列挙を用いたモデリングの進展

宇野　毅明

1. はじめに

列挙問題（あるいは単に列挙）とは，与えられた問題の解を全て重複なく出力する問題である．最適化が，与えられた目的関数を最大（最小）化する解を見つけ，システムのある種の極みの状態を見るという，いわば直線的な解析を行うのに対し，列挙は全ての解を見つけるため，問題の構造を多面的に解析していると考えられる．この点は列挙を用いたモデリングを考える上で，重要なポイントになる．

列挙に関する研究は古くからあり，計算機アルゴリズムに関しても1960年代から始まっている．しかし，列挙は通常多くの解を持つため，現実的に意味のある問題の解を実際に列挙するのはそれほど容易ではない．そのため，列挙を用いて現実問題をモデル化して解くというアプローチは，定理証明などを除き，近年になるまであまり行われてこなかった．応用の欠落から，アルゴリズムも理論中心となり，基礎的な構造（グラフの全張木やパスなど）に対して，計算量理論的な結果を出すというものがほとんどであった．

時代が進むにつれ，計算機の速度は飛躍的に向上してきた．1980年ごろから，現実の問題を計算機で解くことが可能になり，最適化問題を中心に多くの問題が解かれてきた．90年代に入ると列挙もモデリングに使われるようになり，例えば多面体の端点を列挙するアルゴリズムなど，最適化で注目されている構造を効率良く列挙するアルゴリズムが研究されてきた．しかし，難しい最適化を網羅的に解くための手段，あるいはシステムの完全性を示すために全ての場合を尽くすなど，いわば直接的な利用が多かった．

一方，90年代からはデータの爆発が始まってきた．IT技術や機械技術が進歩し，データを自動的に収集できるようになったのがその要因の1つである．また，ヒトゲノムプロジェクト，探査機による惑星観測など，組織的に巨大なデータを収集する機会も多くなり，結果としてデータの大きさが飛躍的な増加を始めた．このような巨大データは，人力での解析は難しく，コンピュータ処理の助けを借りることが多くなってきた．

データ収集というものは，手間がかかることが多い．いわゆるOR的なモデリングでは，産業・科学・社会などのシステムから問題を見つけ出し，それを数理的な条件で定式化し，それを解く解法を設計する，という手順をとる．この結果得られたモデルを現実に適用する場合は，データを収集，あるいは収集するシステムを作り，集めたデータを入力として求解し，求めた解を使って実際に運用する，という手順をふむ．このとき，研究でよく重視される求解の部分は，実はそれほどの手間ではなく，むしろデータ収集の部分が最も手間とコストがかかる．電子機器の導入によりデータの収集と管理を自動化すること自体が「IT化」とよばれるゆえんでもあるだろう．

データ収集の難しさと，巨大データの出現，この2つが合わさったときに，データを対象とするモデリングという新しいスタイルが生み出された．つまり，すでにあるデータを対象としてモデリングを行うことで，データ収集のコストを無くすとともに，1つのデータからより多くの知見を導き出そうというものである．このスタイルでは，出発点はデータとそこから得る知見の選別であり，次にモデリングが来る．学術分野では，データマイニング，人工知能，web科学などでこのスタイルがポピュラーであり，産業では，ヤフーなどのweb検索，amazonなどの顧客データからのマーケッティングなどが相当するし，シンクタンクは，まさに既存のデータを収集して再利用していると考えられるだろう．ここでは，このような研究を「データ中心の研究」とよぶことにする．

さて，このように効率良くデータの利用ができそうに見えるデータ中心の研究であるが，実は多くの問題

うの　たけあき
国立情報学研究所
〒101-8430 千代田区一ツ橋2-1-2

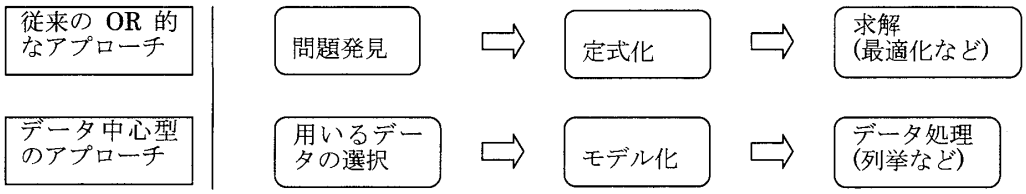

図1　巨大データの出現による問題解決のパラダイムシフト

点を抱えている．それは，データが主に自動的，あるいは散発的に収集されるため，非常に不ぞろいだということである．過去に収集されたデータは，通常ある意思を持って収集されることが多く，その意味でデータは目的を持っていた．その目的に合うように精度は設定され，目的に不向きな項目は削除され，必要な項目は追加された．しかし，例えばゲノムのデータは必然的に読み取りエラーを含み，また，読み取りの実験を行った機関によってデータの精度に違いがある．また，ゲノムそのものも固体による差，種による差を含んでいる．Webのような自立的にネットワークが形成されるようなデータでは，各データの目的が異なり，そのためデータの質・量・詳しさなどが大幅に異なる．中には過失・故意による嘘もある．このようなデータから正確な情報を導き出すことは，簡単ではないだろう．目的のあいまいさも大きな問題である．データ中心の研究では，データから情報を取り出すことを目的とすることが多く，その評価尺度が，「正しい」「特異な」「役に立つ」といった，非常にあいまいで，数理的に表現しにくいものとなりがちである．

このような不ぞろいのデータに対しては，列挙を用いたアプローチが有効である．あいまいな目的には，解候補の列挙という手法が有効である．例えば，webページの検索を考えてみよう．一般にweb上でのデータ検索は，指定した複数のキーワードを含むページを見つけることによって行うが，キーワードの組合せから，望まれるページを唯一に推定することは，おそらく容易ではない．数理モデルを構築し，可能性の高い1ページを表示しても，それが目的のページとは限らないため，最適化的なアプローチは効果が小さいのである．通常は，キーワードを全て含む，あるいは多くを含むページを列挙し，大量に表示する，という方法をとる．これは，ページ検索を「キーワードを含むページを列挙する」という数理的に扱いやすい部分と，「それらの中から望みのページを探し出す」という数理的に扱いづらい部分に分割している，と解釈できる．問題全体の数理モデル化が難しいので，数理的に簡単な部分のみを取り出し，まずその部分の解を列挙してから，なんらかの処理を行い，望まれる解を得ようとしているのである．このような問題の切り分けを行うとき列挙は大きな力を発揮する．

このような列挙の優位さは，不ぞろいのデータ，不正確なデータを扱う場合にも表れる．最適化のように，解を1つしか得ないアプローチは，問題の一部が変化すると解が大きく変化する可能性があり，安定的でない．しかし，列挙の解は問題の構造全体を示すことが多く，変更に対して安定的な解を出す．また，数の多い構造のみを列挙することで，エラーによる特異な構造を除くことも可能である．後に述べる頻出パターン列挙では，この性質を利用した応用が数多く行われている．

列挙を用いたモデル化は，まだそれほど歴史が長いわけではない．そのため，モデル化の手法が系統立てられているわけではなく，直感的な発想を得にくい部分もある．本稿では，近年の代表的な列挙を用いたモデルを紹介する．列挙をモデルに活用する上で，参考となれば幸いである．

2.　クリーク列挙によるグループ発見

データの中の項目間の関係を見るために，関係グラフを構築する手法を考える．例えば，関係ある人物同士，取引のある企業同士，距離が近いもの同士が枝で結ばれたグラフである．逆に，枝の張られているもの同士は似ている，または関係があると思われるグラフがあるとしても良い．このモデルはソーシャルネットワーク，web，リレーショナルデータベース，計算言語学，ゲノム科学など，多くの分野で見られる．

このグラフの頂点の集合で，多くの頂点間に枝が張られている，つまりは密な部分グラフになっているものを見ると，互いに関係が深い，あるいは盛んに何かをしているグループに対応すると考えられる．特に，部分グラフ内の全ての頂点の組が枝で結ばれているとき，これはクリークとよばれ（図2を参照），特徴的なグループ，あるいはその一部分を導くと考えられる．

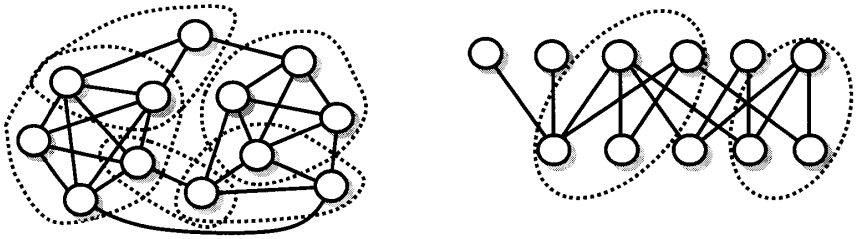

図2　クリークと2部クリークの例：点線で囲われたものがそれぞれ極大クリークと極大2部クリーク（全てではない）左のグラフではクリークは63個あるが，極大クリークは9個である

グラフの中からクリークを見つけることで，グループを見つけることができる，と考えられるだろう．グループ自体は，何かの知識を与えるものであるし，また，グループを組合せることで，より大きなグループを作成し，さらにデータ全体をいくつかのクラスタに分割するようなことも可能である．実際，これは，クラスタリングの基本手法の1つとなっている．

2部グラフの，完全2部グラフとなっている部分グラフは2部クリークとよばれる（図2参照）．2部グラフは，特にデータが2つのグループに分割される場合によく使われる．2部クリークも，クリークと同様に密な部分構造を表すが，2部クリークの場合は，グループ内の関係は背反だが，グループ内は密に関係しているような構造を見つけることができる．例えばwebのニュースサイトとそれらを引用しているサイトはリンクを持つが，ニュースサイト同士，あるいは引用しているサイト同士はリンクがあまりないと考えられる．2部クリークを見つければ，このような構造を自動的に見つけられると考えられる．

このような問題に対して，何らかの意味で評価値の良いクリークを1つ見つける，という手法はあまり効果的でない．理由の1つは，目的関数を数理的に定めることが難しいこと，もう1つはグラフ全体の構造，およびどのようなグループが存在するかを網羅的に調べたいという要求に対して，解を1つだけ見つけても用をなさないからである．このような問題に対しては，列挙的なアプローチが必須である．

一般的に，グラフが密になるとクリークの数は指数的に多くなる．しかし現実に現れるような関係グラフは非常に疎であることが多い．例えばwebネットワークの頂点数は100万，1億といったオーダーになるが，頂点の平均次数は10-20程度といわれており，つまりすかすかなのである．このようなグラフでは，クリークの数はそれほど爆発しない．逆に，疎でないグラフでは，全てのものが密接に関係している，ということになり，そもそもこのような解析する必要がない．そのため，数の爆発は気にせずともよいのである．

たとえグラフが疎でも，局所的に密な部分があるとクリーク数は増大する．例えば頂点数30のクリークは，その中に部分グラフとして10億以上のクリークを含むため，1つ大き目のクリークが存在するだけで，クリークの列挙は困難になる．そこで，包含関係において極大なクリークだけに注目する，というモデル化が行われる．任意のクリークはある極大クリークに含まれるので情報的な損失はなく，しかもグラフに局所的に密な部分が存在しても，極大クリークの数はそれほど大きくならないのである．例えば，webネットワークでは，極大クリークの数はグラフの大きさに対して10から100倍程度ということが実験的に知られている．

クリーク列挙に対しては，ほぼ自明で，効率の良いアルゴリズムがある．クリークの任意の部分集合はクリークであるため，空集合から出発して深さ優先的に探索を行い，全てのクリークを見つけるのである．各反復では再帰的に頂点を加えるのであるが，このとき前回加えた頂点よりも添え字が大きな頂点のみを加えるようにすると，重複を自然に回避できる．この手法はバックトラック法とよばれ，列挙アルゴリズムにおいて基本的である．また，常に「クリークに追加できる頂点」の集合を保持することで，各反復で追加する頂点を効率良く見つけることができる．ある頂点をクリークに追加したとき，クリークに追加できる頂点の集合の中で追加後もクリークに追加できるものは，その頂点に隣接するもののみであるため，各反復での実質的な作業は「追加できる頂点の集合」の中から「追加する頂点に隣接するもの」を選び出す，絞り込み検索になる．これは，グラフの頂点の次数が平均的に小さければ，きわめて短時間で行えるため，疎なグラフのクリーク列挙は短時間でできるのである．極大なクリークの列挙を行う場合には，クリーク列挙に枝刈り

の操作を追加するか[11]，あるいは極大クリーク間に隣接性を定義して，その上で深さ優先探索を行えばよい[7][12]．どちらも疎なグラフに対して高速であるが，特に前者は密なグラフに対しても高速である．

クリークは構造が単純であり，列挙も容易である．しかし，もともとの目的は密な部分グラフを見つけることであり，その意味では本来の目的から少し外れている．そこで，もう少し計算コストをかけ，より目的に近いもの，つまり密な部分グラフを直接見つけようというアプローチもある．この場合，密な部分グラフをどのように定義するかにより問題が変わってくる．

1つ目は，密な部分グラフを，クリークから定数本の枝を抜いたもので定義する．利点は，密な部分グラフの任意の部分グラフは密な部分グラフになるため，バックトラック法と枝刈により，効率的に極大なものが列挙できることである．しかし，大きなクリークと小さなクリーク，両者から同じだけ枝を抜いたものを考えれば，明らかに前者のほうがより密であるとみなせるため，クリークから抜く枝の本数の決め方で，見つかる密な部分グラフの大きさが大きく変化してしまう．これはこの定義の短所である．また，頂点数の小さい部分グラフのほぼ全てが密な部分グラフになってしまうという欠点もある．

2つ目は，枝の割合を使うものである．クリークからある定数割合の枝を取り除いたものを密なグラフとして定義する．利点は，直感的に密と思われる部分グラフを大きさによらず見つけられること，および頂点数の小さい部分グラフで密になるものはそれほど多くならないということである．欠点は単調性が成り立たないこと，つまり，密な部分グラフの部分グラフは密になるとは限らない点である．そのため，列挙を行うには多少工夫が必要であり，また，極大な密な部分グラフの定義が自然な形で行えなくなる．

クリークは，いわゆる情報学の分野では非常に普遍的な構造であり，日本を含め世界的に多くの研究がある．参考文献探しには困らないであろう．

3. 頻出するパターンの発見

データベースの中に多く現れる（頻出する）パターンを見つけるという問題は，データマイニングの分野で中心的な問題である．頻出するパターンは，データベースを特徴付けるものであると同時に，それ自体が何らかの意味を持つからである．頻出するパターンを列挙することで，知識発見やルール生成が行えるのに加え，成功事例と失敗事例を区別するパターンの発見，といった目的にも使える．また，頻出するパターンのみを見つけるということは，ノイズのような特異なパターンを取り除く働きがあるため，2つのデータベースがどの程度類似しているかを測定するために用いられることもある[1]．

例として，コンビニエンスストアなど小売店の売り上げデータを見てみよう．図3を見てほしい．小売店では，レジで会計をする際に，各顧客がどのような品物をいくつ購入したか，データが自動的に収集される．このようにして集められたデータは，データの各項目が，商品の集合，という集合の部分集合（重複を無視すれば）になっている．ここで，商品の部分集合をパターンと定義し，あるパターンがある項目の部分集合になっている（含まれる）とき，その項目にそのパターンが現れたと定義する．あるパターンがデータベースに頻出するとは，多くの項目に含まれること，厳密には，ある閾値よりも多くの項目に含まれるパターンを頻出パターンとする．頻出パターン（パターンが部分集合である場合は頻出集合とよばれる）は多くの顧

客A	おにぎり，弁当，お茶，ヨーグルト
客B	雑誌，おにぎり
客C	弁当，お茶，ヨーグルト
客D	お茶，おにぎり，雑誌
客E	弁当，おにぎり，雑誌，お茶
客F	雑誌
客G	おにぎり
客H	おにぎり，ヨーグルト
客I	雑誌，ヨーグルト
客J	ヨーグルト，弁当，お茶

3人以上の客が買った組合せ
(×人数) (*は飽和パターン)

*{雑誌，おにぎり}×3
{ヨーグルト，弁当}×3
{ヨーグルト，お茶}×3
*{お茶，弁当}×4
*{お茶，弁当，ヨーグルト}×3

図3　小売店の売り上げデータと，良く買われる（頻出する）品目の組合せ

客に共通して買われているものであるので，販売戦略，店舗のレイアウト，勤務体制などを考える上で重要なデータとなる．米国の大規模小売店がこのような分析(バスケット分析とよばれる）を行って，ビールと紙おむつが同時に買われる事が多いという発見をした，という逸話は有名である．

頻出パターンを使った問題解決で最も有名なものは，ルール発見であろう．上記の小売店の例であれば，頻出集合から，お茶とお弁当を買うお客はヨーグルトを買う可能性が高い，といった推測のルールを探し出すものである．こういったルールは，仮定の部分がある程度普遍的である必要があるため，頻出パターンであることが望ましい．また，データの分類規則を見つけるためにも用いられる．オフィス街の小売店の売り上げと住宅地の小売店のそれを特徴付けるべく，片方のグループによく現れ，もう片方にはあまり現れないようなパターンを見つけるために頻出パターン列挙が使われる．また，頻出パターン自体を統計量として用いることもある．小売店同士の売り上げの様子を比較するため，売り上げデータを比較するとき，データの大きさ，品目などが異なると，直接比較することは難しいが，10%以上の項目に現れるパターンを統計量として用いることで，大きさや細かい品目の違いを吸収し，そのまま比較することが可能となる．また，頻出パターンを用いることで，特異な，ノイズのような項目が自動的に排除される上に，組合せ的な構造を比較することができるという利点がある．

このような，各項目が集合の部分集合になっているようなデータベースは，トランザクションデータベースとよばれる．上記の小売店の販売データの他，webリンク，文書の単語ベクトル，アンケートデータ，実験結果など，多くのデータがトランザクションデータベースとなっており，関連の深い言葉の集合，意見・嗜好の組合せ，興味対象が似通ったページの集合，類似する書籍など，多種にわたる情報を得られる．

webや有機化学物質，あるいは関係ネットワークのデータベースは，各頂点，あるいは各枝にラベルのついたグラフであるとみなせる．このようなデータベースに対しては，ラベルつきの部分グラフや有向部分グラフがパターンとなる[6]．また，XMLデータや組織図などのデータは，各頂点にラベルがついた木構造ネットワークであるとみなせる．これらには，ラベルつきの木構造がパターンとして有効である[3][4]．ゲノムデータ，文書データは文字列のデータであり，文字の並び，単語の並びなどがパターンとなる[2]．このように，多岐にわたるデータベースに対して，頻出パターン発見問題が適用可能であり，実際に多くの事例がある．

頻出パターン列挙は，どの程度以上現れるものを見つけるか，という閾値を変化させることで，解の数を調整することができる．解が少なければ，閾値を下げてより頻出度の低い解を探索してみることができ，解が多すぎて計算が終了しないようであれば，閾値を上げて頻出度が高いもののみを見つけるようにすればよい．このため，現実問題への適用性が高い．また，極大なパターンのみを見つけることで解の数を減少させることもできる．ただし，極大な解は閾値に依存して大きく変化するため，あまり安定的ではないという欠点も持つ．最近は，含まれる項目が同じパターンは意味的に等しいとみなし，その中から代表として極大なもの（飽和パターンという）のみを列挙する，というアプローチも研究されている[2][14]．

あるパターンが含まれる項目の数を計算するには，基本的に全ての項目を調べる必要がある．そのため，素朴に作ったアルゴリズムでは非常に計算時間がかかる．しかし，クリークと同じようにバックトラック法を使用し，追加できる項目と含まれる項目を絞り込みながら保持していくことで，再帰的に計算時間を短縮することができる．列挙アルゴリズムの計算構造は先が指数的に広がった木構造をしており，先端部分を高速化するだけで全体の計算時間を短くできる（末広がり性とよぶ）．そのため，様々な種類のパターンが速く列挙できるのである．

頻出パターンに関する文献は数多く，web検索などで容易に見つけることができる．頻出集合についてはFIMIというレポジトリサイトがあり，そこに主なアルゴリズムの論文・コード，ベンチマークデータと実験結果がある[5]．

4. 部分グラフ列挙

グラフの部分グラフの中で，ある種の性質を満たすもの（木である，パスである，2部グラフである，等）を全て見つける問題を部分グラフ列挙問題という．列挙アルゴリズムの研究においては，この種の問題は非常にポピュラーであり，多くの研究が行われてきた．研究が主に数理計画の一部として行われてきたことなどから，分枝限定法の基礎，あるいは難しい問題に対する解の全列挙といった応用が中心であった．通常こ

の種の問題の解は非常に多く，現実的には小さな問題しか解けない．しかし，解の多さは，極大・極小解のみ（コードレスサイクルなど）を列挙する，あるいは何らかの意味で評価地の高いもの（全張木の代わりに最小全張木，パスの代わりに最短パスなど）のみを列挙する，ということで，数の爆発を避けることができる．

近年，列挙して得た解を統計量として利用するアプローチが行われてきている．[10]では，化合物のデータを特徴付けるために，化合物の結合グラフに含まれるコードレスサイクル（ショートカットを持たない，いわば極小なサイクル）を長さでいくつかのグループに分けて総数を計算し，それを化合物の統計量としている．類似する化合物の検索にこの統計量を使うことで，化合物の環構造を考慮に入れることができ，似た性質を持つ化合物を効率良く見つけることに成功している．

5. 終りに

本稿では，近年盛んに行われている列挙を用いたモデリングの一例について紹介をした．列挙を用いたモデリングは，比較的OR関係以外の分野で見かけることが多い．近年の巨大なデータ，複雑なモデルを扱う上で列挙は強力なツールであり，OR関係の研究でもより一層使われることを期待している．本稿が列挙モデルの理解の助けになれば幸いである．

参考文献

[1] A. Apostolico, M. Comin and L. Parida: "Mining, compressing and classifying with extensible motifs," Algorithms for Molecular Biology, 14 (2006).

[2] H. Arimura and T. Uno: "Polynomial Space and Polynomial Delay Algorithm for Enumeration of Maximal Motifs in a Sequence," LNCS 3827, (2005), pp. 724-737.

[3] T. Asai, H. Arimura, T. Uno and S. Nakano: "Discovering Frequent Substructures in Large Unordered Trees," LNAI 2843, (2003), pp. 47-61.

[4] T. Asai, K. Abe, S. Kawasoe, H. Arimura, H. Sakamoto and S. Arikawa: "Efficient Substructure Discovery from Large Semi-structured Data," SDM 2002.

[5] B. Goethals: the FIMI repository, http://fimi.cs.helsinki.fi/, 2003

[6] 猪口明博, 鷲尾隆, 元田浩: "多頻度グラフマイニング手法の一般化," 人工知能学会論文誌, Vol. 19, No. 5 (2004), pp. 368-378.

[7] K. Makino and T. Uno: "New Algorithms for Enumerating All Maximal Cliques," LNCS 3111, Springer, (2004), pp. 260-272.

[8] S. Nakano and T. Uno: "Generating Colored Trees," LNCS 3787, (2005), pp. 249-260.

[9] 応用数理計画ハンドブック, 朝倉書店.

[10] H. Satoh, H. Koshino, T. Uno, S. Koichi, S. Iwata and T. Nakata: "Effective consideration of ring structures in CAST/CNMR for highly accurate 13 C NMR chemical shift prediction," Tetrahedron 61, (2005), pp. 7431-7437.

[11] E. Tomita and T. Seki: "An Efficient Branch-and-Bound Algorithm for Finding a Maximum Clique," DMTCS'03, LNCS 2731, (2003), pp. 278-289.

[12] 宇野毅明: "大規模グラフに対する高速クリーク列挙アルゴリズム," 電子情報通信学会技術研究報告, Vol. 103, No. 31 (COMP 2003 1-8) (2003), pp. 55-62.

[13] 宇野毅明: "コードレスサイクルを列挙する線形時間アルゴリズム," 情報処理学会アルゴリズム研究会 92 (2003), pp. 47-53.

[14] T. Uno, T. Asai, Y. Uchida and H. Arimura: "An Efficient Algorithm for Enumerating Closed Patterns in Transaction Databases," LNAI 3245, (2004), pp. 16-31.

[15] 宇野毅明のホームページ: http://research.nii.ac.jp/~uno/index-j.html