大規模データ処理に対する列挙アルゴリズムの活用

宇野 毅明　情報学研究所）

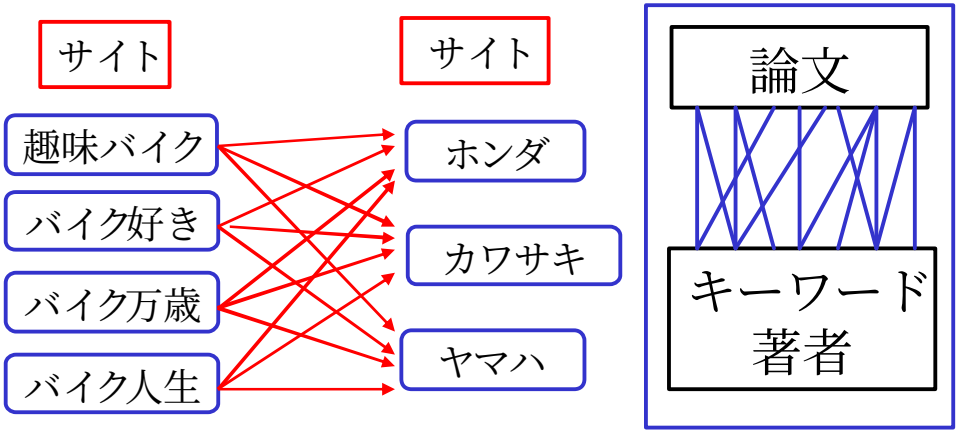
中野 眞一　群馬大学）
房延 慎二　九州大学）
浅井 達哉　富士通研究所）
有村 博紀　北海道大学）

2005年 2月28日 データ工学ワークショップ

巨大なデータからのクラスタ発見

- **90年代以降に巨大な関係グラフが出現**
  - 数百万頂点
  - データベースの関係ある項目を枝で結んだグラフ
  - Web グラフ
- 関連した／似た］項目の集合はグラフのクリークで表される

極大クリークの列挙で、クラスタ発見ができる



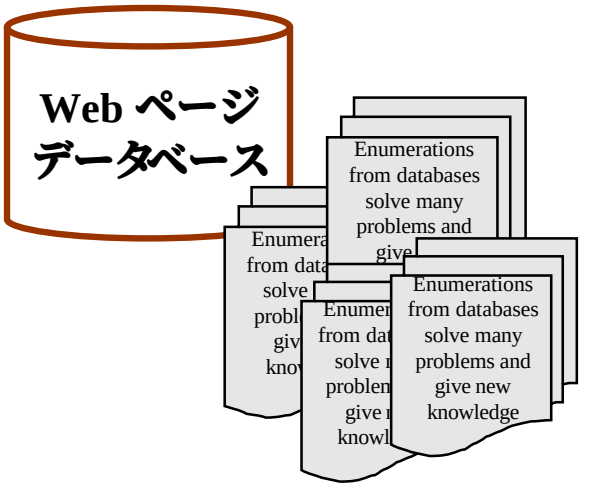
Google で調べると…

- 遺伝子の分類
- 遺伝ネットワーク
- ネットワーク分析
- テキストマイニング
- web コミュニティー

・・・

列挙にかかる時間

- web コミュニティーを列挙する
  ⇒頂点数 **500**万 、枝数 **5000**万 程度



標準的な **PC** で

- 素朴なアルゴリズムだと、**1年** くらい
- 洗練されたアルゴリズムなら、**2時間**程度 [宇野ら02]、[富田ら04]
  (スループット： 秒速**10万** コミュニティー)

いい技術があれば、近似をせずとも
素直な方法で、巨大な問題が短時間で解ける

データマイニング
への応用可能

高速な代表元パターンマイニング
(宇野・有村, DS'04, FIMI'04)

ニュース速報 (Nov. 1, 2004, Brighton, UK)

第2回 FIMI Workshop で宇野・清見・有村組(LCM ver.2) が優勝！
FIMI'04 Best Implementation Award

今年は勝ちました

宇野毅明
(NII, A01班)

- **FIMI Workshop**：
  頻出アイテム集合マイニング実装「(FIMI)」に関する
  データマイニング分野のプログラミングコンテスト
  今年は, 8+5 実装が最終エントリ (問合せ80件超)
- **LCMアルゴリズム**: PPC拡張による出力多項式時間の
  閉アイテム集合列挙アルゴリズム [宇野・有村 DS'04]

データマイニング
への応用可能

動機

モデルが
- シンプルかつ小規模なら、最適化が有効
  （きれいな問題の良質な解を1つ求める）
- 複雑あるいは大規模ならばシミュレーションが有効
  （複雑・大規模な問題の多数の解を見つける）

列挙は中間に位置する

シンプルなモデルをじっくり解く

複雑なモデルを粗く解く

線形計画
局所探索
組合せ最適化

最適化

列挙

良質な解を多数見つける

シミュレーション

アドホックネットワーク
物理現象の計算

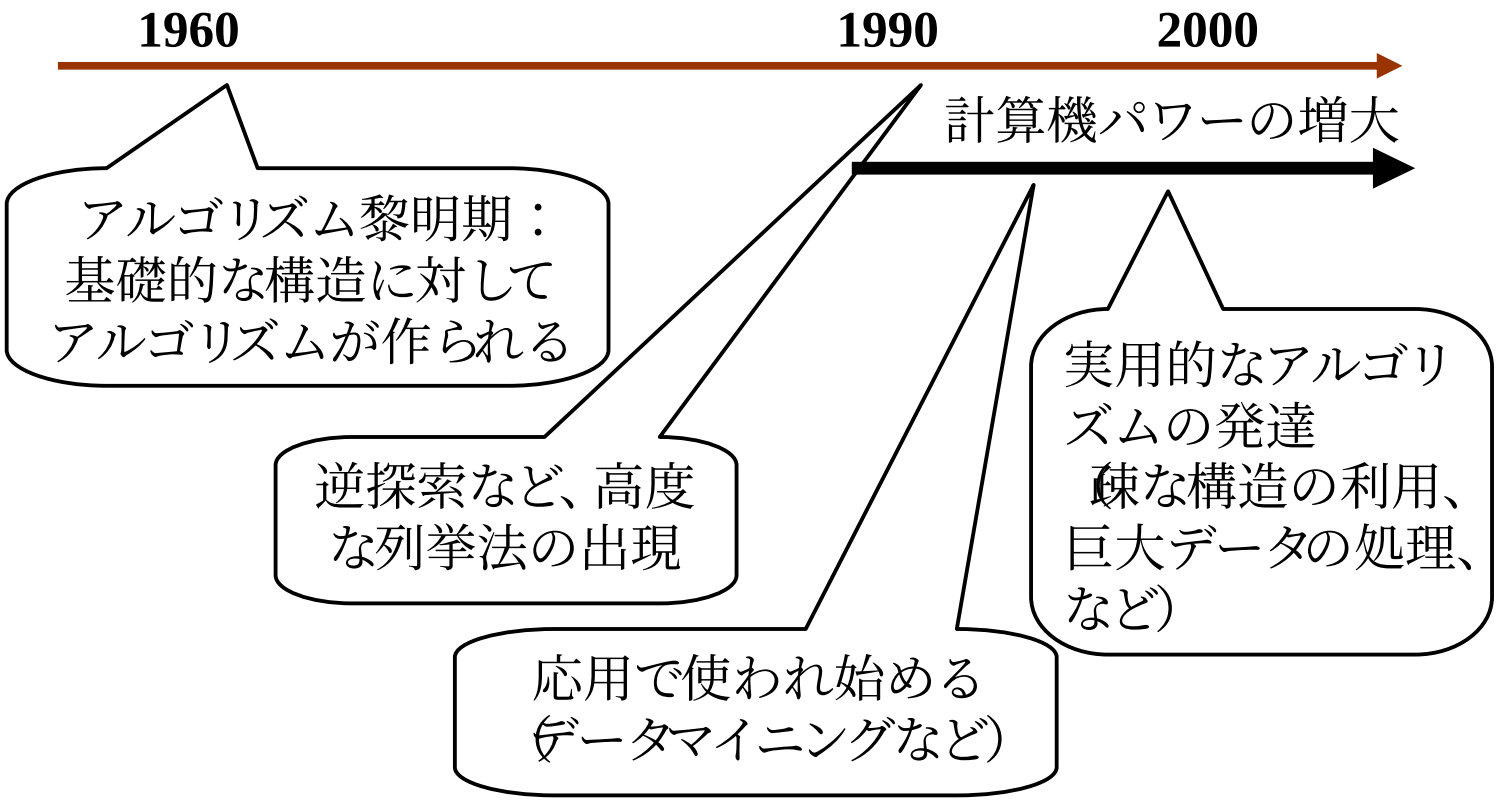
本発表の内容

■列挙問題■

与えられた問題の解を全て見つけ、出力する問題
（いかに役立つ構造を高速に列挙するか）

- 列挙問題と列挙アルゴリズムについて
  - どのような目的で使われるか
  - どのようなアルゴリズムを作りたいか
  - どのような応用研究があるか
  - どのようなアプローチで解けるか

列挙研究の歴史

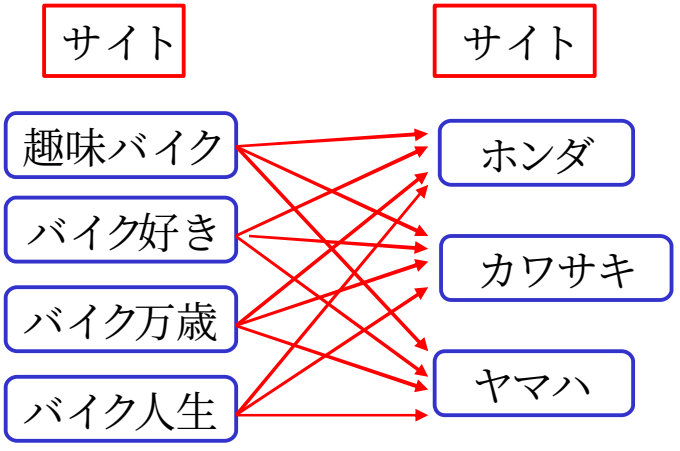

1960
1990
2000
計算機パワーの増大
アルゴリズム黎明期：
基礎的な構造に対して
アルゴリズムが作られる
逆探索など、高度
な列挙法の出現
応用で使われ始める
(データマイニングなど)
実用的なアルゴリ
ズムの発達
(疎な構造の利用、
巨大データの処理、
など)

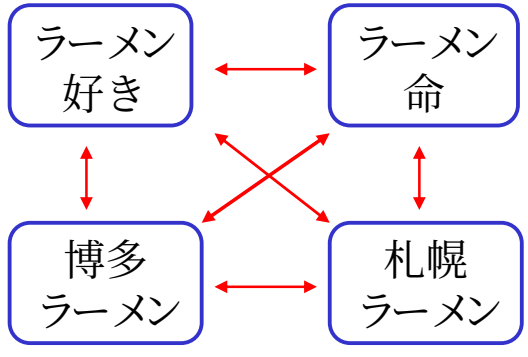
列挙アルゴリズムの応用研究

Web コミュニティ発見

Webコミュニティ：内容や嗜好が似ているweb サイトの集合

モデル：webページ、又はwebサイトの リンク構造からグラフを作る
このグラフの クリーク・2部クリークは、webコミュニティになっている だろう　　(リンクは、似た内容・嗜好のページに貼られるから)





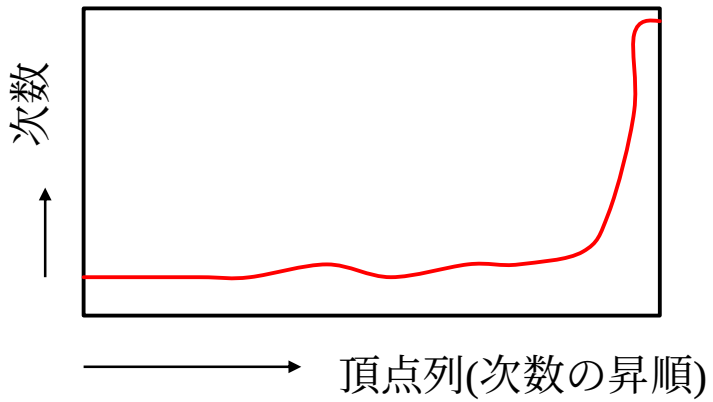
Web マイニングでは基礎的な問題

Web コミュニティ発見 (cond.)

Webグラフ:

- パワー則が成り立つ
- 局所的・大域的に
  密な部分がある

⇒ **極大なクリークは意外と大きく**
その数は意外と少ない



効率良く列挙できる
(秒速 **10**万個。**500**万点でも **2**時間 程度)

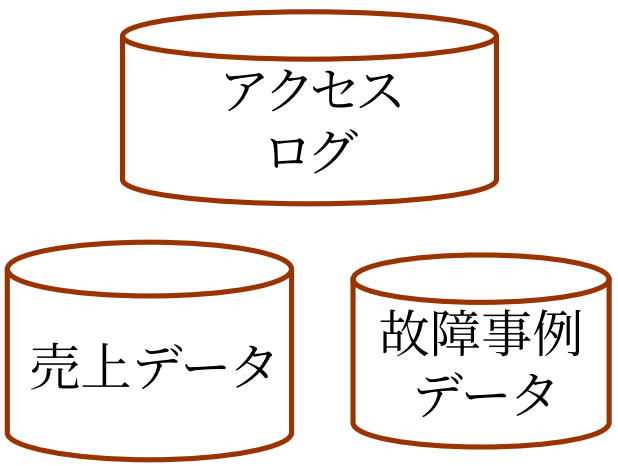
- Kumar、村田(NII)、浅野(東北大)、豊田(東大) など

頻出パターン発見

頻出パターン：

与えられたデータベースの、多くの項目に現れるパターン

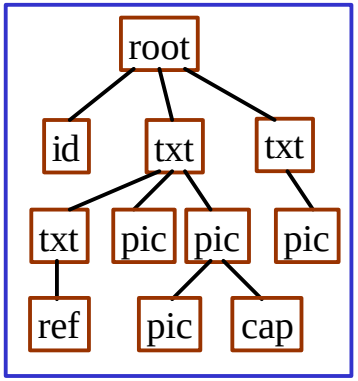
- **これらデータベースの特徴、概要が知りたい**



トランザクション
データベース

1,2,5,6,7
2,3,4,5
1,2,7,8,9
1,7,9
2,7,9

XML
データベース



データマイニングの基礎的な問題

頻出パターン発見　―

データベース

アイテム集合 POS,web-log
グラフ 化合物、関係グラフ、web
木 XML、化合物
文字列 テキスト,ゲノム
時系列データ 観測データ
など

実際のデータは、
大きくて疎なものが多い

良い列挙法と問題の疎性の
利用で、高速な求解が可能

見つけるもの

アイテム集合 :Agrawal、Bayardo
飽和集合 Pasquier、Zaki
極大集合 Han、Zhu
文字列
シークエンス
パス Wang & Liu
順序木 浅井ら、Zaki
無順序木 浅井ら、Termierら、Nijssen & Kok、Ruckert & Kramer,Chi
グラフ 鷲尾、猪口ら、Kuramochi & Karypis、Yan & Han
など

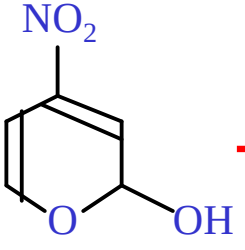
化合物の立体構造の推定

・新たな化合物が得られたときに、その組成は比較的容易に得られるが、結合の構造は容易に計測できず、さらに立体的な構造の計測はもっと難しい

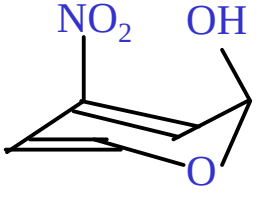
化合物

→ $C_6H_5NO_4$

組成

→ 

平面構造

→ 

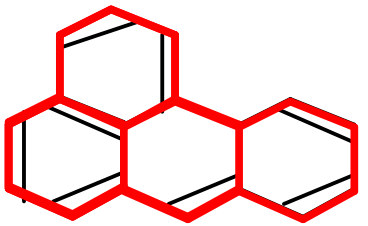
立体構造

すでに構造がわかっている化合物のデータを検索して、構造を推定する

化合物の立体構造の推定 ―

・推定する化合物と部分的な平面構造が一致する化合物をデータベースから探し出す

⇒ 大域的な構造が拾えない



・検索結果を、環構造の複雑さで絞り込む

環構造（コードレスサイクル）の数が似ているもので絞り込むと、精度が上がる

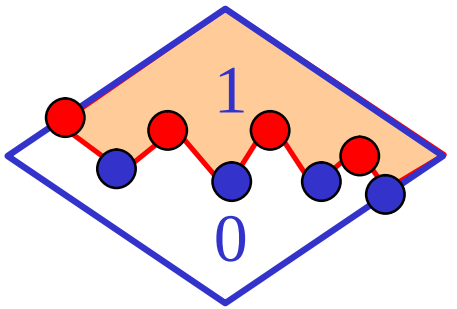
コードレスサイクル数は爆発せず、効率的

・佐藤寛子（NII）& 越野 理（理研）

単調関数の学習

・集合 **E** の部分集合上に定義された**01単調関数 f** があるとする

単調関数：**f(B) = 0** ならば、
B の任意の部分集合に対して
f の値が **0** となるような関数）



問題（双対化）：**f** の極大元（**f(B) = 0** となる極大な部分集合）
全てから、**f** の全ての極小元を列挙（あるいはその逆）
（CNF ⇔ DNF の変換、極小集合被覆・極小横断の列挙と等価）

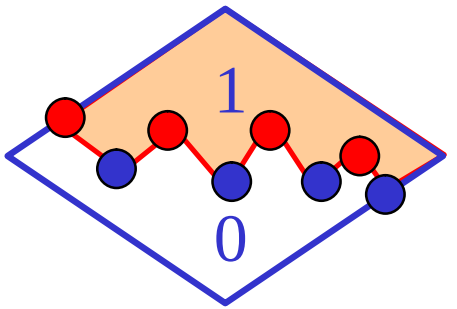
問題（学習）：**f** が陰に与えられたとき、**f** の全ての極大元を列挙

極大頻出アイテム集合マイニング）

単調関数の学習

・集合 **E** の部分集合上に定義された**01単調関数 f** があるとする

単調関数：**f(B) = 0** ならば、
B の任意の部分集合に対して
f の値が **0** となるような関数）



問題（双対化）：**f** の極大元（**f(B) = 0** となる極大な部分集合）
全てから、**f** の全ての極小元を列挙（あるいはその逆）
（CNF ⇔ DNF の変換、極小集合被覆・極小横断の列挙と等価）

問題（学習）：**f** が陰に与えられたとき、**f** の全ての極大元を列挙

極大頻出アイテム集合マイニング）

単調関数の学習　(cond.)

- 双対化 学習は、計算量的に未解決な問題
  - ー　(入力＋出力数)の多項式時間アルゴリズムがあるか不明
  - ーO（(入力＋出力数)$^{(入力+出力数)}$）のアルゴリズムは存在

- 実際には、ほぼ全ての問題は、1つ定数時間で列挙できる
- 「ある種のアルゴリズムを用いると必ず指数時間かかる問題」を作ること自体が難しい

Khachiyan、Eiter、牧野(阪大)…

何が難しいのか、なぜ多項式時間で解けないのか、よくわからない

計算量的にも、効率良い実装上も、面白い問題

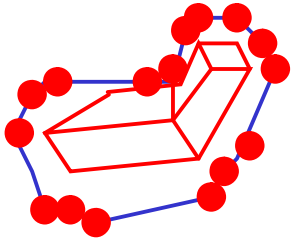
多面体の端点列挙

・線形不等式、あるいは他の陰的な表現で与えられた多面体の端点を列挙する問題
　―多面体上の最適化、性質の解析など

Seidel、Avis、福田

応用例：自動車の各部品が、どの範囲を動けるか調べる
　← 他の部品とぶつかる可能性を調べたい

・どの方向にどれだけ動けるか（回転を含む）を与えると、部品が動きうる範囲が多面体で表される



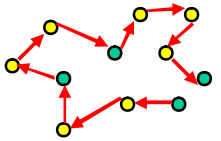
・この多面体の端点を列挙すると、他の部品とぶつかるかどうかが判定できる [福田2004]

列挙アルゴリズム構築法の研究

基本戦略

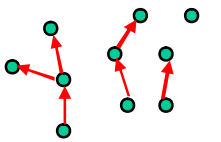
- 列挙は、解を逐次的に見つける

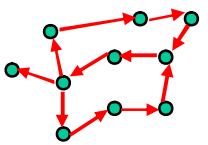
⇒ 列挙は、解空間に効率良い探索ルートを作る問題

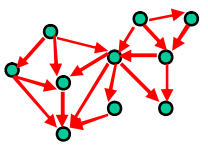


探索ルートは

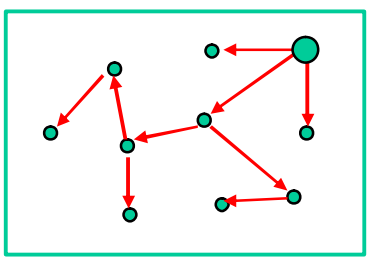
- 解のみを通るものが良い　（効率化）
- 連結が良い　（見つけ損なわないため）
- 非巡回的が良い　（重複を避ける）
- 枝が少ないほうが良い　（重複を避ける＆効率化）





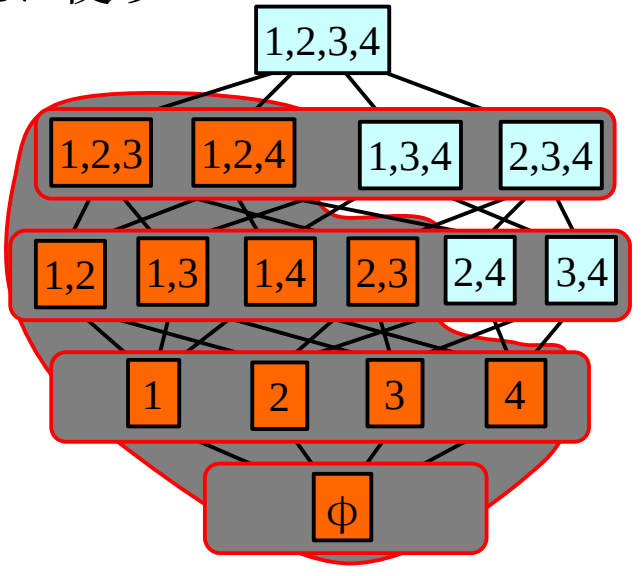


探索ルートは全域木（あるいは全域森）が良い



apriori

- 単調な集合族（独立集合）の列挙などに使う
- 大きさ**k** の解を、
  大きさ**k-1** の解から生成する
- 大きさ**k** の解を全て保持して
  重複を回避



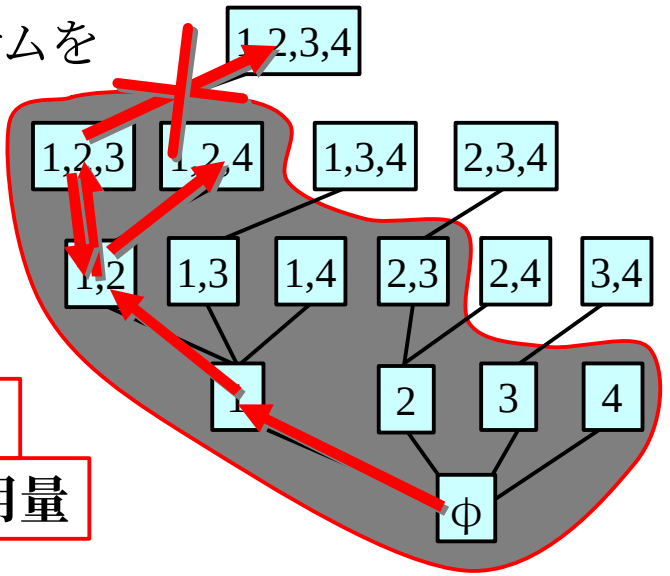
解1つあたり ＝ **1反復の計算時間**

メモリ使用量 ＝ **解集合の大きさ**

データベースの
頻出集合、頻出文字列、頻出シークエンス
など

バックトラック

- 単調な集合族（独立集合）の列挙などに使う
- 空集合から出発し、現在解にアイテムを1つ加えて新しい解を作る
- 現在解の末尾より大きなアイテムのみを追加して重複を回避



解1つあたり ＝ **1反復の計算時間**

メモリ使用量 ＝ **1反復のメモリ使用量**

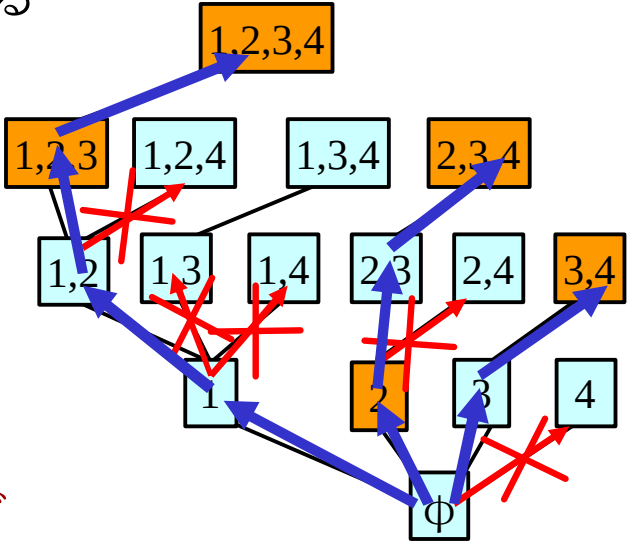
グラフのパス、有向パス、木、根付き木、連結成分
クリーク、独立集合、マッチング、2部マッチング、頂点被覆
ナップサック問題の解、集合被覆　など

枝狩り

- 単調性が成り立たない場合
  解でないものをたどる
- 先に解がないときには枝狩りができる

- 完全な枝狩りができれば、
  解1つあたり
  (木の深さ)×(反復の時間)

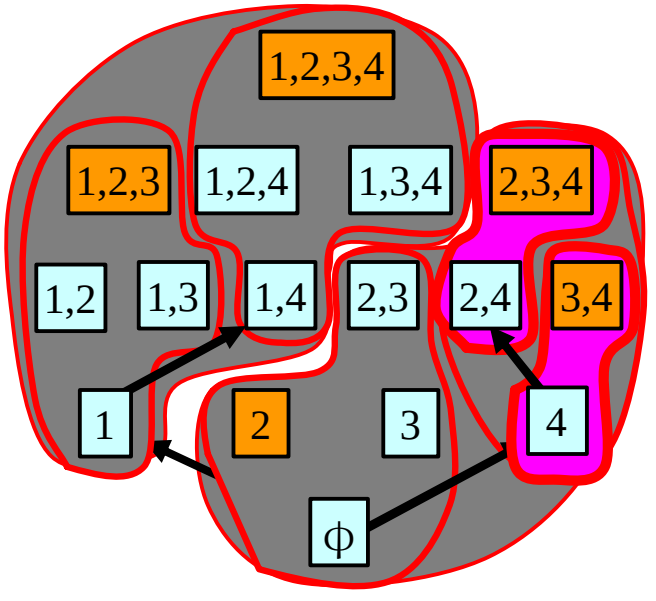
極大頻出集合、極小集合被覆、
極大クリーク、SATの充足解、など



分割法

- 探索木を動的に生成
- 各反復で、解集合を2つの非空な集合に再帰的に分割する

完全な枝狩りを常にしていることに相当する）



解1つあたりの計算時間は
解の数）×（1反復の時間）

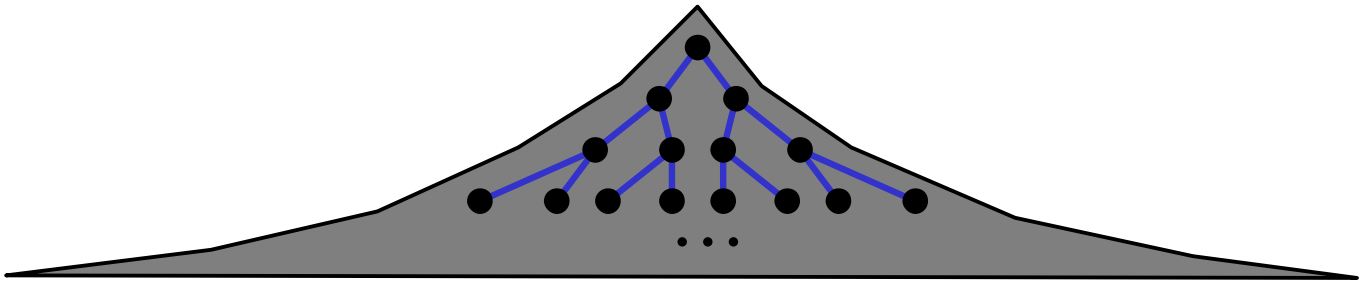
使用メモリは1反復のメモリ使用量

グラフの、パス ・サイクル ・有向パス ・有向サイクル ・木 ・根付き木 ・全張木 ・全域森 ・完全マッチング ・完全2部マッチング　など

実装時の計算速度の向上

- ほぼ全ての列挙アルゴリズムは再帰型
- 各反復で複数の再帰呼び出しをする

⇒ 計算木は、下に行くほど大きくなる



再帰呼び出しの際に問題（入力）を縮約して子供の仕事を軽くすると、
その子孫全てが恩恵を受け、劇的に高速化される

効率良い実装の多くがこの手法を利用（例えばFIMIの多くの実装）

難しい問題

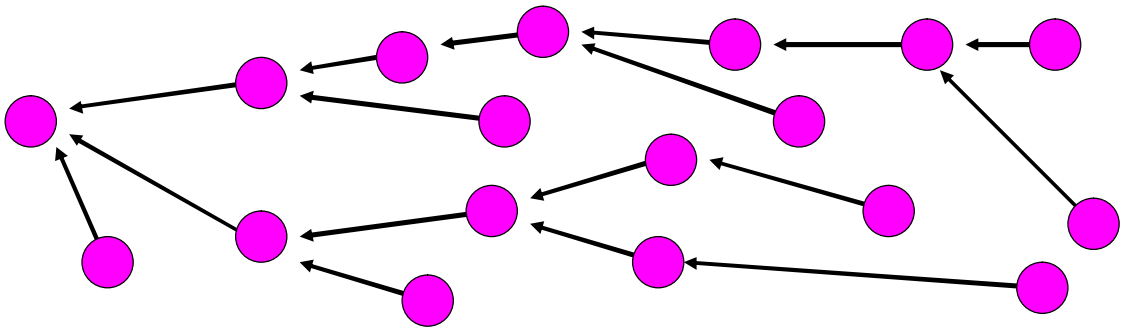
- バックトラック、分割法では難しい問題
  - 単調性が成り立っていない
  - 枝狩りが困難
  - 同型なものが多数ある

例） 極大クリークの枝狩りは NP-complete
  - 指定した複数の頂点を含まない極大クリークは存在するか？

極大／極小なもの、グラフクラス、多面体の面、
NP-complete問題の解

逆探索　（効率良い探索木を直接生成）

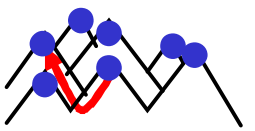
- 各解に対して、その親を非巡回的になるよう定義する



- 木型の探索ルートができる
- この木を深さ優先探索する　⇒　全ての解が列挙できる

解1つあたり　＝　**1反復の計算時間**

メモリ使用量　＝　**1反復のメモリ使用量　＋　木の深さ**

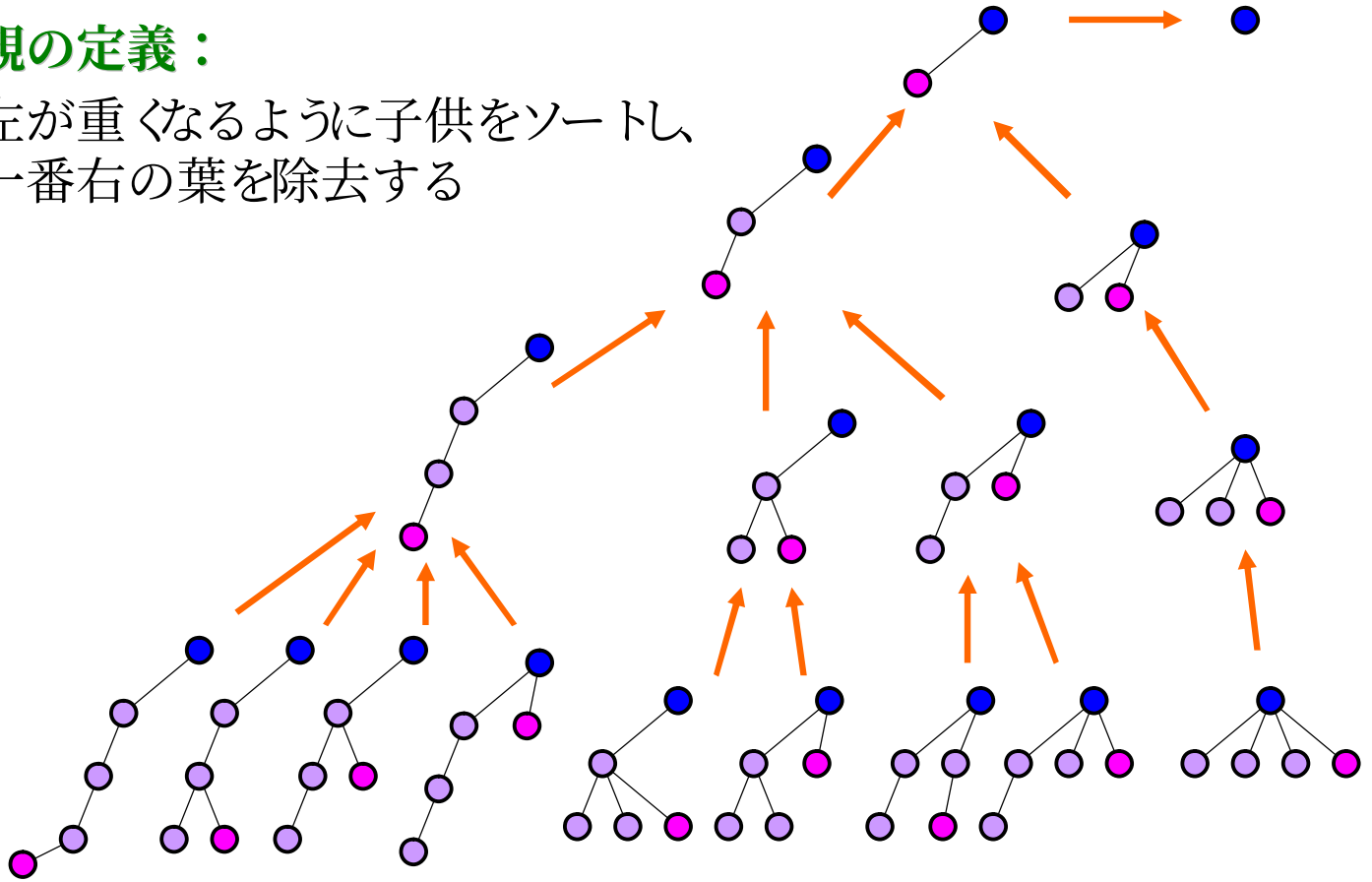


木、根付き木、直並列グラフ、コーダルグラフ、フロアプラン
グラフの極大クリーク、多面体の頂点、三角形分割　など

根付き木

親の定義：

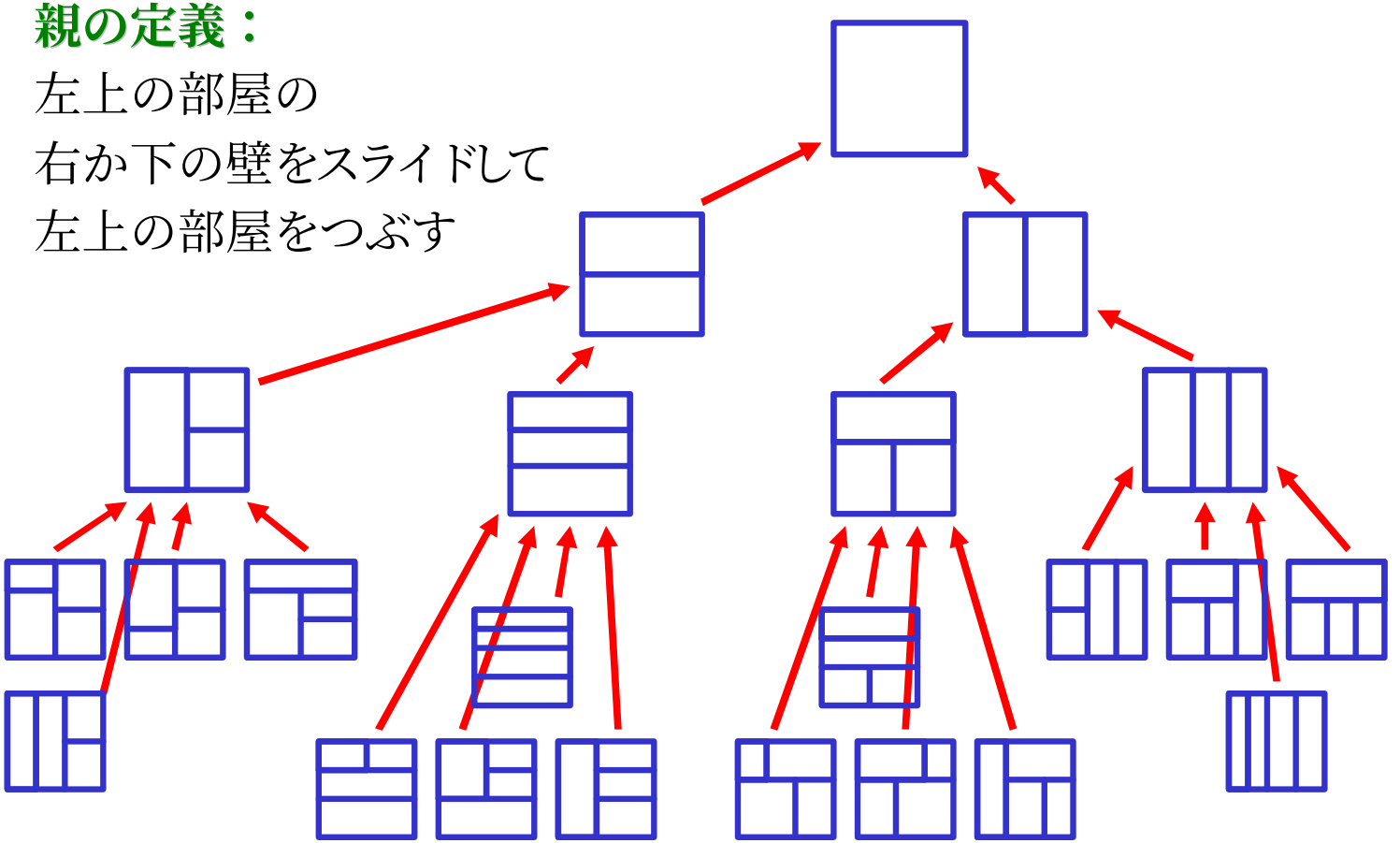
左が重くなるように子供をソートし、
一番右の葉を除去する



フロアプラン　長方形による部屋分け）

親の定義：

左上の部屋の
右か下の壁をスライドして
左上の部屋をつぶす



まとめ

- 列挙問題の応用を紹介
  （クラスタリング、データマイニング、評価値計算、計算幾何学）
- 列挙アルゴリズムの基礎的な手法を解説
  （探索木の構築、バックトラック、枝狩り、逆探索）

- 列挙は、手法の中でまだまだ発展途上
- これから利用価値が高まり、モデル・アルゴリズムともに、
  これから面白いものが出てくるだろう