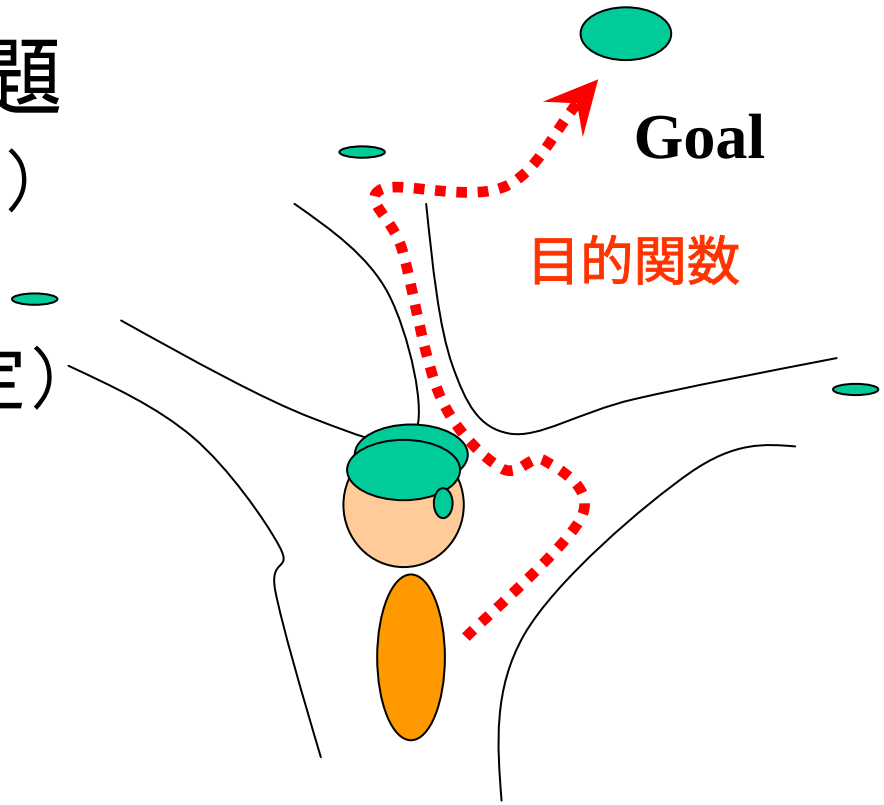
体の一部を機械に置き換え、脳が機械と直接つながった新たな存在・サイボーグ。最先端の現場では、脳の機能を機械に置き換えるサイボーグ化も進んでいる。難病の治療に劇的な効果をもたらしたが、うつ病や強迫神経症などにも応用され、人の心＝精神の調整にまで踏み込もうとしている。また、アメリカ国防総省では、不死身のサイボーグ兵士を作る計画も。世界各地の研究を取材し、科学の最前線を追ってきた立花隆がサイボーグ技術の行方を見つめる。

▲世界的にも珍しい「5本指の手」の実験に立ち会う立花

- 自己紹介
  - 自分の苦手とする分野を選ぶ傾向にある.
  - 偏っている(世間知らず).
  - 純愛志向(理想主義)
  - 争いが嫌い(平和主義)

一般の情報処理的問題

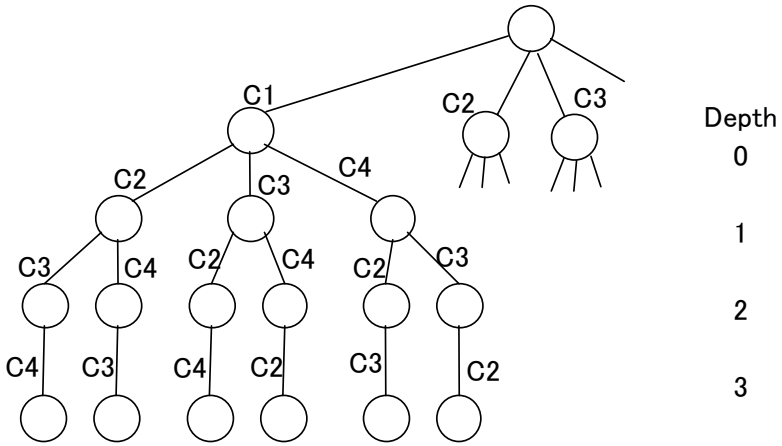
- 意思･戦略決定の問題
  - ルートの選択（駅ナビ）
  - 株式の選択
  - パズルなど（順序決定）



- 関数最適化
  - 補間（外挿と内挿）
  - 写像

選択⇒評価⇒探索

- 例えば，目的地までの最短経路を知りたい場合
- (C1,C2,C3,C4)の中から経由地を選択→経路長を評価
- 最短路を与える順列を探索



枚挙図

上三角行列＝下三角行列の場合
↓
幾何学的最短路問題

距離	C1	C2	C3	C4
C1	0			
C2		0		
C3			0	
C4				0

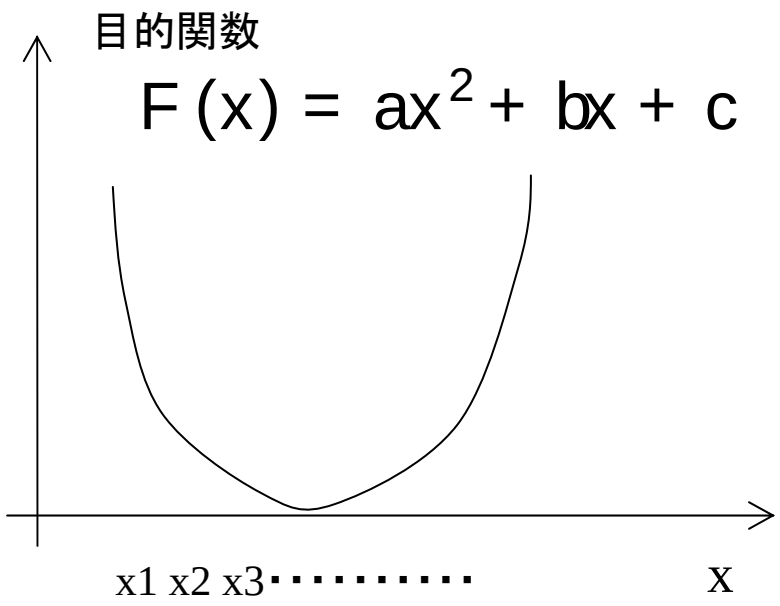
評価値を与える関数⇒目的関数

目的関数とは

- 最適化問題を解くための最重要項目
- 消費エネルギーや維持費などのコスト
  - 企業利益や貯蓄などに直結
  - 経営効率などの指標
- 効用関数，コスト関数，評価関数，適応度関数，Utility Functionなどと呼ばれることもある.

探索とは

- 最適化問題において，解を見つけるために行うオペレーションの総称.



解析的アプローチ

F=0 ⇒ $x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$

探索的アプローチ

F=0 ⇒ x1 x2 x3・・・・・・・・・・
F1 F2 F3・・・・・・・・・・

よりF=0に近い値を与えるxを探す

組み合わせ問題

(Combinatorial Optimization)

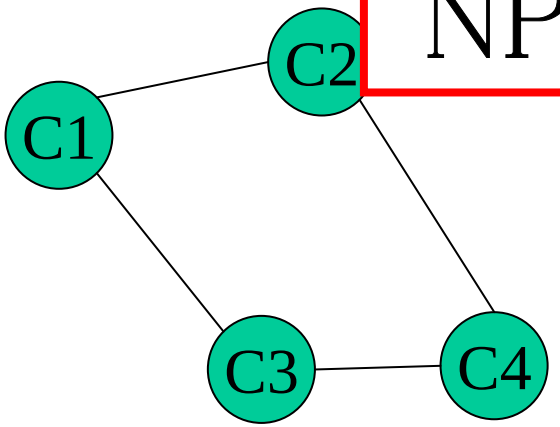
- 組み合わせ最適化問題とは，一般的な最適化問題における集合S およびXが，例えばn次元整数ベクトルや方策（要素列）の集合など，有限個あるいは加算無限個の要素を持つ離散集合である場合をいう.
  - 一般的な最適化問題
    - 目的関数:f(x)を最小(最大)にするxを
    - 制約条件:x∈S, ただしS⊂X のもとで決定せよ.
    - 許容領域S:集合Xの部分集合
    - x:Sの要素
  - 組み合わせ最適化問題の例
    - 巡回セールスマン問題(Traveling Salesman Problem)
    - 輸送問題
    - スケジューリング問題
    - 最適配置問題
    - ナップザック問題

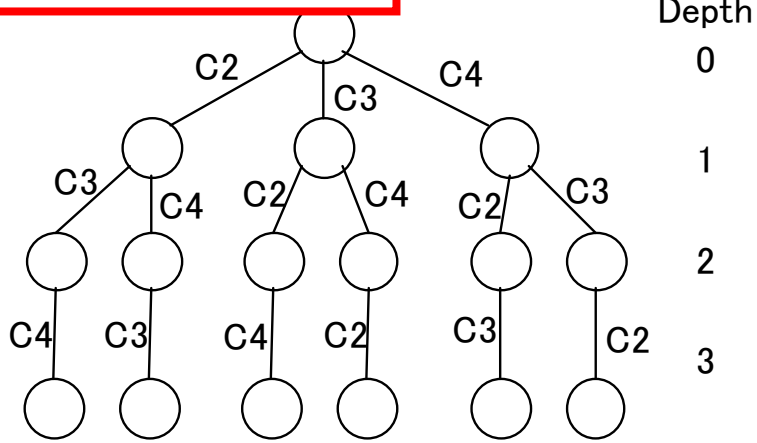
4都市TSP の例(Ci(i=1,2,3,4):都市)

最短距離で期間可能な一巡閉路を発見する問題

$$O(N^N) = (N-1)!/2$$

NP完全問題





NP完全問題

- 問題の規模が小さい組合せ最適化問題は,
- **枚挙図**をつくりその許容解の中で目的関数の値を最小(最大)にする解(最適解)を見つけ出すことが可能であるが,
- 問題の規模が大きくなると場合の数が天文学的な数となる. このことを**組合せ爆発**(Combinatorial explosion)とよび, そのような問題はNP完全問題とされている.

NP完全問題の例

- ハミルトン閉路問題:
  - あるグラフにおいて，全頂点を一度だけ通過し，始頂点から終頂点へと到達するような経路が存在を示す問題.
- 巡回セールスマン問題:
  - 複数の都市を一度だけ訪問し，出発点の都市に戻る巡回路のなかで最短経路になるコースを求める問題.
- ナップザック問題:
  - 相異なるサイズと価格が割り当てられているn個の荷物と，ある容量を持つナップザックが与えられた時，価格の和が最大となるようにナップザックに詰めこめる荷物を選ぶ問題.
- ジグソーパズル問題:
  - 複数種の多角形の組が与えられた時，これらを平面上に隙間なく敷き詰めることが可能かを問う問題.
- ブール式の充足可能性問題**(SAT)**:
  - あるブール式F(x1,x2,…xn)をTrueとするようなブール変数 x1,x2,…xn∈{True,False} の割り当ては存在するかを問う問題.
  - ただしFはブール変数，¬，∧，∨，(，)，からなる.

NP完全性の定義

- 問題Lは、Lが計算量のクラスNPに属し(L∈NP)、

かつ

- 他のどのNPに属する問題L'も、多項式時間限定変換機(チューリングマシン)によってLに還元可能であるとき(∀L'∈NP(L'≦L))、
- NP完全であると言う。」

「クラスNP」:
多項式時間限定の非決定性チューリングマシンによって解かれる問題(受理される言語)のクラス。

「L'は多項式時間限定変換機MによりLに還元可能⇔L'≦L via M」:
変換機Mとは"入力テープ(read onlyなテープ)"と"出力テープ(write onlyかつヘッドがその上を左に動くことができない)"をもつチューリングマシンで、入力記号列xに対する計算が停止したとき、出力テープ上には変換結果として文字列M(x)が書き出されている。言語L'の任意の語を入力xとするならば(x∈L')、その変換結果M(x)が言語Lに含まれている(M(x)∈L)ような変換機Mが存在するとき、L'はLに還元可能であると言う。要するに、問題L'を多項式時間で問題Lに変換するアルゴリズムが存在し、すでに存在するLを解くチューリングマシンを使ってL'の任意のインスタンスを解かせるときに、変換にかかる計算量を無視できるということである。たとえば、ハミルトン閉路問題は巡回セールスマン問題に多項式時間還元可能である。

守屋悦朗著 (1997) "情報処理シリーズB-2 チューリングマシンと計算量の理論" 培風館.

NP完全問題の利用例(暗号化)

原始的な暗号系:シーザー暗号

例: 原理は簡単でアルファベットをある数だけずらす

Crazy Sexy Cool

3文字ずらす

Fudcb Vhab Frro

解読されやすいのでスペースを削除, すべて大文字へ

FUDCBVHABFRRO

復号化: 3文字だけ元に戻す.

CRAZYSEXYCOOL

よく見ると, 単語の区切りは想像できる.

暗号を解く鍵は「3」となる. ---→簡単にわかってしまう.

NP完全問題の利用例
(公開鍵方式)

1976年：Diffe,Hellman（スタンフォード大学）により提案

公開鍵（暗号化鍵）責任ある公的な機関で公開し，公知の鍵
秘密鍵（復号化鍵）秘密鍵は自分自身しか知らない鍵

ネットワーク上の誰とでも，相手先の公開鍵を調べ，その鍵で暗号化したメッセージを送ることで即通信をすることができる．

しかし，シーザー暗号のような従来の方法では暗号化鍵が分かってしまうと復号化鍵もばれてしまうので安全性が低い．

そこで，暗号化する鍵が分かっても，復号化する鍵を解読することはほぼ不可能であるといった「一方通行」の関数が必要．
この「一方通行」の関数として使われるのが**素因数分解**

■　認証の必要性

公開鍵暗号系での認証の例.

登場人物:　A君 C君: 品物Tをインターネットで店Bに注文

暗号化して注文: 「Aだけど，品物Tをお願いします」A
(店Bが公開している公開鍵で暗号化して送信)
しかし，C君でも同じことができる？:「Aだけど，品物Tをお願いします」C
(店Bが公開している公開鍵で暗号化して送信)
●本当にA君が送ったものかどうか確認が必要. この確認のことを**認証**と呼ぶ.

■　認証の方法

A君は自分の秘密鍵を使って次の二つのメッセージを送ります。

1.「Aだけど，品物Tをお願いします」(店Bの公開鍵で鍵をかける)
2.「僕はAです」(A君の秘密鍵で鍵をかけ，さらに店Bの公開鍵でもう一度鍵をかける)

メッセージを受け取った店Bは店Bの秘密鍵でメッセージを開きます。すると

1.Aだけど，品物Tお願いします
2.A君の秘密鍵『僕はAです』

店BはA君の公開鍵を使って意味不明のメッセージを開封
「僕はAです」と書かれたメッセージ. **認証の成功**

鍵の作成

送信文→コード化→素数→公開鍵１　**2279**

CAT　　1371　　43　53　　2279＝43×53

↳素数→オイラー関数→公開鍵２　**1241**

43 53　ϕ（2279）＝(43-1)×(53-1)＝**2184**　と互いに素となる数

↳ユークリッドの互除法→秘密鍵 **1649**

2184 を法としたときの 1241 の逆元　1241×1649＝2046409＝1（mod2184）

暗号化

↳ 43 と 53 と 2184 は秘密

（公開鍵2）

（送信文）＝　暗号文　　Mod（公開鍵1）

1371^{1241} = 2003 mod(2279)

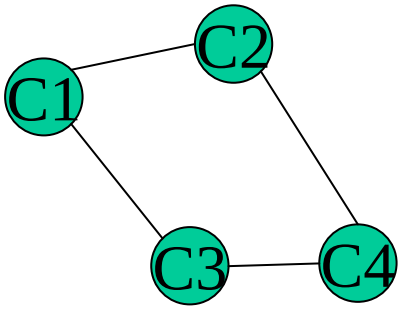
復号化

（秘密鍵）

暗号文　＝　送信文

2003^{1649} = $(1371^{1241})^{1649}$ mod(2279) = 1371

組み合わせ最適化問題

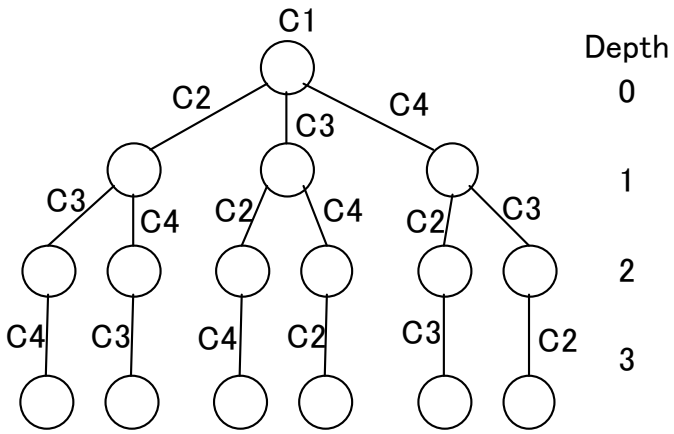


- 例）巡回セールスマン問題
- 都市の順列：C1→C2→C3→C4
  - xi-{ 1, 2, 3, 4, 1}
- 目的関数：巡回距離の最小化
  - **F**＝Σd(Cxi, Cxi+1),　i=1→4
- 制約条件：各都市を必ず一度は通り，スタート地点に戻ってくる.
  - G=Σa(xi) =4,　　xiを通過した場合にはa(xi)=1それ以外は0

列挙法(枚挙法)(Enumeration)

- 許容領域Sに属するは有限個の要素の中の，すべての許容解を列挙し，その中で目的関数f(x)を最小(最大)にする解(最適解)を見つける方法.

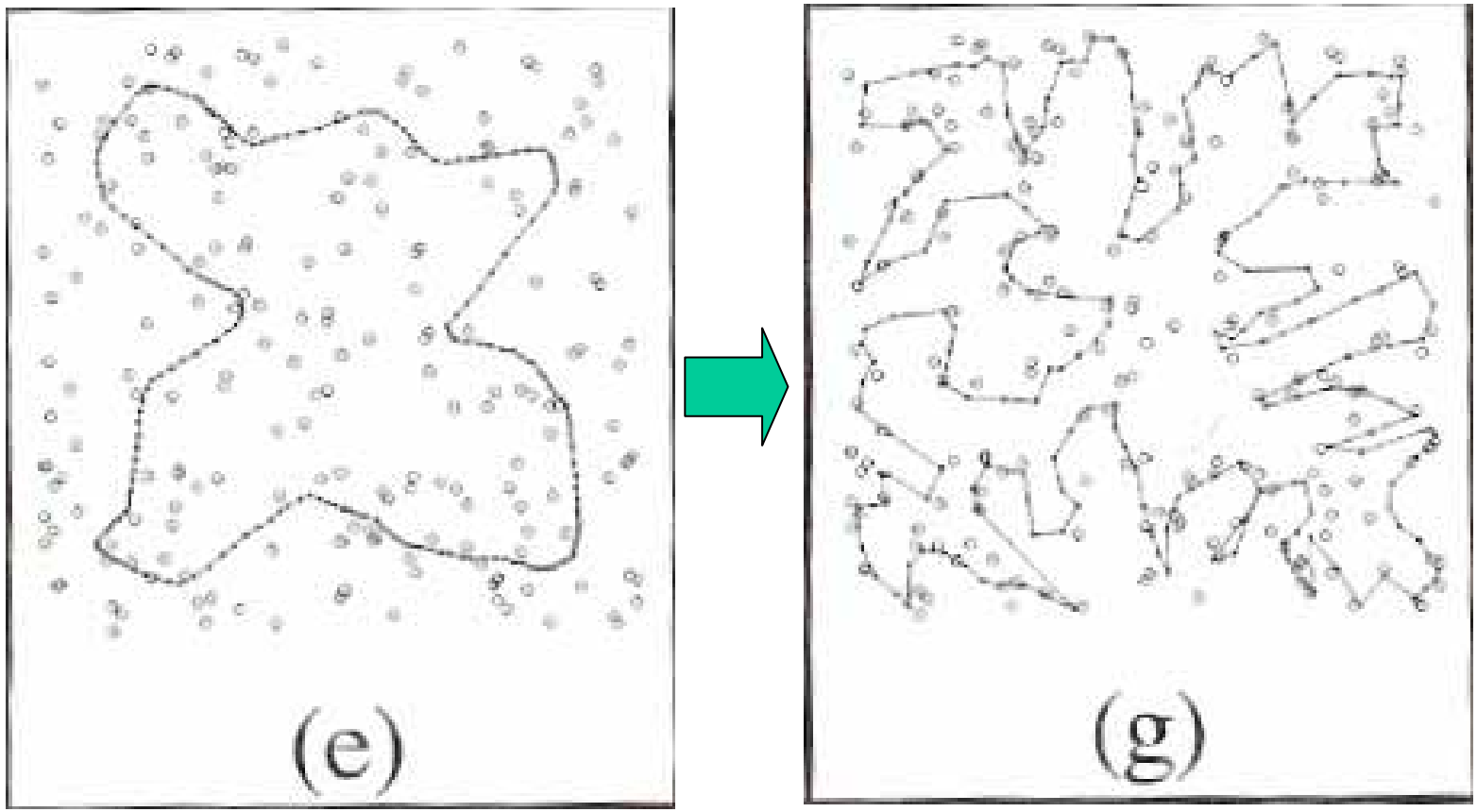
  - 解を列挙するために，
  - 木(Tree)を用いて表記し，
  - 図に表したものを列挙図(enumeration diagram)と呼ぶ.
  - 木は節点(node)と葉(leaf)から構成され，
  - C1を根(Root)，木の深さ(depth)，高さ(height).



探索法あれこれ

- モンテカルロ法(Montecaro Algorithm)
- 酔歩アルゴリズム(Drinker's Algorithm)
- 最近接点法(Nearest Neighbor Algorithm)
- 古典的最適化（欲張り法，ケチケチ法，分枝限定法，ディスパッチングルール，タブーサーチ）
- 弾性ネットモデル（Elastic Net)
- 進化的計算法(Evolutional Computing)
- ホップフィールド型ニューラルネット(Hopfield type NN)

例) 弾性ネットモデル(Elastic Net)



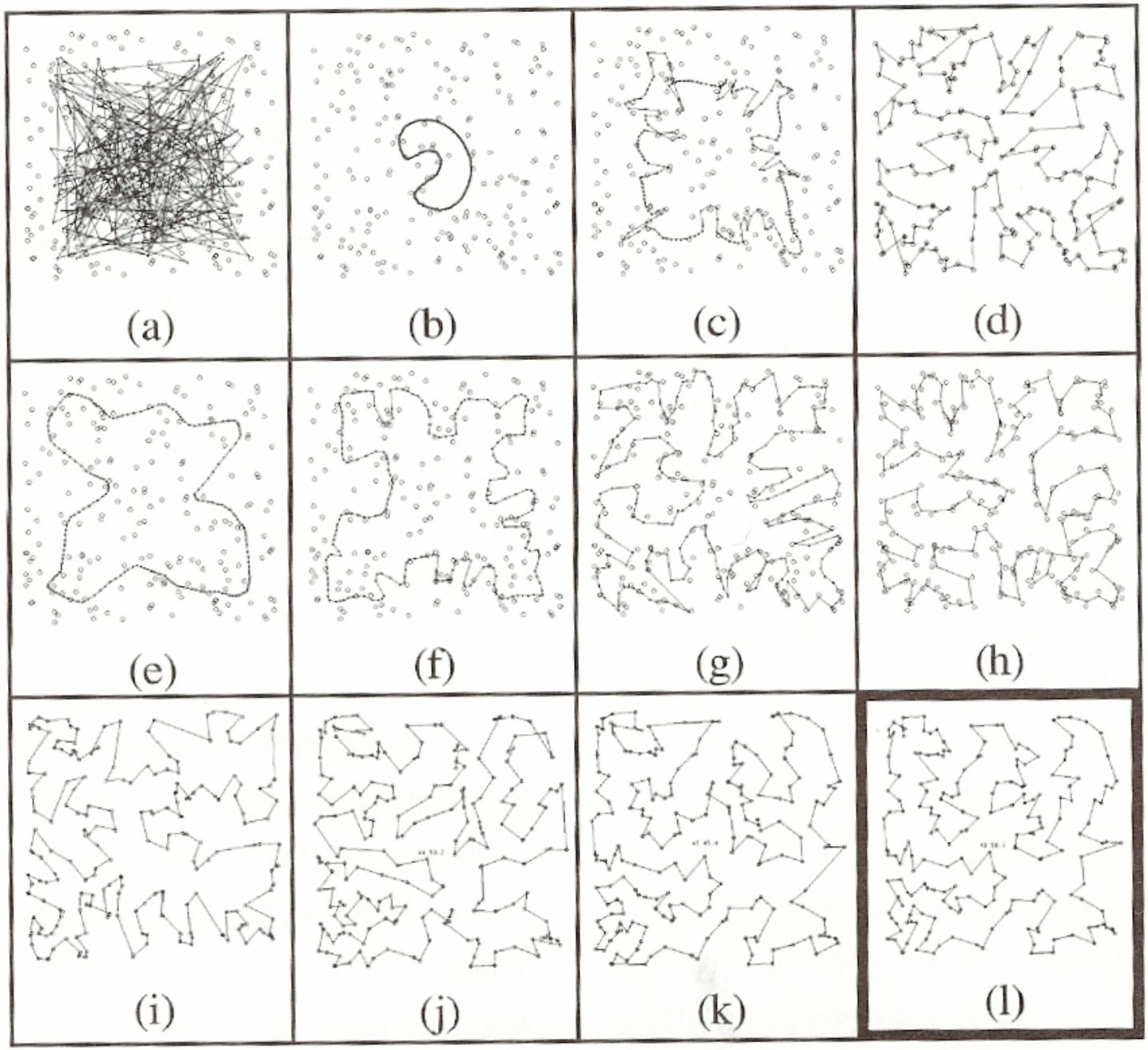


図 20　探索過程

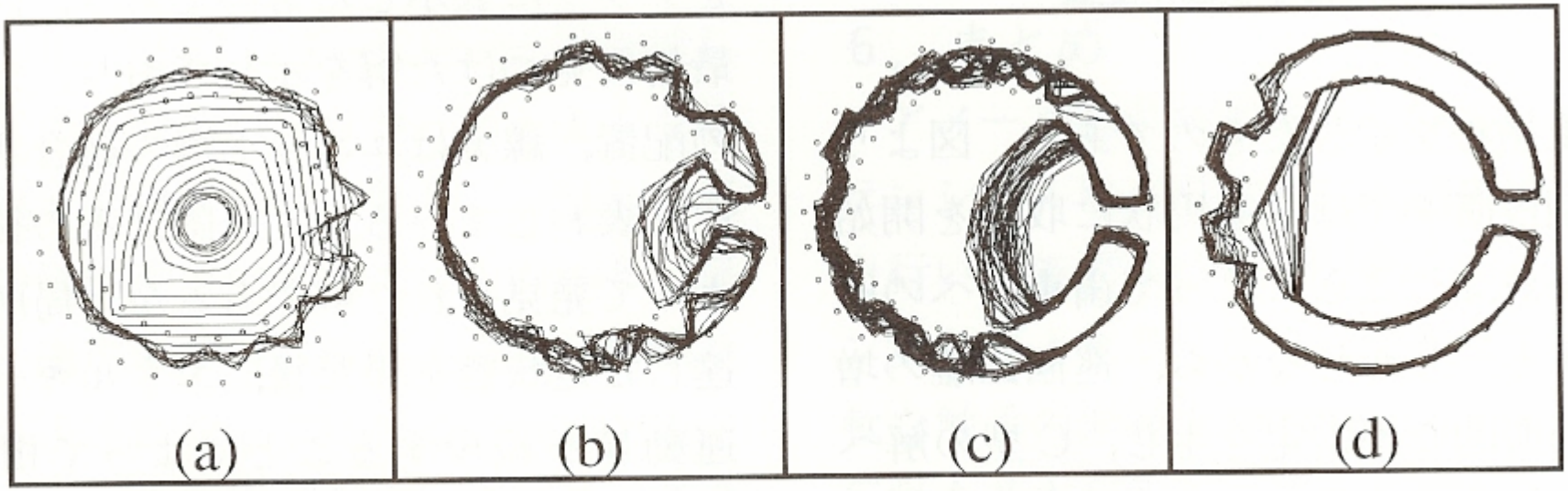
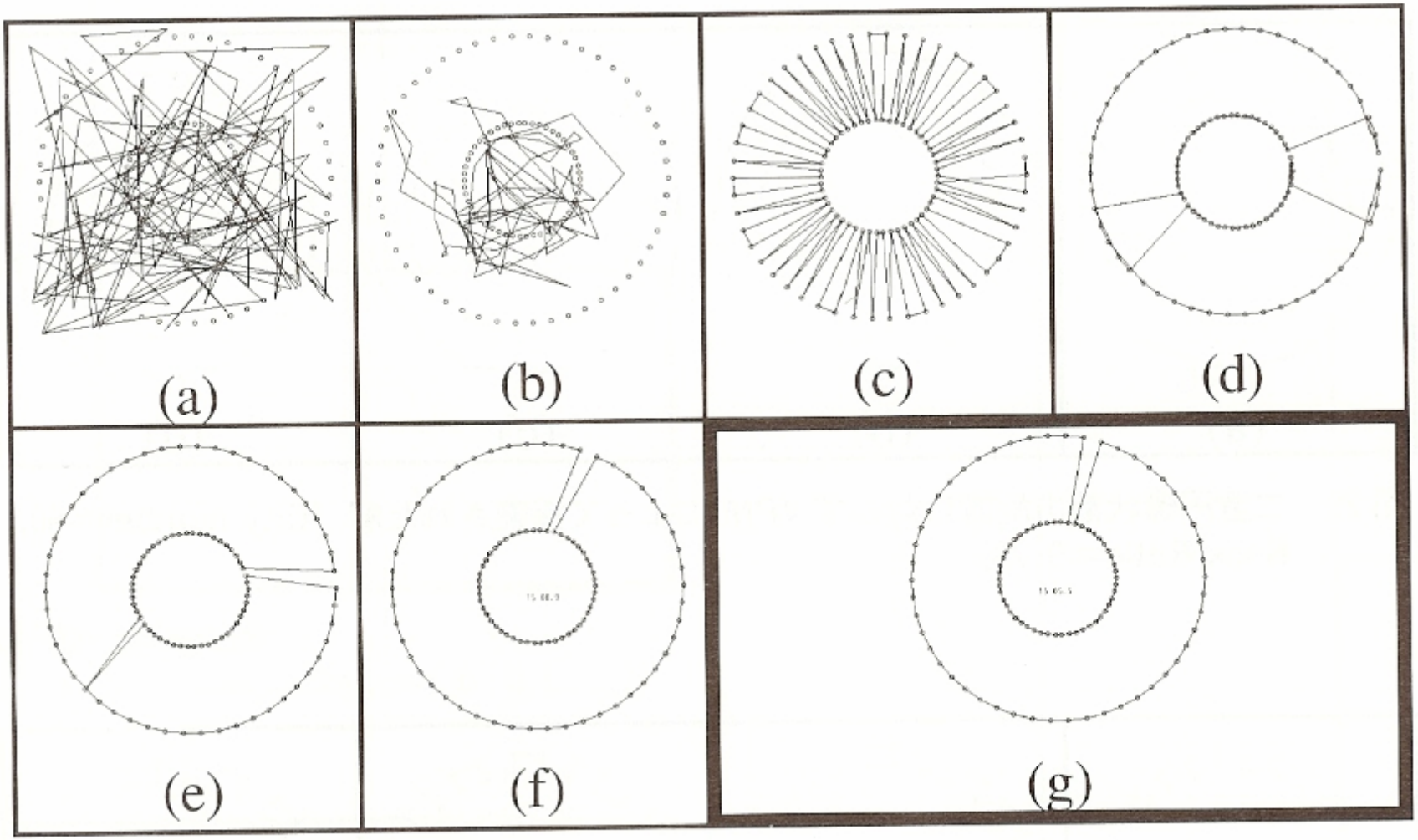


図 24　二重円環状都市配置に対するユニット群の探索軌跡（City number＝60，Boss Ratio＝0.73）



City Number = 100 Boss Ratio = 0.43

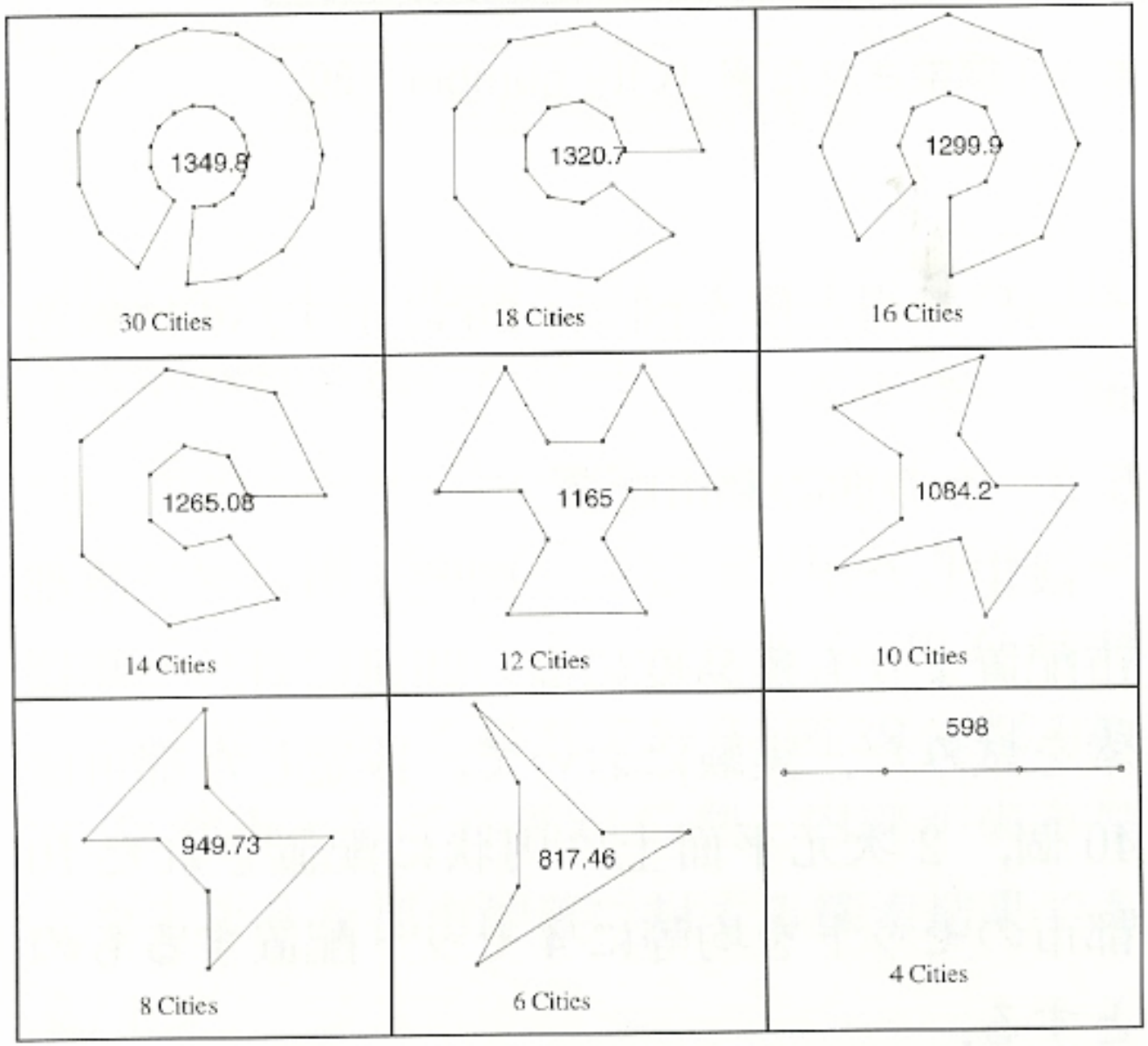


図 23　二重円環状都市配置（Boss Ratio＝0.43）

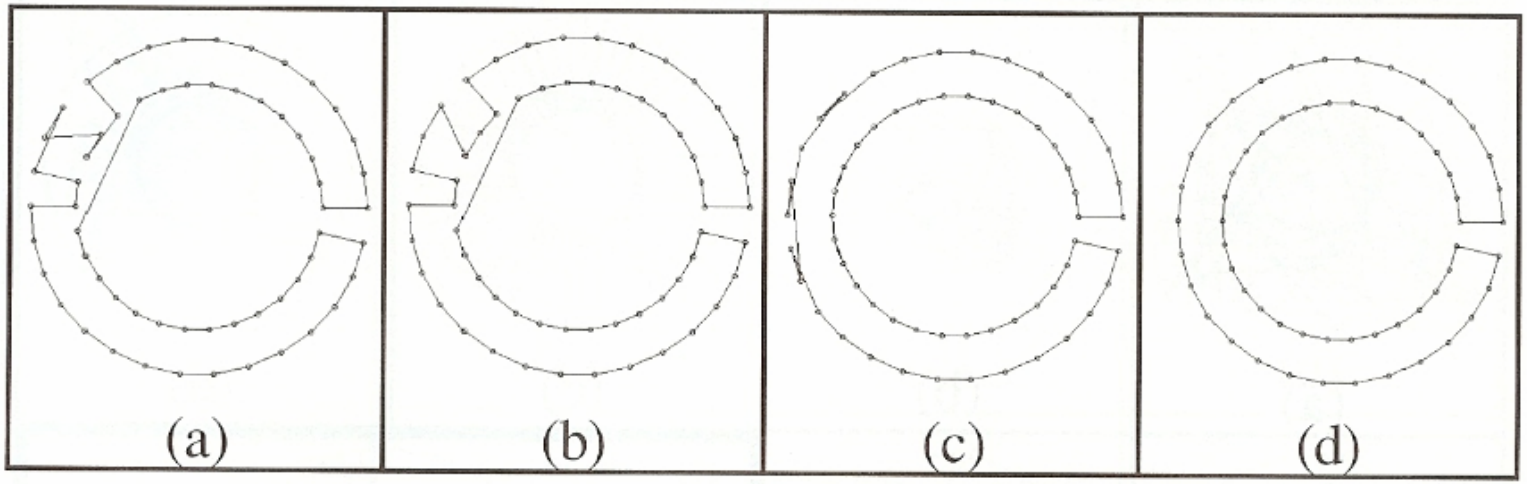


図 25　二重円環状都市配置に対して VPM によって探索された解（City number＝60, Boss Ratio＝0.73）

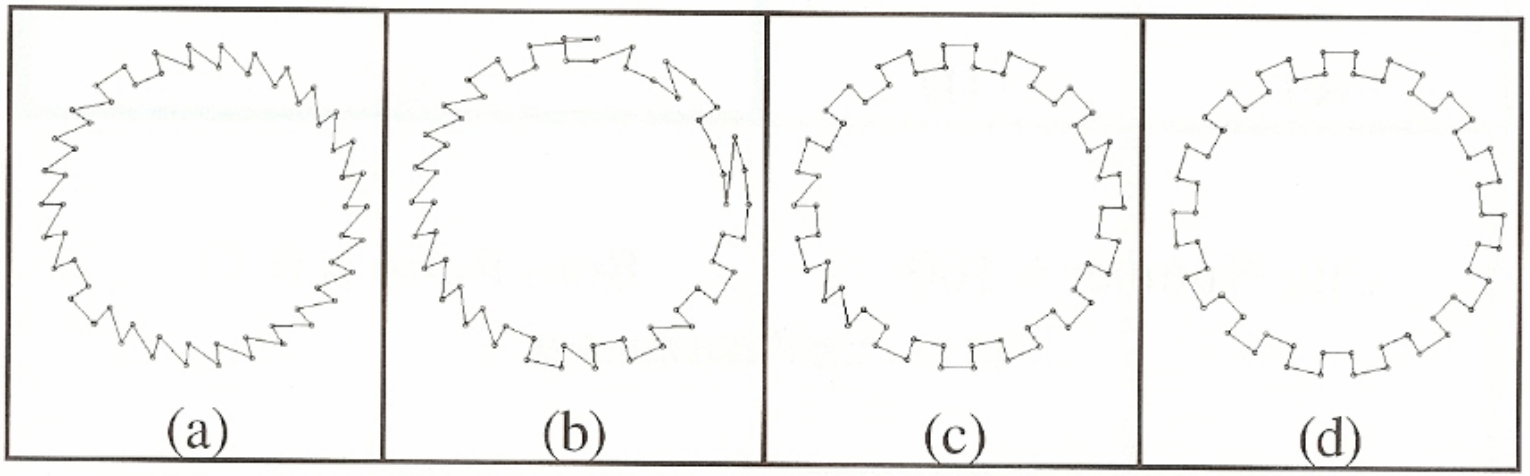


図 26　二重円環状都市配置に対して VPM によって探索された解（City number＝60, Boss Ratio＝0.86）

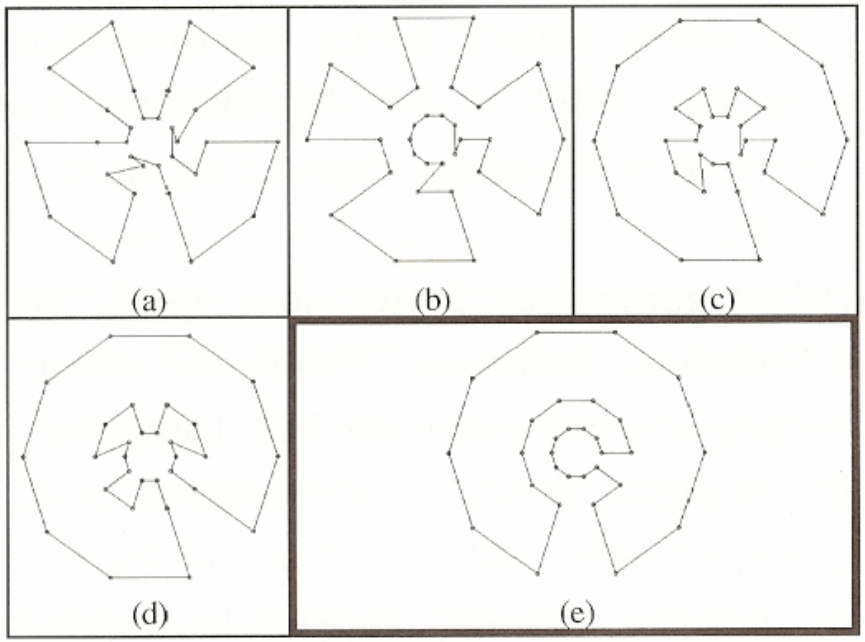


図 28　三重円環状都市配置
(City number＝30, Boss Ratio＝1.0：0.4：0.2)

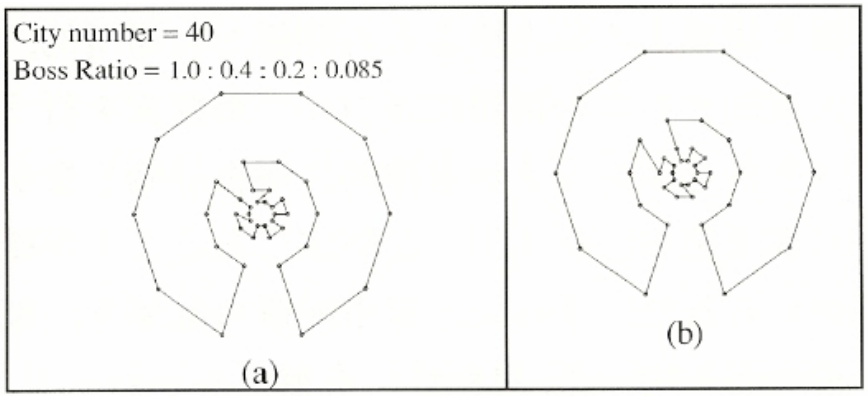


図 31　四重円環状都市配置

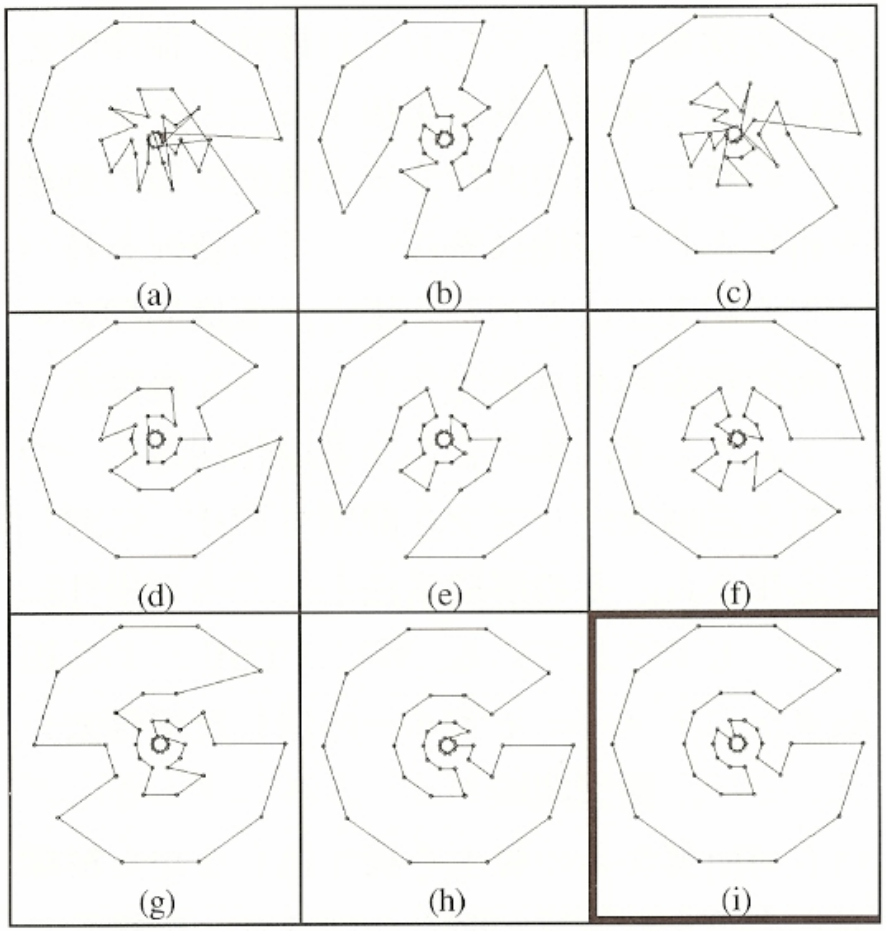


図 30　四重円環状都市配置

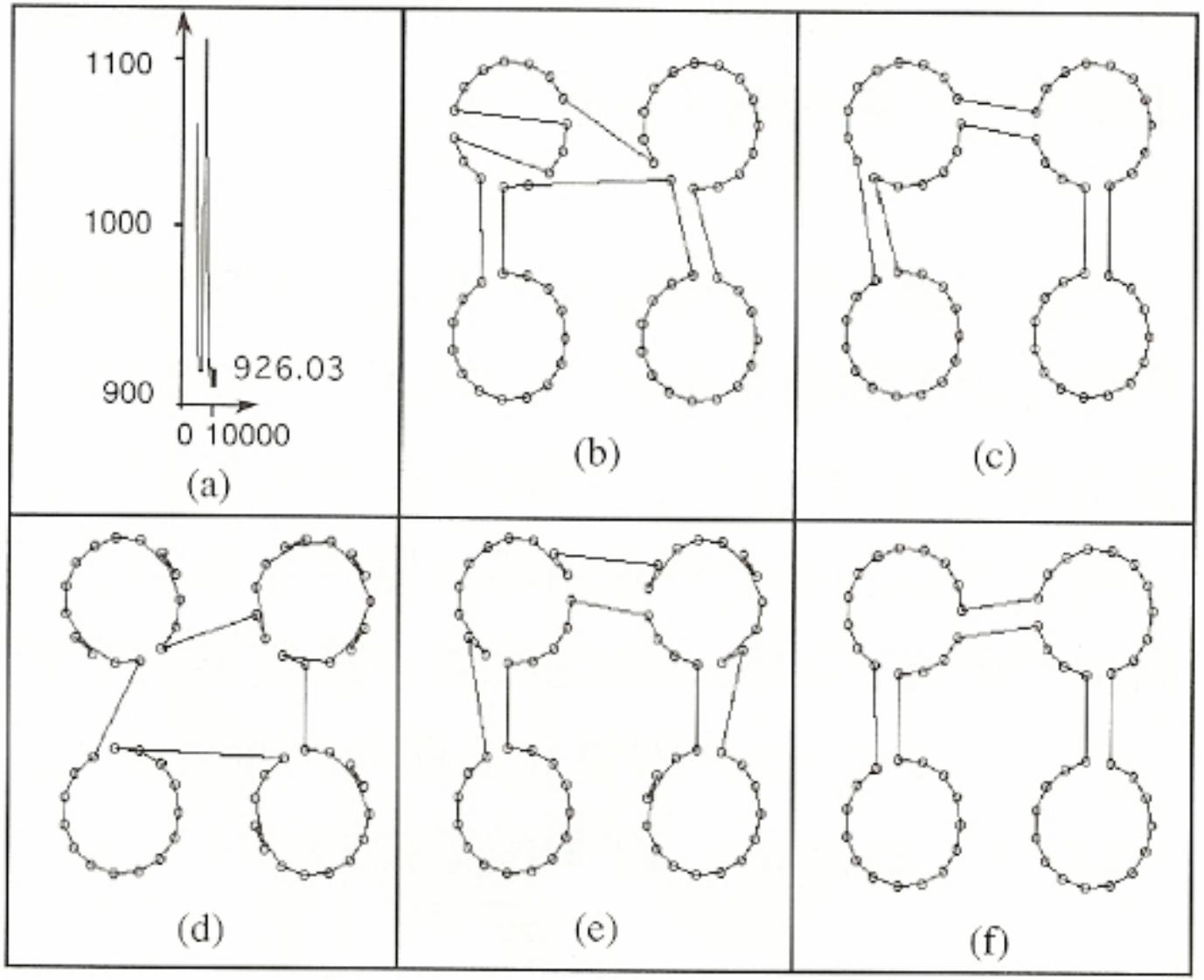


図 27　四円格子状都市配置

列挙法（例題:ナップザック問題）

問題

新規事業　$x_i = 0$（行わない）
　　$= 1$（行う）
　　（$i = 1, \ldots, m$）

b_i：予算
c_i：期待利益
a_i：経費

$$f = \sum_{i=1}^{m} c_i x_i \rightarrow \max$$

$$\sum_{i=1}^{m} a_i x_i \leq b_i$$

$$x_i = 0 or 1$$

2^n回評価

n次元整数ベクトル｛xi｝

$f = 7x_1 + 8x_2 + x_3 + 2x_4 \rightarrow \max$

$g = 4x_1 + 5x_2 + x_3 + 3x_4 \leqq 6$

$x_i = 0, 1$（$i = 1, \ldots, 4$）

(x_1, x_2, x_3, x_4)	f	g	
(0, 0, 0, 0)	0	0	
(0, 0, 0, 1)	2	3	
(0, 0, 1, 0)	1	1	
(0, 0, 1, 1)	3	4	
(0, 1, 0, 0)	8	5	
(0, 1, 1, 0)	9	6	
(0, 1, 0, 1)	10	8	×
(0, 1, 1, 1)	11	9	×
(1, 0, 0, 0)	7	4	
(1, 0, 0, 1)	9	7	×
(1, 0, 1, 0)	8	5	
(1, 0, 1, 1)	10	8	×
(1, 1, 0, 0)	15	9	×
(1, 1, 0, 1)	17	12	×
(1, 1, 1, 0)	16	10	×
(1, 1, 1, 1)	18	13	×

$x_1 = 1, x_2 = 2$
の場合は，既に制約を充足しない
制約×　　計算の無駄

解

ランダム探索

- 探索手順
  - 最小値　M＝∞，最小値n次元整数ベクトルは空
  - Step1：n次元整数ベクトルをランダムに生成
  - Step2:制約条件式により評価x∈S，ただしS⊂X
    - falseならStep1へ
  - Step3:目的関数f(x)を計算
    - すべてのn次元整数ベクトルを評価したら終了
    - Mより小なら，最小値n次元整数ベクトルを書き換えて，Step1へ.
    - Mより大なら，Step1へ

組み合わせ問題の解法

1. 先週の復習
2. ナップザック問題を例題として，
   欲張り法（近似解法）
   ケチケチ法（近似解法）
   分枝限定法（厳密解法）

直感的方法

復習　組み合わせ問題

(Combinatorial Optimization)

- 組み合わせ最適化問題とは，一般的な最適化問題における集合S およびXが，例えばn次元整数ベクトルや方策（要素列）の集合など，有限個あるいは加算無限個の要素を持つ離散集合である場合をいう.
  - 一般的な最適化問題
    - 目的関数：f(x)を最小（最大）にするxを
    - 制約条件：x∈S，ただしS⊂X　のもとで決定せよ.
    - 許容領域S：集合Xの部分集合
    - x：Sの要素
  - 組み合わせ最適化問題の例
    - 巡回セールスマン問題（Traveling Salesman Problem）
    - 輸送問題
    - スケジューリング問題
    - 最適配置問題
    - ナップザック問題

復習

列挙法(例題)

問題

新規事業　$x_i = 0$（行わない）
　　$= 1$（行う）
　　$(i = 1, \ldots, m)$

b_i：予算
c_i：期待利益
a_i：経費

$$f = \sum_{i=1}^{m} c_i x_i \rightarrow \max$$

$$\sum_{i=1}^{m} a_i x_i \leq b_i$$

$$x_i = 0 or 1$$

2^n回評価

n次元整数ベクトル{ xi }

$f = 7x_1 + 8x_2 + x_3 + 2x_4 \rightarrow \max$

$g = 4x_1 + 5x_2 + x_3 + 3x_4 \leqq 6$

$x_i = 0, 1$（$i = 1, \ldots, 4$）

(x_1, x_2, x_3, x_4)	f	g	
(0, 0, 0, 0)	0	0	
(0, 0, 0, 1)	2	3	
(0, 0, 1, 0)	1	1	
(0, 0, 1, 1)	3	4	
(0, 1, 0, 0)	8	5	
(0, 1, 1, 0)	9	6	
(0, 1, 0, 1)	10	8	×
(0, 1, 1, 1)	11	9	×
(1, 0, 0, 0)	7	4	
(1, 0, 0, 1)	9	7	×
(1, 0, 1, 0)	8	5	
(1, 0, 1, 1)	10	8	×
(1, 1, 0, 0)	15	9	×
(1, 1, 0, 1)	17	12	×
(1, 1, 1, 0)	16	10	×
(1, 1, 1, 1)	18	13	×

$x_1 = 1, x_2 = 2$
の場合は，既に制約を充足しない
制約×　　計算の無駄

解

復習

ランダム探索

- 探索手順
  - 最小値　M＝∞，最小値n次元整数ベクトルは空
  - Step1：n次元整数ベクトルをランダムに生成
  - Step2:制約条件式により評価x∈S，ただしS⊂X
    - falseならStep1へ
  - Step3:目的関数f(x)を計算
    - すべてのn次元整数ベクトルを評価したら終了
    - Mより小なら，最小値n次元整数ベクトルを書き換えて，Step1へ.
    - Mより大なら，Step1へ

復習

PMXによる都市配置

巡回セールスマン問題における制約条件は，全ての都市に一度ずつ訪れなければならないというものである.

これを満たすような都市配置を自動的に生成する方法を紹介する.

総都市数:N

都市番号列{Cn}　（1，2，3，4，5，6）

整数列{On}，On＜＝N－n，適当に選ぶ　（3，2，2，2，2，1）

都市配置列{Xn}　（3，2，4，5，6，1）

Xi=Cm，m＝Oi

Cm=Cm+1　C(m)要素を削除する･･･MathLabではC(m)=[]

欲張り法

（Greedy Method）（近似解法）

問題：ナップザック問題

目的関数：$f = \sum_{i=1}^{m} c_i x_i \rightarrow \max$

制約条件：$\sum_{i=1}^{m} a_i x_i \leq b_i$

$x_i = 0 or 1$

$$r_i = \frac{c_i}{a_i} \quad (i = 1 \cdots n)$$

x_iをr_iの大きい順
に並べかえる

- 0) $\tilde{b} = b, f = 0, j = 0$
- 1) j = j+1　j > nなら終了
- 2) $aj > \tilde{b}$　ならばx_j = 0,　1)へ
- $aj \leq \tilde{b}$　ならばx_j = 1

  $\tilde{b} = \tilde{b} - a_j, f = f + c_j$　1)へ

•r_1 = 7/4 = 1.75, r_2 = 8/5 = 1.6, r_3 = 1/1 = 1, r_4 = 2/3 = 0.67,

$$j = 1 : x_1 = 1, \tilde{b} = 6 - 4 = 2, f = 7$$
$$j = 2 : x_2 = 0$$
$$j = 3 : x_3 = 1, \tilde{b} = 2 - 1 = 1, f = 8$$
$$j = 4 : x_4 = 0$$

一番高価なものから順に入れ，一杯になったら終了.

•解（1, 0, 1, 0）f=8

•（真の解（0, 1, 1, 0）f=9）

けちけち法

（Stingy Method）（近似解法）

$$r_i = \frac{c_i}{a_i} \quad (i = 1 \cdots n)$$

- x_nをr_iの大きい順に並べ替える
- 0) $\tilde{b} = \sum_{i=1}^{n} a_i,\ f = \sum_{i=1}^{m} c_i,\ x_i = 1,\ j = n$
- 1) $\tilde{b} \le b$ ならば 3)へ
- 2) $x_j = 0, \tilde{b} = \tilde{b} - a_j,\ f = f - c_j,\ j = j - 1$ 1)へ
- 3) $x_j = 0$ となったjに対し欲張り法を適用

•j = 2, 3, 4に対し欲張り法

入る可能性のあるものを全部押し込もうとしてみる.

$$\tilde{b} = 13,\ f = 18$$

$$j = 4 : x_4 = 0, \tilde{b} = 13 - 3 = 10,\ f = 18 - 2 = 16$$

$$j = 3 : x_3 = 0, \tilde{b} = 10 - 1 = 0,\ f = 16 - 1 = 15$$

$$j = 2 : x_2 = 0, \tilde{b} = 9 - 5 = 4,\ f = 15 - 8 = 7$$

•解（1, 0, 1, 0）f = 8

押し込めない部分について価値の低いものから順にはじき出す.

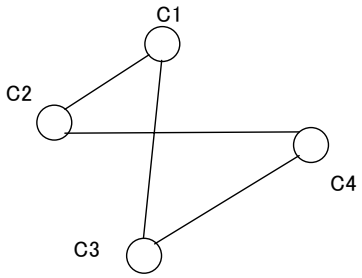
$$j = 2 : x_2 = 0$$

$$j = 3 : x_3 = 1, \tilde{b} = 4 + 1 = 5,\ f = 7 + 1 = 8$$

$$j = 4 : x_4 = 0$$

分枝限定法

(厳密解法)



方法論

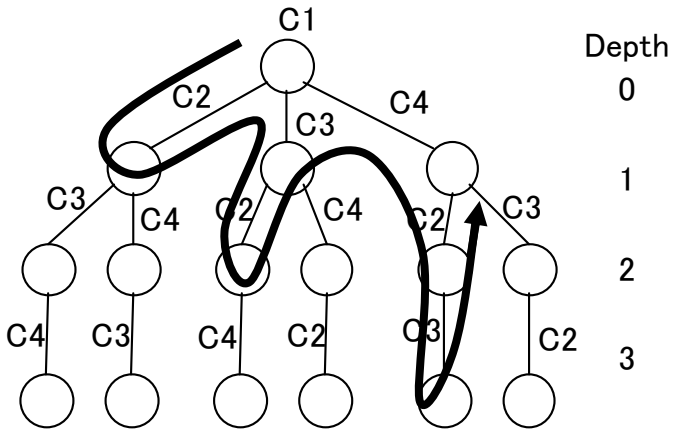
・与えられた問題P0を，複数の部分問題(Partial problems)Pi(i=1,2,...)に分解する(分枝操作:Branching operation)

・部分問題Piを何らかの方法で終端する(限定操作:bounding operation)

ただし，ある部分問題Piを終端するとは

(1)探索中にその部分問題Piの最適解が求まった場合か，あるいは

(2)その部分問題Piに元の問題P0の最適解が存在しないことが何らかの方法で判明した場合.



分枝限定法の方法

分枝操作と限定操作

ここで扱う最適化問題P0を次のように定義する.

最小値探索問題P0

目的関数:f(x)を最小にするxを

制約条件:x∈S, ただしS⊂Xの元で決定せよ.

ただしSおよびXは有限個あるいは加算無限個の要素を持つ離散集合.

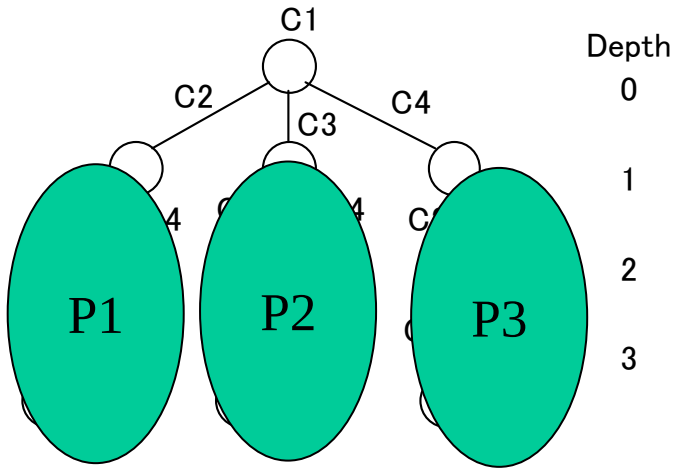
分枝限定法は, 分枝操作と限定操作から構成される.

・分枝操作:P0の部分問題

Pi(i=1,2,....)への分解

・限定操作:部分問題

Pi(i=1,2,....)の終端



分枝操作と限定操作

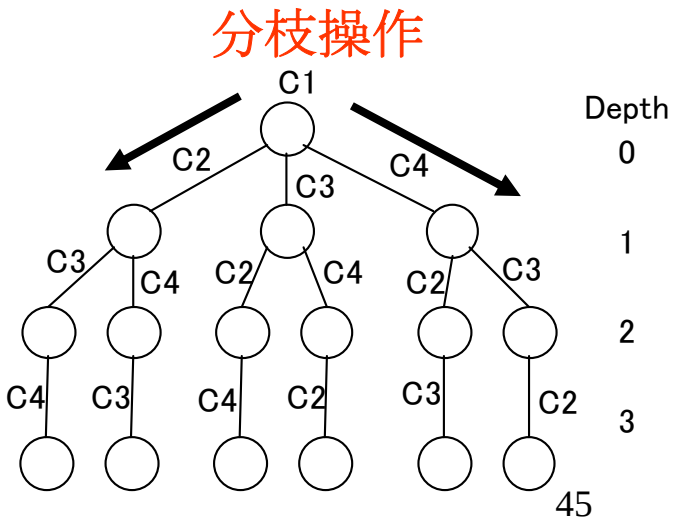
分枝操作は与えられた問題に対して，一意に決定されるものではなく，同じ問題に対しても，様々な分枝操作が考えられる．分枝操作によって，分枝図の構造が大きく変化し，処理効率に多大な影響がある．

分枝操作だけを用いる場合，分枝限定法は，列挙法と同一である．

限定操作は，分解された部分問題について，「それ以上は調べる必要はない」と判定して，探索を打ち切る処理であり，処理の効率化のために非常に重要．

1．下界値による限定操作

2．優劣関係を用いる限定操作



緩和問題

ある部分問題Piを調べる際，Piよりも制約条件がゆるく，解きやすい問題を考えてP~iとおく．
Piの緩和問題P~I

x：実数（小数点以下を許す）

目的関数　：　g(x)→最小
制約条件　：　x∈S~i　ただしSi⊂S~i⊂X,
g(x)≦f(x)，x∈Si

すなわち，Piの最適値をf(Pi)，P~iの最適値をg(Pi)とおくと，上の制約条件に示されるように

$$g(Pi) \leqq f(Pi)$$

このため，g(Pi)を下界値と呼ぶ．

下界値の性質

下記のような下界値の性質により，部分問題Piを終端するかどうかが決まる．
（分枝図の木において，Piより深い部分については，調べる必要がないことを判定）

（1）P~iが許容解を持たなければ，Piももたない．
（部分問題Piに元の問題P0の許容解が含まれていない）

（2）P~iの目的関数にg（x）＝f（x）を用いるとき，P~iの最適解がPiの許容解であれば，それはPiの最適値でもある．（部分問題Piの最適解が求まった）

（3）それまでに他の部分問題によって，元の問題P0の許容解が求められており，その中の最小値（すなわち現時点で最も優れた解）をz（暫定値(incumbent value)）とするとき，

$$g(Pi) \geqq z$$

が成立する（下界値テスト）
（部分問題Piおよびそれから生成される部分問題に対する許容解はいずれもz以上の値を持つ，すなわち，現時点の最良の解より優れた解が含まれる可能性がない）

優劣関係を用いる限定操作

ある部分問題PiとPj(i≠j)において，次式が成り立つことがわかるとき，

$$f(Pj) \leqq f(Pi)$$

部分問題PjはPiに優越する（dominate）という．このため，部分問題Piよりも優越する他の部分問題Pjが既に生成されているなら，Piを終端することができる（優越テスト）．

分枝限定法における限定操作

（1） Piの緩和問題P~iが許容解を持たなければ，Piを終端する．

（2） g(Pi)=f(Pi)すなわちP~iの最適値がPiの最適値に一致すれば，Piを終端する．

（3） （下界値テスト） g(Pi)≧zが成立すれば，Piを終端する．

（4） （優越テスト） f(Pj)≦f(Pi)すなわちPiより優越する部分問題Pjが既にあることがわかれば，Piを終端する．

探索法

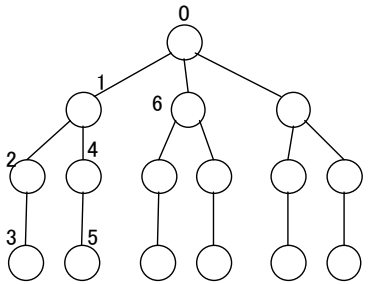
問題を部分問題に分解する手順について述べる.
これは，解くべき部分問題をAとするとき，Pi=s(A)という探索関数(search function)をけっていすることである．探索関数としては，発見的探索，最良下界探索，深さ優先探索，幅優先探索などがある.

(1)発見的探索(Heuristic search)

(2)最良下界探索(best-bound search, minimum value search)

(3)深さ優先探索(depth-first search)

(4)幅優先探索(breadth-first search)



(1)発見的探索(Heuristic search)

部分問題Piの最適値f(Pi)に対する推定値h(Pi)を求めることができる発見的関数(heuristic function)hをもちい,

$$s(A) = Pi|h(Pi)=\min\{h(Pi)|P\in A\}$$

すなわち, Aのうち最小のh値をもつPiを選択する方法である. ここでは, 発見的関数hとして, 目的関数f(x)に近い関数を設計することができるか否かが探索の成功を握る鍵となり, 良好なhを設計することは, 問題にも依存するが一般的に非常に困難である.

(2)最良下界探索(best-bound search, minimum value search)

発見的関数hとして, 下界値関数gを用いた探索法である. この探索法では, あらゆる探索法の中で, 計算終了までに分解する部分問題の数を最小にするという特徴がある. しかしながら, 計算の最終段階になるまで暫定解が得られず, 計算を途中で打ち切ると, それまでの計算がまったく無駄になる可能性が高いという特徴を持っている.

(3)深さ優先探索(depth-first search)

計算の各時点で最大の深さの部分問題から優先的に探索する方法であり, 線形探索(linear search), LIFO(Last-In-First-Out)探索ともよばれる. この探索法による探索順序の例を図に示す.

同図に示されるように, 深さ優先探索では, 選択された部分問題に関する処理が, 終了するまで, 同じ深さの他の部分問題は, 調べられないことがわかる.

(4)幅優先探索(breadth-first search)

計算の各時点で深さが最小の部分問題から処理する方式であり, 深さ優先探索とは, 対作用的な探索法である.

分枝限定法の処理手順(Procedure of Branch and Band)

最適解が一個の場合
A:既に生成されているがまだ分解も終端もされていない部分問題の集合
N:既に生成されている部分問題の集合.
z:最適値の暫定値
O:最適解の集合

Step1初期設定: A={P0}, N={P0}, z=∞, および, O=∅(空集合).
Step2探索処理: A=∅ならStep9へ, A≠∅ならPi=s(A)としてStep3へ.
Step3暫定値更新:x∈Sかつf(x)＜zの解xがあれば, z＝f(x), O={x}としStep4へ.
Step4緩和問題によるテスト:次のいづれかが成り立てば, Step8へ, それ以外はStep5へ
.

(1)Piの緩和問題P~iが許容解を持たない.
(2)g(Pi)=f(Pi), すなわち, P~iの最適値がPiの最適値に一致する.

Step5下界値テスト:g(x)≧zが成立すれば, Step8へ, それ以外はStep6へ.
Step6優越テスト:Piより優越するPjが既にあれば, Step8へ, それ以外はStep7へ
Step7分枝処理:Piの子節点Pi1,Pi2,....Pinを作り,
A = A∪{Pi1,Pi2,....Pin}－{Pi}, N = N∪{Pi1,Pi2,....Pin}としてStep2へ.
Step8部分問題Piの終端処理: A = A-{Pi} としてStep2へ
Step9停止
z = ∞のとき, 与えられた問題P0は許容解を持たない.
z＜∞のとき, 最適値＝z, 最適解は, Oに記憶されている.

０－１ナップザック問題：

目的関数：　$z = -\sum_{j=1}^{n} c_j x_j$　→最小　$x_j = 0,1 (1 \le j \le n)$

制約条件：　$\sum_{j=1}^{n} a_j x_j \le b,$

これは線形計画問題の一種であるが，制約条件の不等式がただ1つの場合であり，ajを品物j(1≦j≦n)の容積，cjを品物jの価値とすると，容積bの袋に総合的な価値が最大となるように，詰め込むべき品物を選ぶ（xj＝０，１）問題である．この問題を分枝限定法で解くにあたり，次に示すようなxjに0以上1以下の実数値を許す緩和問題を設定する．

０－１ナップザック問題の緩和問題：

目的関数： $z = -\sum_{j=1}^{n} c_j x_j$ →最小　　$0 \le x_j \le 1 (1 \le j \le n)$

制約条件： $\sum_{j=1}^{n} a_j x_j \le b,$

このような設定の下で，上記の0−1ナップザック問題を解く場合は，特に次の事項が成り立つことが証明できるので，限定操作の際に利用することができる．

（1）Pi~の最適解が整数解なら，それは元の部分問題Piの最適解である．

（2）次のような部分問題P1,P2に対して，

$$P1: z = -\sum_{j=1}^{n} c_{1j} x_{1j} \quad \text{→最小，ただし，} \sum_{j=1}^{n} a_{1j} x_{1j} \le b, \quad x_{1j} = 0,1 (1 \le j \le n)$$

$$P2: z = -\sum_{j=1}^{n} c_{2j} x_{2j} \quad \text{→最小，ただし，} \sum_{j=1}^{n} a_{2j} x_{2j} \le b, \; x_{2j} = 0,1 (1 \le j \le n)$$

次式が成り立つとき，

$$-\sum_{j=1}^{n} a_{1j} x_{1j} \le -\sum_{j=1}^{n} a_{2j} x_{2j} \quad \text{かつ} \quad -\sum_{j=1}^{n} c_{1j} x_{1j} \le -\sum_{j=1}^{n} c_{2j} x_{2j}$$

f(P1) <= f(P2) すなわち部分問題P1は部分問題P2に優越する．

例題P0:

目的関数 $z= -5\ x_1 -4\ x_2 -3\ x_3 -2\ x_4$ $\quad$ ->Min

制約条件 $2x_1 +2x_2 +3x_3 +3x_4 <= 5$ $\quad$ 0<=x1<=1(1<=j<=4)

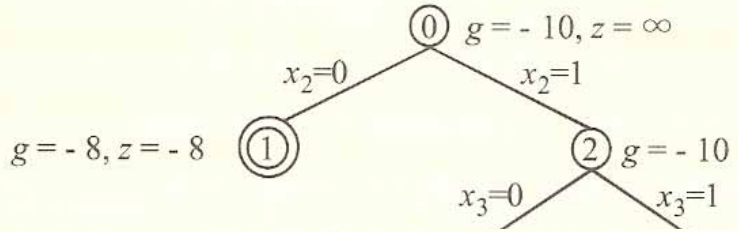
緩和問題の最適解

Step1 xi(1<=i<=n)をci/aiの大きい順に優先度を割り付ける.

{ 5/2, 4/2, 3/3, 2/3}

Step2 優先度順にxi=1とおいてゆく. ただし, xi=1とおくと条件式が満たされなくなる場合には, xi=K(1/ai)(Kは定数)とおいて等式が成立するようにKを決定し, それ以降のxiについては, xi=0とおく.

０－１ナップザック問題の解法



（節点０）

先に示した緩和問題 $\overline{P_0}$ を前述のように解いて次の解を得た．

$$(x_1, x_2, x_3, x_4) = (1, 1, 1/3, 0) \rightarrow \text{下界値：}\ g = -10$$

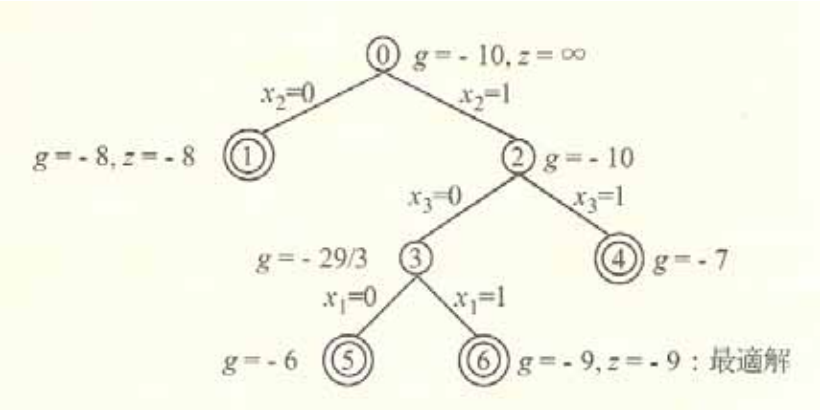
これは整数解ではないので最適値の暫定値 $z = \infty$，最適解の集合 $O = \phi$（空集合）．x_2 の値によって２つの節点０を節点１と節点２に分解した．

（節点１）

部分問題 $\overline{P_1}: -5x_1 - 3x_3 - 2x_4 \rightarrow$ 最小，条件: $2x_1 + 3x_3 + 3x_4 \leq 5$ を解いて $(x_1, x_2, x_3, x_4) = (1, 0, 1, 0) \rightarrow g = -8$ となる．これは整数解なので部分問題 P_1 の最適解でもあるから $z = -8$ とおき，このときの x を O に代入してからこの節点を終端した．

（節点２）

部分問題 $\overline{P_2}: -4 - 5x_1 - 3x_3 - 2x_4 \rightarrow$ 最小，条件: $2x_1 + 3x_3 + 3x_4 \leq 3$ を解いて $(x_1, x_2, x_3, x_4) = (1, 1, 1/3, 0) \rightarrow g = -10$ となる．分枝限定法の処理手順 4～6 の



テストの後，節点２を変数 x_3 の値によって節点３と４に分枝した．

（節点３）

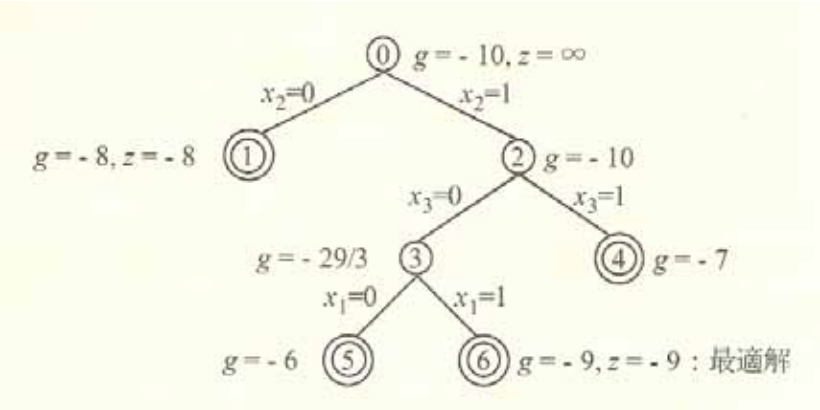
部分問題 $\overline{P}_3 : -4-5x_1-2x_4 \to$ 最小，条件: $2x_1+3x_4 \le 3$ を解いて

$(x_1, x_2, x_3, x_4) = (1, 1, 0, 1/3) \to g = -29/3$ となる．分枝限定法の処理手順 4～6 のテストの後，節点３を変数 x_1 の値によって節点５と６に分枝した．

（節点４）

部分問題 $\overline{P}_4 : -7-5x_1-2x_4 \to$ 最小，条件: $2x_1+3x_4 \le 0$ を解いて

$(x_1, x_2, x_3, x_4) = (0, 1, 1, 0) \to g = -7$ となる． $g = -7 > z(=-8)$ なのでこの節点は終端された．



（節点５）

部分問題 $\overline{P}_5 : -4-2x_4 \to$ 最小，条件: $3x_4 \le 3$ より $(x_1, x_2, x_3, x_4) = (0,1,0,1) \to$ $g = -6 > z(=-8)$ となり，この節点は終端された．

（節点６）

部分問題 $\overline{P}_6 : -9-2x_4 \to$ 最小，条件: $3x_4 \le 1$ より $(x_1, x_2, x_3, x_4) = (1,1,0,0) \to$ $g = -9$　これは整数解であり，かつ今までの z の値 $(=-8)$ より小さいので $z = -9$ と更新し，解 $(x_1, x_2, x_3, x_4) = (1,1,0,0)$ を O に代入して終端した．

以上の操作によって全ての節点が終端され，処理が終了した．この結果次の最適解および最適値を得た．

0-1 ナップザック問題の例題 P_0 ：

最適解：$(x_1, x_2, x_3, x_4) = (1,1,0,0)$，**最適値**：$z = -9$

非対称・巡回セールスマン問題

非対称・巡回セールスマン問題とは，都市iからjへの距離（コスト）をdijとするとき，dijが必ずしもdjiと等しくないような巡回セールスマン問題である.

ここで，都市iからjに直接向かう経路があるときxij＝1，ないときxij=0とし，i=jのときdij=∞とすると，

非対称巡回セールスマン問題：

目的関数： $z = \sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij} x_{ij}$ → 最小

制約条件： $\sum_{i=1}^{n} x_{ij} = 1, \quad i = 1,2,\cdots,n$ かつ $\sum_{j=1}^{n} x_{ij} = 1, \quad j = 1,2,\cdots,n$

$x_{ij} = 0,1 \quad i = 1,2,\cdots,n \quad j = 1,2,\cdots,n$

注意） 一巡閉路になるかどうかのチェックを必ず行なう

この制約条件だけでは一巡閉路かどうかを判別できない

下界値の導出（緩和問題の解法）

巡回セールスマン問題における緩和問題は，一巡閉路でなければならないという制約をはずした場合であると考えられる.

よって，現時点で対象とすべき距離dijの表をDとあらわし，次の変換を考えて，下界値を導出する.

変換1) 表Dの各行について，各行の最小値をそれぞれの値から引くとともに，各行の最小値の和Slを求める.
変換2) 次に，各列について，各列の最小値をそれぞれの値から引くとともに，各列の最小値の和Scを求める.
変換3） g=Sl+Scとする.

i \ j	1	2	3	4	5
1	∞	21	7	13	15
2	11	∞	19	12	25
3	15	24	∞	13	5
4	6	17	9	∞	22
5	28	6	11	5	∞

表D

下界値導出のための変換1)

- 変換1)
- 表Dの各行について，各行の最小値をそれぞれの値から引くとともに，各行の最小値の和Slを求める.

i \ j	1	2	3	4	5	最小値
1	∞	21	7	13	15	7
2	11	∞	19	12	25	11
3	15	24	∞	13	5	5
4	6	17	9	∞	22	6
5	28	6	11	5	∞	5

i \ j	1	2	3	4	5	最小値
1	∞	14	0	6	8	-7
2	0	∞	8	1	14	-11
3	10	19	∞	8	0	-5
4	0	11	3	∞	16	-6
5	23	1	6	0	∞	-5

下界値導出のための変換

- 変換2)　次に，各列について，各列の最小値をそれぞれの値から引くとともに，各列の最小値の和Scを求める.

- g = Sl + Scとする.

i \ j	1	2	3	4	5	最小値
1	∞	14	0	6	8	-7
2	0	∞	8	1	14	-11
3	10	19	∞	8	0	-5
4	0	11	3	∞	16	-6
5	23	1	6	0	∞	-5
最小値	0	1	0	0	0	

i \ j	1	2	3	4	5	最小値
1	∞	13	0	6	8	-7
2	0	∞	8	1	14	-11
3	10	18	∞	8	0	-5
4	0	10	3	∞	16	-6
5	23	0	6	0	∞	-5
最小値	0	-1	0	0	0	

下界値

- これらの操作は，いずれも与えられた問題の最短巡回路そのものを変化させることはない．この変換前の最短経路長をz，変換後の最短経路長をz’とすると，
- $z = z' + g \qquad \therefore z \geqq g$
- となり，gを下界値として利用できる．

- 具体的には，
- 表Dに対して，これに対する上記の変換により，結果としてD’を得る．また，変換中に各行，各列から引いた値（欄外にマイナスで表記されている値）の和は，35となるので，この時の下界値**g=35**となる．

i \ j	1	2	3	4	5	最小値
1	∞	13	0	6	8	-7
2	0	∞	8	1	14	-11
3	10	18	∞	8	0	-5
4	0	10	3	∞	16	-6
5	23	0	6	0	∞	-5
最小値	0	-1	0	0	0	

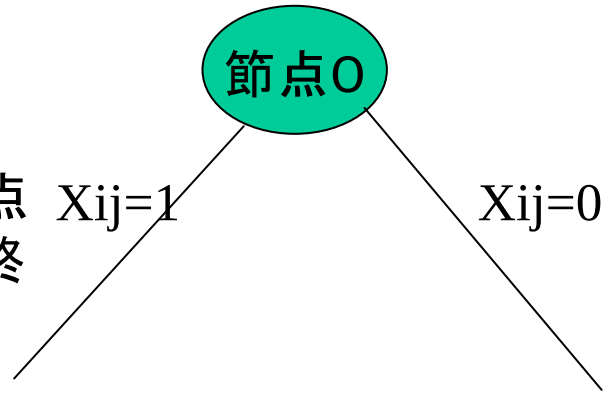
表D’

分枝の方法

次に，木を生成しながら，探索を行う場合に，探索のある時点において，xijの値によって分枝する際，それぞれの場合について，次のようにして下界値を計算できる.

（1）xij=1の場合：
　都市i→jが決定したので，現在の距離の表Dにおいて，もはや不要となったi行とj列を取り除いて新たに表を作り，このようについて，上記野下界値を計算し，dijを加えて，さらにその節点までに選択した経路の長さの和を加えて最終的な下界値とする.

（2）xij=0の場合：
　都市i→jを選択しないので，dij=∞とする.
あらためて，下界値を計算し，さらにその節点までに選択した経路の長さの和を加えて最終的な下界値とする.



分枝の方法（具体例）

経路3→5を選ぶ場合（P35と表記）：$x_{35}=1$
図1のような変換を行い，差し引いた数値の合計が30となり，$d_{35}=5$を加えて，下界値g=35となる.

経路3→5を選択しない場合（$\overline{P_{35}}$と表記）：　$x_{35}=0$
図2のような変換を行い，g=51となる.

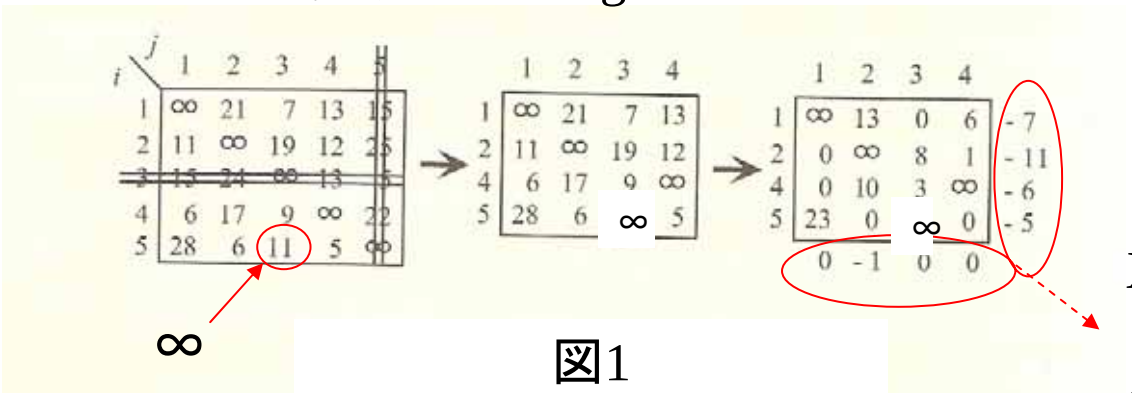


図1

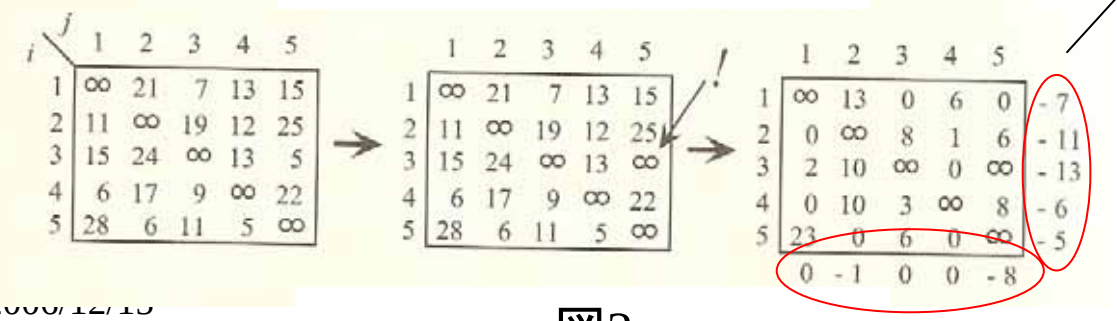
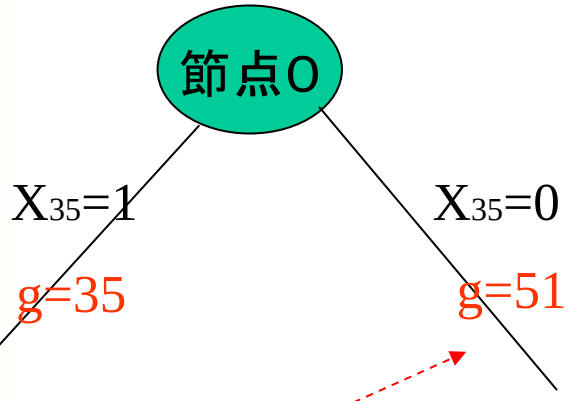


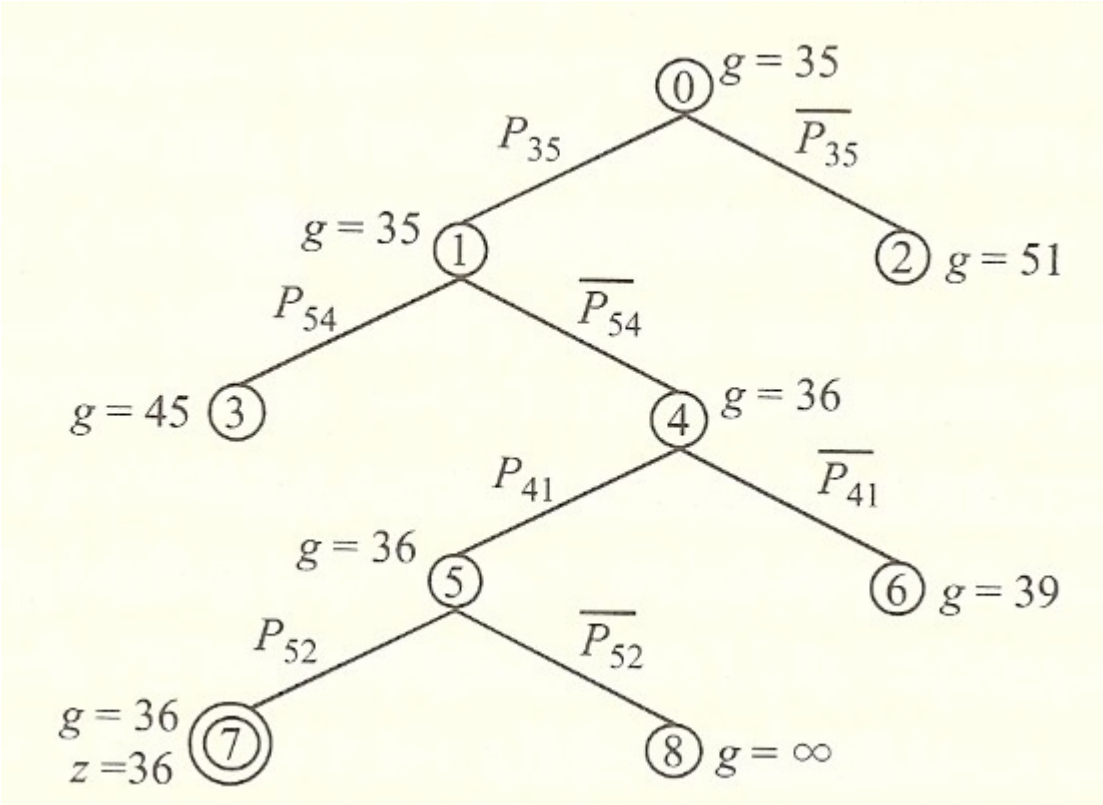
図2



表Dに対する分枝図

このように，順次下界値を求めながら，許容解が得られるまで処理を繰り返す．
表Dに対して，最良下界探索を実行した結果の分枝図を以下に示す．

２分岐の

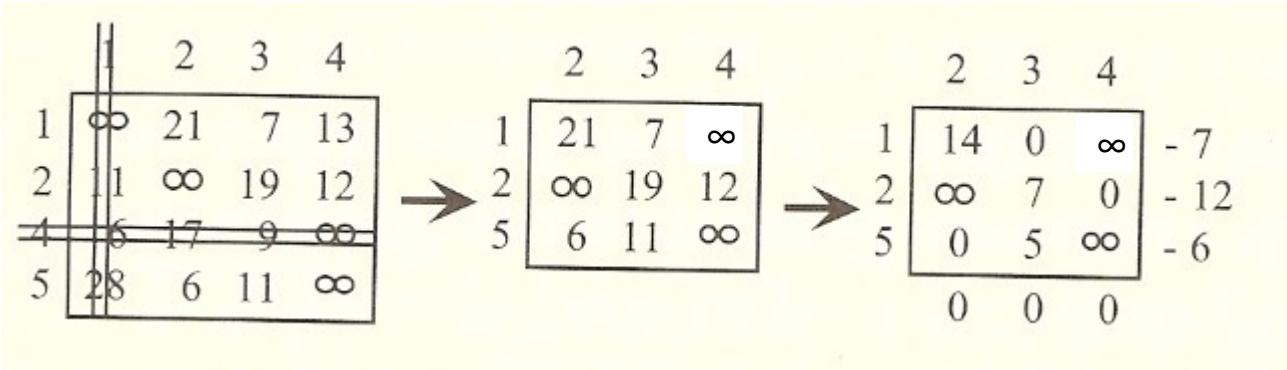


一巡閉路になるかどうかのチェックを必ず行なう

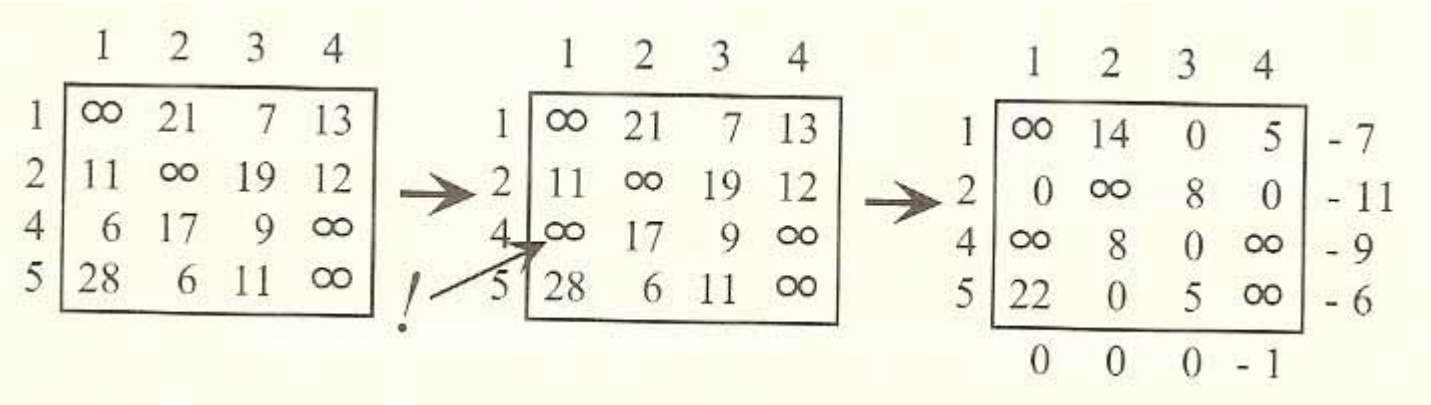
(部分問題4：$P_{35}\overline{P_{54}}$）次図の計算値３１にd_{35}=5を加えて，g=36を得た．現時点で最良の下界値なので，部分問題4を経路4→1を用いて部分問題5と6に分解した．

$$
\begin{array}{c|cccc}
 & 1 & 2 & 3 & 4 \\
\hline
1 & \infty & 21 & 7 & 13 \\
2 & 11 & \infty & 19 & 12 \\
4 & 6 & 17 & 9 & \infty \\
5 & 28 & 6 & 11 & 5
\end{array}
\rightarrow
\begin{array}{c|cccc}
 & 1 & 2 & 3 & 4 \\
\hline
1 & \infty & 21 & 7 & 13 \\
2 & 11 & \infty & 19 & 12 \\
4 & 6 & 17 & 9 & \infty \\
5 & 28 & 6 & 11 & \infty
\end{array}
\rightarrow
\begin{array}{c|cccc|c}
 & 1 & 2 & 3 & 4 & \\
\hline
1 & \infty & 14 & 0 & 5 & -7 \\
2 & 0 & \infty & 8 & 0 & -11 \\
4 & 0 & 11 & 3 & \infty & -6 \\
5 & 22 & 0 & 5 & \infty & -6 \\
\hline
 & 0 & 0 & 0 & -1 &
\end{array}
$$

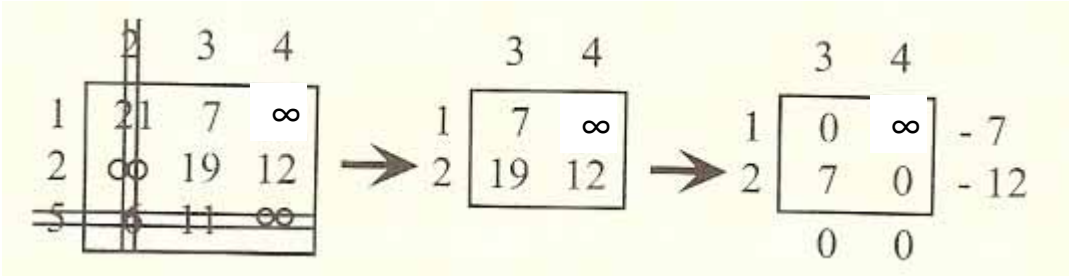
(部分問題５：$P_{35}\overline{P_{54}}P_{41}$）次図の計算値２５に$d_{35}=5$と$d41=6$を加えて，**g=36**を得た．



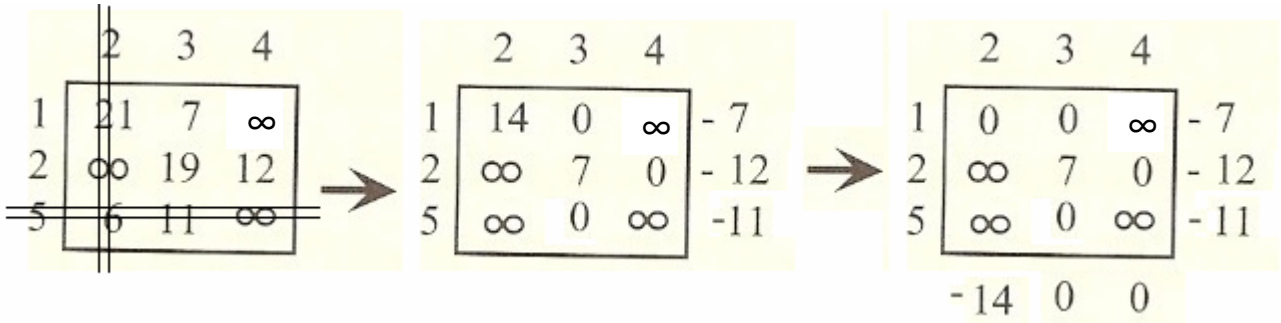
(部分問題６：$P_{35}\overline{P_{54}}\overline{P41}$）次図の計算値３４に$d_{35}=5$を加えて，**g=39**を得た．この時点で，部分問題5の下界値が最小なので，部分問題5を経路5→2を用いて，部分問題7と8に分解した．



(部分問題7：$P_{35}\overline{P_{54}}P_{41}P_{52}$）次図の計算値19に$d_{35}$=5, d41=6, d52=6を加えて，**g=36**を得た．この時点で，巡回路1→3→5→2→4→1が決定し，許容解を得たので，目的関数の暫定値**z=36**とした．



(部分問題8：$P_{35}\overline{P_{54}}P_{41}\overline{P_{52}}$）次図の計算値44に$d_{35}$=5, d_{41}=6,を加えて，**g=55**を得た．この時点で，部分問題7が最適値を与えることがわかる．



以上，分枝限定法の例題として，0-1ナップザック問題と非対称巡回セールスマン問題を扱い，それぞれの詳しい解法について述べた．

分枝限定法を有効に機能させるためには，与えられた問題の緩和問題の定義および下界値の決定方法が重要となる．これらが適切に設定できる問題に対しては，分枝限定法はきわめて有効な方法といえる．

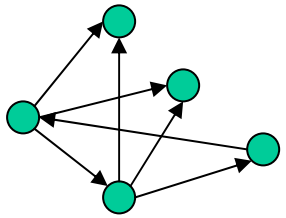
他方，これらの定義が容易ではない問題も多数存在する．例えば，将棋の戦略けって問題などでは，一つの局面において，可能な次の指し手が平均80個程度存在するといわれており，1つの部分問題が80個の部分問題に枝分かれすることになる．このため，分枝限定法を用いて，数手先までの状況変化を考慮した現状での最善手を決定することは事実上困難になる．このような場合，何らかの制約条件により可能な指し手を数手に制限するなど，扱う問題独自の特徴や知識を利用した工夫が必要になる．

グラフとネットワーク

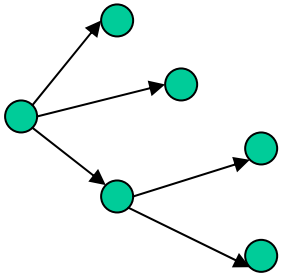
- グラフ
  - 節点(Vertex) V
  - 枝(Edge) E
  - グラフ G = (V, E)
- 有向グラフ
- 無向グラフ

グラフの種類

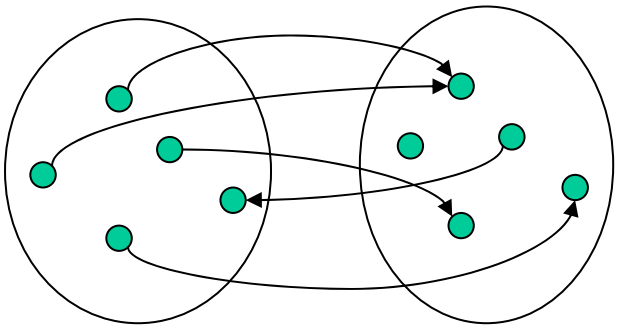
- ネットワーク



- ツリー（根付き木）
  - ・　巡回セールスマン問題
  - ・　最短路問題
  - ・　最大流量問題



- 2部グラフ



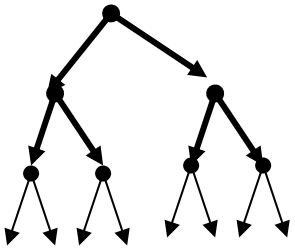
問題の解法種類

- 問題は目的関数，制約条件により記述
  - 1価関数 vs 多目的問題(ex.安くて良いものを買う)
- 線形計画法(LP: Linear Programming)
  - 目的関数が線形関数(1次関数)
  - 制約条件が連立1次(不)等式
- 整数計画法(IP: Integer Programming)
- 0/1計画法
  - 0/1 Knapsack Problem
  - 1つのナップサックに出来るだけ良いものを詰める問題
- 組合せ最適化問題(Combinatorial Optimization)
  - 組合せ問題
- TSP(巡回セールスパーソン問題)
  - 順序問題

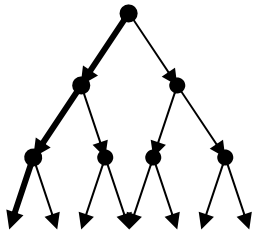
根つき木の探索法

- 節点vから枝分かれする点の集合T(v)
- 0) A = {v_0}, B = {v_0}
- 1)　※Aから<u>一つの成分vを取り出し</u>，T(v)の中でBに入っていないものをAとBに入れる.
- 2)　Aが空なら終了
- そうでなければ1)へ

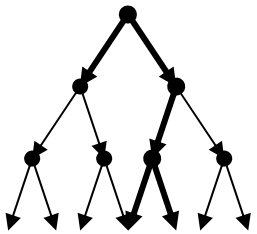
  – ※　先にいれたものを先に取り出す
  　　…　キュー（queue）
  – 　後にいれたものを先に取り出す
  　　…　スタック（stack）



幅優先探索
キュー

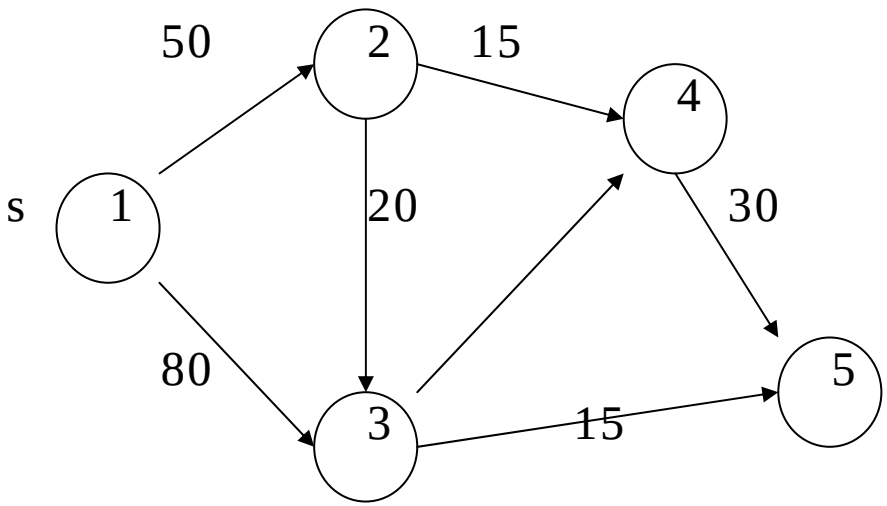


深さ優先探索
スタック



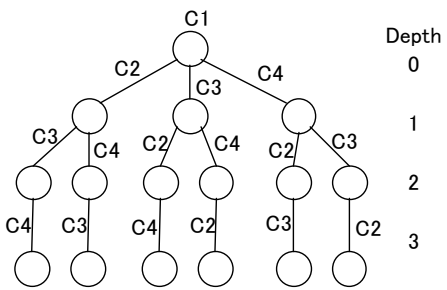
最短路問題

- 長さつきネットワーク



s → t への最短路

探索木



ダイクストラ法（Dijkstra Method）

iからjへの長さ　aij≧0	p(i)：iへの最短路へのポインタ
d(i)：sからiへの最短長さの上限	T(v)：vから出る節点の集合

手続き(Procedure)

0) d(s) = 0, d(i) = ∞　（i ≠ s）

$$A = \phi, \bar{A} = V$$

1) A = Vならば終了
そうでなければ

$$d(v) = \min\left\{ d(i) \mid i \in \bar{A} \right\}$$

2) $A = A \cup \{v\}$

$\bar{A} = \bar{A} - \{v\}$

$T(v) \cap \bar{A}$　の節点jに対し

d(j) > d(v) + a_{vj}ならば

d(j) = d(v) + a_{vj}

p(j) = v

1) へ

	1	2	3	4	5
1		○			
2	○				
3				○	
4					○
5			○		