集約度制約付き飽和系列パターンマイニング

Intensity Constraint Based Closed Sequential Pattern Mining

武　井　宏　将

要　約　本稿は，集約度制約付き飽和系列パターンマイニング（intensity constraint based closed sequential pattern mining）について述べる．系列パターンマイニング（sequential pattern mining）とは，大量のアイテム列から頻出する部分アイテム列パターンを抽出するマイニング手法である．系列パターンマイニングで抽出されるパターンの中で，出現回数が同じであり，他のアイテム列の部分集合とならないパターンのみを抽出するマイニング手法を飽和系列パターンマイニング（closed sequential pattern mining）と呼ぶ．系列パターンマイニングや飽和系列パターンマイニングは，発生頻度の低いアイテムを含むパターンを抽出することが難しいという問題がある．この問題に対して，集約度制約を付けた飽和系列パターンマイニングのためのアルゴリズムを開発し，この問題を解決した．コーディングパターン抽出問題を用いた実験により，集約度制約付き飽和系列パターンマイニングの有効性を示す．

Abstract　In this paper, a mining method called intensity constraint based closed sequential pattern mining is discussed. Sequential pattern mining is a mining method which extracts frequent sequential patterns from massive sequence data. Moreover closed sequential pattern mining is the mining method that extracts sequential patterns except sequential patterns have same frequent super sequential pattern. Sequential pattern mining and closed sequential pattern mining have a problem that these methods have difficulty in extracting sequential pattern including the low frequency item. To solve above problem we developed the algorithm of intensity constraint based closed sequential pattern mining. The result of experiment which utilized the coding pattern extraction problem showed the effectiveness of this method.

1. は　じ　め　に

私たちの周りでは，日々大量のデジタルデータが生み出されている．多くの企業は，大量のデジタルデータを抱えているが，これらのデータの活用は中々進んでいない．蓄積された大量のデータを用いることで，これまで得ることができなかった新たな知見が得られるのではないかという期待が高まっており，大量のデータから情報を抽出する手法として，データマイニングが注目されている．データマイニングとは，大量に蓄積されるデータを解析して，その中に潜むアイテム間の相関関係やパターンなどを探し出す手法である．本稿は，アイテム列に対するマイニングを扱う．アイテム列とは，順序付けされたアイテムの列を指す．アイテム列の集合に対するマイニング手法として，系列パターンマイニング（sequential pattern mining）が知られており，これまで多くの研究がされている．系列パターンマイニングとは，大量のアイテム列集合から頻出する部分アイテム列のパターンを抽出するマイニング手法である．さらに，系列パターンマイニングで抽出されるパターンの中で，出現回数が同じであり，他のアイ

テム列の部分集合とならないパターンのみを抽出するマイニング手法を飽和系列パターンマイニング（closed sequential pattern mining）と呼ぶ．系列パターンマイニングや飽和系列パターンマイニングには，発生頻度の低いアイテムを含むパターンを抽出することが難しいという問題があるが，集約度制約を付けた飽和系列パターンマイニング手法を開発し，これを解決した．

本稿は以下の構成をとる．2章において，系列パターンマイニングと飽和系列パターンマイニングについて述べる．3章において，系列パターンマイニングと飽和系列パターンマイニングの問題点について述べ，その問題点を解決する手法として開発した集約度制約付き飽和系列パターンマイニングについて述べる．4章において，コーディングパターン抽出問題に対して集約度制約付き飽和系列パターンマイニングを適用した実験結果について考察し，5章でまとめる．

2. 系列パターンマイニング/飽和系列パターンマイニング

本章では，系列パターンマイニングと飽和系列パターンマイニングについて述べる．アイテム列 s とは，順序付けされたアイテムの列とする．また，アイテム列を $s = <s_1, s_2, \ldots, s_N>$（ただし $s_i (1 \leq i \leq N)$ はアイテム）とする．このようなアイテム列の集合をアイテム列データベースと呼び，アイテム列データベースに含まれる各アイテム列をトランザクションと呼ぶ．アイテム列データベースは様々な場面で現れる．ここで，顧客の購買履歴を例としてアイテム列データベースについて述べる．

四人のユーザについて，以下のような購買履歴が得られたとする．

ユーザ A：コンピュータ A → 増設用メモリ → キーボード

ユーザ B：コンピュータ B → キーボード → USB メモリ

ユーザ C：コンピュータ A → 増設用メモリ → USB メモリ

ユーザ D：コンピュータ A → モニターケーブル → 増設用メモリ

このとき，ユーザが購入した商品をアイテムとすると，表1のようなアイテム列データベースが得られる．

表1　購買履歴から得られたアイテム列データベース

A	<コンピュータA，増設用メモリ，キーボード>
B	<コンピュータB，キーボード，USBメモリ>
C	<コンピュータA，増設用メモリ，USBメモリ>
D	<コンピュータA，モニターケーブル，増設用メモリ>

このアイテム列データベースを解析すると，<コンピュータ A, 増設用メモリ> という部分アイテム列が頻出していることがわかる．これは，コンピュータ A を購入したユーザは購入後に増設メモリを購入していることを意味している．このことから，コンピュータ A のメモリが十分でない可能性が推測される．このように，購買履歴からアイテム列データベースを作成し解析することで，ユーザの商品購入パターンを得ることができる．また，これらのパターンと他の現象を組み合わせることで，現象の因果関係の解析に利用することもできる．

次に，系列パターンマイニングについて述べる．系列パターンマイニングは，1995 年に IBM 研究所の R. Agrawal と R. Srikant によって提唱された[1]．系列パターンマイニングとは，アイテム列データベース S と最小出現回数 *min_sup* を与えて，*min_sup* 回以上のトランザク

ションで出現する部分アイテム列を抽出するマイニング手法である．系列パターンマイニングが適用される場面は多岐に渡り，例えば，顧客の購買履歴の解析，ウェブアクセスパターンの解析，医療データの解析などに適用されている．

表2の例を用いて，系列パターンマイニングについて説明する．最小出現回数を3とすると，<a, c>，<a>，<c> という三つの部分アイテム列がパターンとして抽出される．

表2　アイテム列データベースの例

s1	<a，b，c>
s2	<a，d，e，c>
s3	<d，b，a，c>

系列パターンマイニングにおいて，抽出されたパターンの部分アイテム列は必ずパターンとして抽出される．表2の例においては，パターン <a, c> の部分アイテム列 <a> および <c> もパターンとして抽出されている．長さ n のパターンの部分アイテム列の数は $2^n - 1$ 通りあるため，系列パターンマイニングで抽出されるパターンが膨大な数となってしまう場合がある．しかし，この例においてパターン <a> やパターン <c> はアイテム列 <a, c> がパターンとして抽出されていれば，抽出しなくても問題のないパターンであるとみることができる．そこで，系列パターンマイニングで抽出されるパターンの中で，出現回数が同じであり，他のアイテム列の部分集合とならないパターンのみを抽出する飽和系列パターンマイニング（closed sequential pattern mining）が知られている．表2の例において最小出現回数を3として系列パターンマイニングを適用すると，<a>，<c>，<a, c> は出現回数が3であり，<a> および <c> はアイテム列 <a, c> の部分集合であることから，表2の例に飽和系列パターンマイニングを適用すると <a, c> のみが出力される．飽和系列パターンマイニングを用いることで，系列パターンマイニングを用いる場合と比較して抽出される系列パターン数を抑えることができる．

系列パターンマイニングの手法として，PrefixSpan アルゴリズム[2]がよく知られている．また，飽和系列パターンマイニングの手法として，BIDE アルゴリズム[3]がよく知られている．

3.　集約度制約付き飽和系列パターンマイニング

本章では，3.1節で系列パターンマイニングや飽和系列パターンマイニングの問題点について述べ，3.2節で集約度制約付き飽和系列パターンマイニングについて述べる．

3.1　系列パターンマイニング/飽和系列パターンマイニングの問題点

系列パターンマイニングや飽和系列パターンマイニングの問題点として，発生頻度の低いアイテムを含むパターンを抽出することが難しく，一方で，出現回数が多いアイテムは同時に呼び出される頻度が高くない組であってもパターンとして抽出されてしまう場合があることが挙げられる．この問題は，系列パターンマイニングや飽和系列パターンマイニングが最小出現回数を閾値としてパターンを抽出することに起因する．例として，アイテム a の出現回数が100，アイテム b の出現回数が100，アイテム列 <a, b> の出現回数が5の場合と，アイテム c の出現回数が5，アイテム d の出現回数が5，アイテム列 <c, d> の出現回数が4の場合を比較する（図1）．アイテム a およびアイテム b の出現回数は大きいが，アイテム a およびアイテ

ム b の出現回数と比較してアイテム列 <a, b> の出現回数が小さい例である．一方，アイテム c およびアイテム d の出現回数は小さいが，アイテム c およびアイテム d の出現回数と比較してアイテム列 <c, d> の出現回数は大きい例となっている．

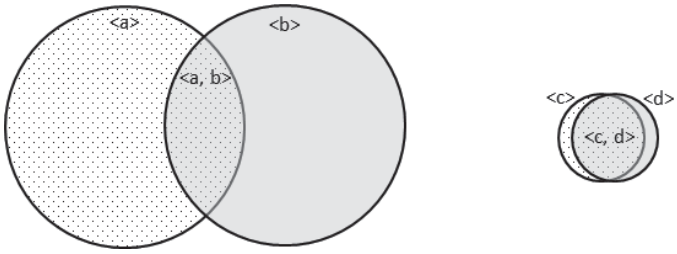

図 1　系列パターンマイニングで問題となる例

これらの例について，最小出現回数を 5 として系列パターンマイニングを適用する．このとき，部分アイテム列 <a, b> は抽出されるが，部分アイテム列 <c, d> は抽出されない．系列パターンマイニングは出現回数を閾値としているため，アイテムの発生頻度が大きいほどパターンに含まれやすくなり，パターン内のアイテムが同時に発生する頻度は考慮されない．アイテム列 <a, b> はアイテム a またはアイテム b が出現するトランザクションのうち，5/195 ≅ 0.025 でしか出現しないにも関わらずパターンとして抽出され，一方，アイテム列 <c, d> はアイテム c またはアイテム d が出現するトランザクションのうち，4/6 ≅ 0.667 で出現しているにも関わらずパターンとして抽出されない．

アイテム <c, d> のような出現回数が少ないパターンを抽出するためには，最小出現回数の値を小さく設定すればよい．しかし，系列パターンマイニングアルゴリズムや飽和系列パターンマイニングアルゴリズムにおける一般的な問題として，最小出現回数を小さくすると抽出されるパターン数や処理時間が大幅に増大することがある．図 2 は BIDE アルゴリズムを用いた飽和系列パターンマイニングについて，最小出現回数を変えた場合の抽出されるパターン数および処理時間のグラフを表している．これらのグラフから最小出現回数を小さくするとパターン数および処理時間が指数的に増加することがわかる．この傾向は系列パターンマイニングにおいても同様である．

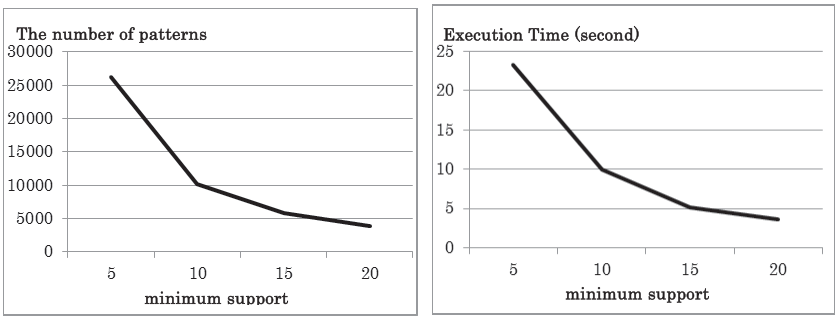

図 2　BIDE アルゴリズムの処理時間とパターン数（左：パターン数，右：処理時間）

抽出されたパターンを利用することを考えると，数千・数万のパターンが抽出されることは望ましくない．また，処理時間の増大の問題もあり，最小出現回数として小さな値を設定して系列パターンマイニングアルゴリズムや飽和系列パターンマイニングアルゴリズムを適用することは困難である．

系列パターンマイニングに対して，ある特定の制約を課し，その制約を満たす部分アイテム列を抽出するマイニング手法を制約付き系列パターンマイニングと呼ぶ．制約を用いることで，抽出パターン数の絞り込みや処理時間の削減が可能となる．ここで，部分アイテム列の少なくとも一つのアイテムを含むトランザクションのうち，部分アイテム列全体を含むトランザクションの割合を集約度として定義する（定義の詳細は 3. 2 節で述べる）．集約度を制約とすることで，より小さな最小出現回数を設定することが可能となる．例であげたアイテム列 <a, b> のようなパターンは集約度の制約により抽出されなくなる．一方，アイテム列 <c, d> のようなパターンは最小出現回数を小さく設定することで抽出することが可能となる．

3. 2　集約度制約付き飽和系列パターンマイニング

本節では，はじめに集約度制約を定義し，次に集約度制約付き飽和系列パターンマイニングについて述べる．

アイテム列データベースを SDB とし，min_sup を最小出現回数，$\gamma(0 \le \gamma \le 1)$ を最小集約度とする．$\alpha = <a_1, \ldots, a_N>$ をアイテム列とするとき，$S_{a_i}(1 \le i \le \mathrm{N})$ はアイテム $a_i(1 \le i \le \mathrm{N})$ を含むトランザクションの集合とする．つまり，

$$S_{a_i} = \{s \mid s \in SDB, a_i \in s\}$$

と定義する．

また，$SC_0(\alpha)$ と $SC_1(\alpha)$ を以下のように定義する．

$$SC_0(\alpha) = \bigcup_{i=1}^{N} S_{a_i} \qquad (1)$$

$$SC_1(\alpha) = \{s \mid s \in SDB, a \subseteq s\} \qquad (2)$$

つまり，$SC_0(\alpha)$ はアイテム列 α に含まれるアイテムを少なくとも一つ含むトランザクションの集合，$SC_1(\alpha)$ はアイテム列 α を部分アイテム列として含むトランザクションの集合とする．

このとき，集約度制約を (3) の式で定義する．

$$\frac{|SC_1(\alpha)|}{|SC_0(\alpha)|} \ge \gamma \qquad (3)$$

特に，(3) の式の左辺の値をアイテム列 α の集約度と呼ぶ．(3) の式は，アイテム列 α の各アイテムが同時出現する頻度を表しており，0.0 から 1.0 の間の値をとる．もし，すべてのトランザクションがアイテム列 α を含んでいる場合，集約度の値は 1.0 となる．

飽和系列パターンマイニングによって抽出される部分アイテム列のうち，集約度制約を満足する部分アイテム列を抽出するマイニング手法を集約度制約付き飽和系列パターンマイニング（intensity constraint based closed sequential pattern mining）と呼ぶ．集約度制約を用いることで，集約度が小さい部分アイテム列を排除することができ，飽和系列パターンマイニング

を用いる場合と比較して，パターン数の絞り込みと処理時間の削減が可能となる．

集約度制約付き飽和系列パターンマイニングについて，表3のアイテム列データベースを例に説明する．アイテム列 α = <a, d, b> とする．このとき，アイテム a を含むトランザクションは {T1, T2, T3, T4, T5} であり，アイテム d を含むトランザクションは {T1, T2, T3} であり，アイテム b を含むトランザクションは {T1, T2, T3, T4, T5} である．また，アイテム列 <a, d, b> を部分アイテム列として含むトランザクションは {T1, T2, T3} である．よって，$|SC_0(\alpha)| = 5$　$|SC_1(\alpha)| = 3$ となり，アイテム列 α の集約度は 3/5 = 0.6 となる．

表3　アイテム列データベースの例

T1	<a，d，e，c，b>
T2	<a，d，f，c，b>
T3	<a，d，c，b>
T4	<a，g，i，h，b>
T5	<a，g，j，h，b>

最小出現回数を $min\_sup = 2$ とする．このとき，表3のアイテム列データベースに対して飽和系列パターンマイニングを適用すると，アイテム列 <a, b>，<a, d, c, b>，<a, g, h, b> がパターンとして抽出される．一方，もし最小出現回数を $min\_sup = 2$，最小集約度を $\gamma = 1.0$ として集約度制約付き飽和系列パターンマイニングを適用すると <a, b>，<d, c>，<g, h> が抽出される（表4）．

表4　集約度制約による飽和系列パターンマイニング結果の違い

集約度なし	<a, b>, <a, d, c, b>, <a, g, h, b>
集約度あり	<a, b>, <d, c>, <g, h>

表3の例では，アイテム a とアイテム b はすべてのトランザクションに含まれている．また，アイテム d とアイテム c はトランザクション {T1, T2, T3} に含まれている．よって，飽和系列パターンマイニングの結果として，部分アイテム列 <a, d, c, b> が抽出される．一方，アイテム d とアイテム c はトランザクション {T4, T5} に含まれない．そのため，集約度制約付き飽和系列パターンマイニングを適用すると，部分アイテム列 <a, d, c, b> の集約度は 0.6 となりパターンとして抽出されず，部分アイテム列 <a, b> と <d, c> がそれぞれ異なるパターンとして抽出される．さらに，飽和系列パターンマイニングの適用結果には，アイテム列 <a, b> は三つのパターンの部分アイテム列として含まれる．これは，アイテム a およびアイテム b がすべてのトランザクションに出現しているためである．しかし，集約度制約付き飽和系列パターンマイニングを適用すると，集約度の制約により部分アイテム列 <a, b> は唯一自分自身のみに含まれ，他のアイテムと組み合わさって作られたパターンは，集約度が低くなり抽出されなくなる．

集約度制約付き飽和系列パターンマイニングを抽出するアルゴリズムには，飽和系列パターンマイニングを抽出するアルゴリズムとして知られている BIDE アルゴリズムを拡張した IC-BIDE（intensity-constraint BIDE）アルゴリズム[4]を適用することが可能である．

4. コーディングパターン抽出を利用した実験

本章では，コーディングパターン抽出問題に飽和系列パターンマイニングと集約度制約付き飽和系列パターンマイニングを適用した実験について述べ，集約度制約の有効性を示す．コーディングパターンとは，ソースコード内に頻出する関数の呼び出し順序や制御構造の組み合わせを指す．例えば，データベースを利用したソースコードにおける *lock* と *unlock* の関係はその典型例である．コーディングパターンを抽出することで，開発者のコードの理解に役立てることができる．

まず，コーディングパターン抽出に系列パターンマイニングを適用する方法について述べる．図 3 の例において，関数 funcA は，関数 func1，関数 func2，関数 func3 を呼び出している．このとき，<func1, func2, func3> をアイテム列とすることでトランザクションを作成することができ，アイテム列データベースを構築することができる．このアイテム列データベースに対して，系列パターンマイニングを適用することで，頻出する関数呼び出し関係を抽出することができる．ここで抽出された呼び出し関係は，そのソースコードの中で特徴的に頻出するパターンとみることができる．これらのパターンをコーディングパターンとする．

```
void funcA(){
  int a = 0, b = 0;
  func1(&a);
  func2(&b);
  int c = 0;
  c = a + b;
  func3(c);
 return;
}
```

図 3　関数例

コーディングパターン抽出に関して，集約度制約付き飽和系列パターンマイニングが有効であることが期待できる．図 4 は，MySQL のソースコード内で呼び出されている各関数の呼び出し回数を降順に並べたものである．このグラフから一部の関数は頻繁に呼び出されているが，ほとんどの関数の呼び出し回数は少ないことがわかる．そのため，最小出現回数を大きく設定すると，ほとんどの関数が抽出されたパターンに含まれない．多くの関数がパターンに含まれるようにするためには，最小出現回数を小さく設定する必要がある．

図 5 は，MySQL のソースコードから最小出現回数を 10 として飽和系列パターンマイニングを適用することで抽出した各コーディングパターンに関して，集約度の分布を示したグラフである．

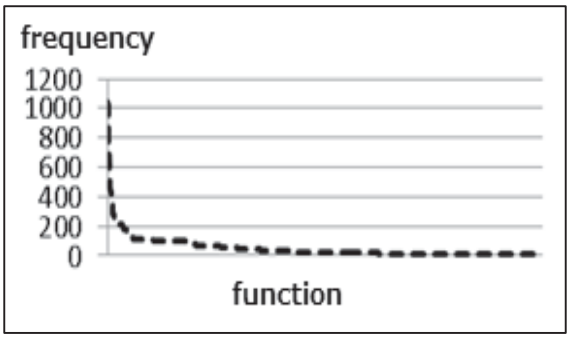

図 4　関数の呼び出し回数

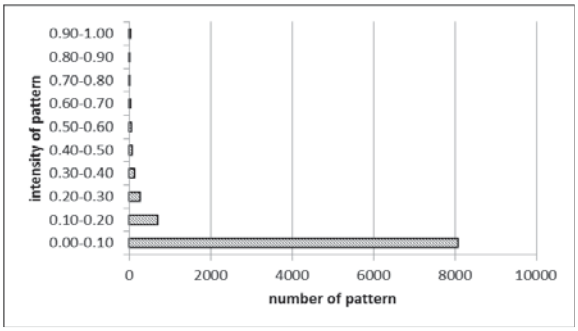

図 5　MySQL のコーディングパターンの集約度の分布

図 5 より，抽出されたほとんどのコーディングパターンの集約度は 0.10 以下であることがわかる．つまり，集約度を制約とすることで多くのパターンを排除することができ，処理時間も削減することができるため，最小出現回数として小さな値を設定することが可能となる．

実験として，IC-BIDE アルゴリズムを適用した集約度制約付き飽和系列パターンマイニングと BIDE アルゴリズムを適用した飽和系列パターンマイニングの結果を比較する．実験には，オープンソースのコードである Bullet Physics[*1]，MySQL[*2]，OpenCV[*3] のソースコードを用いた．各ソースコードの呼び出し関数の数および呼び出し元関数の数は表 5 の通りである．ここで，呼び出し関数の数は，アイテム列データベースに出現するアイテムの総数に対応し，呼び出し元関数の数はトランザクション数に対応する．

表 5　呼び出し関数および呼び出し元関数の数

	呼び出し 関数の数	呼び出し元 関数の数
Bullet Physics	9,399	21,762
MySQL	18,076	23,393
OpenCV	13,078	20,423

IC-BIDE アルゴリズムと BIDE アルゴリズムの最小出現回数として 20，15，10 を用いた．また IC-BIDE アルゴリズムの集約度として 0.9，0.6，0.3 を用いた．実験環境は CPU：Intel® Core™ i7 CPU 960 3.20GHz，Memory：8.00GB のマシンを用いた．

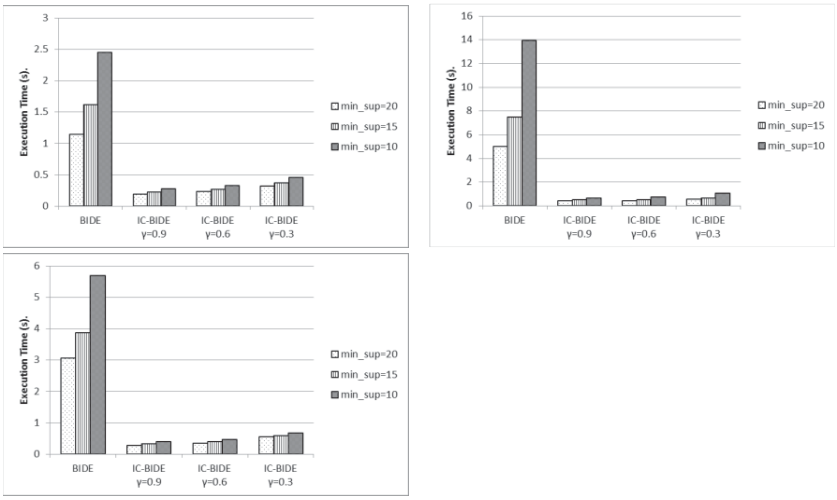

図 6　コーディングパターン抽出処理時間
（左上：Bullet Physics　右上：MySQL　左下：OpenCV）

図 6 は，各ソースコードおよび各アルゴリズムを用いてコーディングパターンを抽出した場合の処理時間である．IC-BIDE アルゴリズムの処理時間は BIDE アルゴリズムの処理時間と比較して劇的に削減されていることがわかる．最小出現回数を 10，集約度を 0.90 とした場合，IC-BIDE アルゴリズムは BIDE アルゴリズムと比較して，平均約 14.6 倍の速度向上を達成した．IC-BIDE アルゴリズムでは，集約度制約により抽出されるパターン数が絞り込まれ，その結果，処理を高速化することができた．

次に，抽出されたパターンを観察する．表 6 は，集約度制約付き飽和系列パターンマイニングにより抽出されたパターンを示している．これらのパターンが集約度の高いパターンとなっている．この結果から，それぞれのソースコードからコーディングパターンが抽出されていることがわかる．

表 6　抽出されたコーディングパターン

Bullet Physics	<btHfFluid, addHfFluid, prep> < setCol0, setCol1, setCol2, setCol3> < tdotx, tdoty, tdotz>
MySQL	<_db_enter_, _db_return_> <_db_pargs_, _db_doprnt_> < WSASetLastError, GetProcessHeap, WSAIoctl, HeapFree >
OpenCV	< cvCreateSubdiv2D, cvInitSubdivDelaunay2D > < CharUpperW, uaw_CharUpperW > < deviceID, setDevice >

ここで MySQL のソースコードから得られたコーディングパターンを用いて，より詳細に観察する．MySQL のソースコードにおいて，最小出現回数を 10 として BIDE アルゴリズムを適用してコーディングパターンを抽出する．このとき，アイテム列 <_db_enter_, _db_return_> というコーディングパターンは 2431 回出現しており，非常に出現回数が多いパターンである．また，アイテム列 <_db_enter_, _db_return_> を部分アイテム列として含むコーディングパターンは 1330 通り抽出される．これは，アイテム列 <_db_enter_, _db_return_> が非常に頻

繁に出現することから，他のアイテムや他のパターンと組み合わさったパターンが多数作成されていることに起因する．しかし，これらのパターンのほとんどは，集約度の小さいパターンである．そのため，IC-BIDE アルゴリズムを用いてコーディングパターンを抽出すると，これら小さい集約度を持つパターンを排除することができる．最小出現回数を 10，集約度を 0.9 として IC-BIDE アルゴリズムを適用すると，アイテム _db_enter_ とアイテム _db_return_ の両方を含むパターンは <_db_enter_, _db_return_> ただ一つとなる．これは，集約度を用いた制約が効果的に無駄なパターンを排除していることを意味している．

5. お わ り に

本稿では，集約度制約付き飽和系列パターンマイニングについて述べ，コーディングパターン抽出を用いた実験により，集約度制約の有効性を示した．集約度制約付き飽和系列パターンマイニングは，コーディングパターン抽出のみならず，他の分野への系列パターンマイニングに対しても有効であると考えており，今後，新たなデータマイニングの道具の一つとして活用していきたい．

また，本稿の集約度制約付き飽和系列パターンマイニングは，早稲田大学の山名早人教授の下での研究成果によるものである．教授には多くのご助言とご指導を頂いたこと心より感謝申し上げる．

＊ 1　Bullet Physics Library, http://bulletphysics.org/wordpress/
＊ 2　MySQL Developer Zone, http://dev.mysql.com/
＊ 3　OpenCV, http://opencv.willowgarage.com/wiki/Welcome
（上記注釈の URL は 2013 年 1 月 16 日時点での存在を確認）

参考文献　［1］ R. Agrawal R. Srikant, “Mining sequential patterns”, *Proc. of ICDE’95, 1995, pp.3-14.*
［2］ J. Pei J. Han B. Mortazavi-Asl H. Pinto Q. Chen U. Dayal M. Hsu, “PrefixSpan: Mining Sequential Patterns by Prefix-Projected Growth”, *Proc. of 17th ICDE, 2001, pp.215-224.*
［3］ J. Wang J. Han, “BIDE: Efficient mining of frequent closed sequences”, *Proc. of 20th ICDE, 2004, pp.79-90.*
［4］ H. Takei H. Yamana, “IC-BIDE: Intensity Constraint-based Closed Sequential Pattern Mining for Coding Pattern Extraction”, *Proc. of 27th AINA, 2013, to appear.*

執筆者紹介　武　井　宏　将（Hiromasa Takei）
2004 年日本ユニシス（株）入社．入社時より CAD/CAM 分野のシステム開発業務に従事．2011 年より社会人大学院生として早稲田大学に留学し，データマイニングおよび形状検索の研究に従事．