Data Mining Package TAKE(竹) Documentation

対応 NYSOL パッケージ: Ver. 1.2,2.0

revise history:

October 6, 2014 : mclque2g.rb,mbiclique.rb などの追加、枝ファイル、節点ファイルの指定方法の変更

March 10, 2014 : nysol パッケージへの統合に伴いインストール方法変更

March 10, 2014 : mitemset.rb 改良,msequence.rb,mpolishing.rb など追加

February 28, 2014 : first release

2014 年 10 月 7 日

目次

第 1 章

はじめに

1.1 概要

本パッケージ「TAKE(竹)」は、宇野毅明氏 (国立情報学研究所教授) を中心に開発されたデータマイニングソフトウェア [3] を拡張し利用しやすくしたコマンド群である。パッケージ名は開発者の名前の音からとったものである。本パッケージが提供するコマンドの多くはパターン列挙を基本としたものであり、その対象はアイテム集合、系列、一般グラフと多様である。

例えば、スーパーマーケットにおける買い物かごに入った商品をアイテム集合として考えれば、多くの買い物かごに共通の商品の組合せを高速に列挙できる。それだけでなく、クラスの概念を導入し、優良顧客に特徴的なパターンを列挙することも可能である。

また web ページの巡回ログから、多くのユーザに共通の巡回パターンを系列パターンとして列挙でき web ページの構成に参考となる知見が見つかるかもしれない。系列パターンにおいても、クラスの概念を導入することが可能で、男性と女性での巡回パターンの差を列挙することもできる。

さらに、一般グラフを扱うコマンドもいくつか用意されている。企業の取引ネットワークデータや SNS のユーザネットワークデータ、そしてアイテム間類似関係を表現した類似度グラフなどを扱うことができる。例えば、商品の共起情報を類似度と考えれば、商品の類似度グラフを構築でき、そのグラフから極大クリークを列挙することでお互いにつながりの強い商品クラスタを抽出できる。抽出されたクラスタは、商品購買モデルにおける説明変数として利用することも可能である。また、時に膨大に列挙される極大クリークの数を抑制し、中規模サイズの少数の極大クリークを列挙するための前処理としてデータ研磨手法も用意されている。

本パッケージが提供するコマンドの全ては ruby 言語によって記述されており、内部でネイティブなコマンドをシェルインターフェースによって起動実行している。ネイティブコマンドとのデータのやり取りは、基本的にはファイルによる。

1.1.1 2014 年 10 月 6 日リリースによる変更点

`mclique.rb,mpolishing.rb` コマンドで、枝ファイルの入出力を指定するパラメータのキーワードを、`i=`および `o=`から `ei=`および `no=`に変更している。これは節点ファイルを `ni=`および `no=`で指定できるようにした変更に伴うものである。節点ファイルを指定することで、これまで列挙できなかった一つの節点から構成されるクリークも出力されるようになる。

1.1.2 インストール

本パッケージは全て nysol パッケージに含まれている。nysol パッケージをインストールすれば必要なソフトウェアは全てインストールされる。詳しくは nysol パッケージのインストールの説明 (`http://www.nysol.jp/install`) を参照のこと。

1.1.3 ライセンス

本パッケージには、宇野氏の開発したコマンドソースも含まれているが、それらのライセンスは [3] で配布されているアーカイブに含まれる `readme.txt` ファイルを参照のこと。それ以外のソフトウェアは GNU AGPL(AFFERO GENERAL PUBLIC LICENSE: `http://www.gnu.org/licenses/agpl-3.0.html`) のもと自由に利用可能である。

第 2 章

コマンド

2.1　mitemset.rb 頻出アイテム集合の列挙

トランザクションデータから頻出アイテム集合 (頻出パターンとも呼ぶ) を列挙する。列挙のコアアルゴリズムには LCM(Linear time Closed itemset Miner) を用いている [1, 3]。

本コマンドは以下のような特徴を持つ。

- 他のアイテム集合に含まれないアイテム集合「極大アイテム集合」を列挙することができる。
- 同一の出現を持つ極大アイテム集合「飽和アイテム集合」を列挙することができる。
- アイテムの階層分類 (taxonomy) を用いることができる。
- 分類クラスを指定することで、あるクラスに特徴的なパターン (顕在パターン:emerging patterns) を列挙することができる。3 つ以上のクラスにも対応している。

入力データとしては、表 2.1,2.2,2.3 で示されるような多様なデータ型を用いることができるが、本コマンドが直接扱うのは key 型データ (表 2.1) のみである。その他のデータ型は、MCMD パッケージの `mtra` および `mtraflg` コマンドによって、あらかじめ key 型データに変換しておく必要がある。

表 2.1 において `key` 項目の値が同じ行、もしくは表 2.2,2.3 における各行のことを特にトランザクションと呼び、トランザクションに含まれる要素をアイテムと呼ぶ。スーパーマーケットの例では、トランザクションをレシート、アイテムを購入商品と考えればわかりやすいであろう。

表 2.1　key 型データ

key	item
T1	C
T1	E
T2	D
T2	E
T2	F
:	:

表 2.2　tra 型データ

id	item
T1	C E
T2	D E F
T3	A B D F
T4	B D F
T5	A B D E
T6	A B D E F

表 2.3　行列型データ

id	A	B	C	D	E	F
T1			1		1	
T2				1	1	1
T3	1	1		1		1
T4		1		1		1
T5	1	1		1	1	
T6	1	1		1	1	1

頻出アイテム集合

頻出アイテム集合とは、出現頻度 (サポートと呼ぶ) がユーザの与えた最小サポート以上であるようなアイテム集合のことを言う。表 2.2 を例にとると、最小サポートが 3 件とすると、アイテム集合 {B,D,F} は T3,T5,T6 の 3 件に出現しているので頻出であるがアイテム集合 {B,D,E} は T5,T6 の 2 件にしか出現していないので頻出ではない。ここで最小サポート 3 件を満たす頻出アイテム集合は、{A}, {A,B}, {A,B,D}, {A,D}, {B}, {B,D}, {B,D,F}, {B,F}, {D}, {D,E}, {D,F}, {E}, {F} の計 13 件である。

頻出アイテム集合を列挙すると、時にその数は膨大なものとなることがある。そこで、列挙された頻出アイテム集合から代表的なアイテム集合のみを出力する方法として、極大アイテム集合と飽和アイテム集合の列挙がある。

極大アイテム集合

ある頻出アイテム集合について、そのアイテム集合がその他の頻出アイテム集合に包含されていなければ、そのアイテム集合を頻出極大アイテム集合と呼ぶ。前節に示してた 13 件の頻出アイテム集合のうち、{A,B,D},{B,D,F},{D,E} の 3 つのアイテム集合は、他のどのアイテム集合にも包含されないので極大アイテム集合である。その他のアイテム集合は、上記 3 つの極大アイテム集合のいずれかに包含されているため極大ではない。

飽和アイテム集合

任意の二つの頻出アイテム集合について、それらのアイテム集合が出現するトランザクションが同じであれば、それら二つのアイテム集合は同じグループに属すると考える。このようにして頻出アイテム集合をグルーピングする

と、各グループには、そのサイズが最も大きいアイテム集合がただ一つあることが分かっている。このような頻出アイテム集合を飽和アイテム集合と呼ぶ。例えば、{A},{A,B},{A,D},{A,B,D} は全て T3,T5,T6 のトランザクションに出現するので同一グループである。そしてこの中で最もサイズの大きい {A,B,D} が飽和集合ととして出力され、{A},{A,B},{A,D} は出力されない。全ての飽和集合を列挙すると表 2.4 の通り 7 つ存在する。飽和集合という代表的なアイテム集合を列挙することで列挙件数を大幅に減少できている。

表 2.4 飽和アイテム集合

飽和集合	出現トランザクション	グループ
{A,B,D}	T3,T5,T6	{A},{A,B},{A,D},{A,B,D}
{B,D}	T3,T4,T5,T6	{B},{B,D}
{B,D,F}	T3,T4,T6	{B,F},{B,D,F}
{D}	T2,T3,T4,T5,T6	{D}
{D,E}	T2,T5,T6	{D,E}
{D,F}	T2,T3,T4,T6	{F},{D,F}
{E}	T1,T2,T5,T6	{E}

顕在パターン

各トランザクションが属する「クラス」を導入し、あるクラスに特徴的なパターン (ここでは頻出アイテム集合) を列挙する。ここで特徴的とは、あるクラスには多頻度で、他のクラスでは多頻度でないことである。例えば、スーパーマーケットでは、男性と女性で購買されるアイテムの違いを識別したい時などに使われる。顕在パターンについてのより詳細な定義は資料 1 を参照のこと。クラスデータは、表 2.5 のように、トランザクションデータにクラス項目を結合することで与える。同じトランザクションキーを持つ行は同じクラス値が与えられなければならない。

表 2.5 key 型データ

key	item	class
T1	C	pos
T1	E	pos
T2	D	neg
T2	E	neg
T2	F	neg
:	:	

階層分類

アイテムの階層分類を反映させることができる。例えば、スーパーマーケットでは「○○牛乳 500ml」というアイテムは「牛乳」に分類され、「牛乳」は「乳製品」に分類され、さらに「乳製品」は食品に分類される。このような階層分類を導入することで、「○○牛乳 500ml」は「果物」と購入される頻度が多いといった、階層分類の異なるアイテム間の関係を見つけることができるかもしれない。ただし、現行バージョンでは、階層は 1 段階のみ対応している。

内部で行われる処理は至ってシンプルで、アイテムと分類の対応関係 (表 2.7) を与え、入力データ (表 2.6) のアイテムに応じた分類アイテムを追加 (表 2.8)、もしくは置換 (表 2.9) した後に、頻出アイテム集合を求める。ただし、冗長なアイテム集合は出力されない。冗長なアイテム集合とは、親子関係にあるアイテムペアを含むようなアイテム集合である。例えば、表 2.8 においてアイテム集合 {A,X} は T3,T5,T6 の 3 件に出現するが、アイテム X はアイテム A と親子関係にあるため、このようなパターンは出力されない。なお、冗長なアイテム集合の削除には nysol の ZDD パッケージ (http://www.nysol.sakura.ne.jp/zdd/jp/) を利用している。

表 2.6 元データ

id	item
T1	C E
T2	D E F
T3	A B D F
T4	B D F
T5	A B D E
T6	A B D E F

表 2.7 item-分類対応表

item	taxonomy
A	X
B	X
C	Y
D	Z
E	Z
F	Z

表 2.8 分類追加後データ

id	item
T1	C E Y Z
T2	D E F Z
T3	A B D F X Z
T4	B D F Y Z
T5	A B D E X Z
T6	A B D E F X Z

表 2.9 分類置換後データ

id	item
T1	Y Z
T2	Z
T3	X Z
T4	Y Z
T5	X Z
T6	X Z

出力

本コマンドが出力するデータは、大きく分けて 2 つあり、一つは、列挙されたパターンデータ (`patterns.csv`)、そして他方は、それらのパターンをどのトランザクションが含むかについての情報 (`tid_pats.csv`) である。パターンデータは、顕在パターンの場合異なる CSV 項目を出力する。表 2.10〜表 2.12 にそれらのサンプルを示す。

表 2.10 patterns.csv のデータ例。pid 項目は一つのパターンを識別するための ID で、size はパターンとしてのアイテム集合を構成するアイテム数、count はそのパターンが出現したトランザクション数、そして total は全トランザクション数である。support は出現確率で、count/total で計算される。lift はリフト値で、期待される出現確率に対する実出現確率の比である。最後に pattern がアイテム集合で、アイテムは半角スペースで区切られている。

pid	size	count	total	support	lift	pattern
1	1	5	6	0.8333333333	1	D
7	2	4	6	0.6666666667	1.2	D F
6	1	4	6	0.6666666667	1	F
4	1	4	6	0.6666666667	1	E
2	1	4	6	0.6666666667	1	B
3	2	4	6	0.6666666667	1.2	B D
8	2	3	6	0.5	1.125	B F
13	2	3	6	0.5	1.2	A D

アイテム集合 $I = \{i_1, i_2, \cdots, i_n\}$ のリフト値 $\mathrm{lift}(I)$ は次の通り定義される。

$$\mathrm{lift}(I) = \frac{\Pr(I)}{\prod_{k=1}^{n} \Pr(i_k)}$$

表 2.11 tidpats.csv の内容例。tid はトランザクション ID で、tid=パラメータで指定した入力データ項目に対応している。そしてそれぞれのトランザクションに含まれるパターンの ID が pid で示されている。

tid	pid
T1	4
T2	1
T2	4
T2	7
T2	6
T2	5
T3	10
T3	6

2.1.1 書式

```
mitemset.rb i= [x=] [O=] [tid=] [item=] [class=] [taxo=] [s=|S=] [sx=|SX=] [l=] [u=]
            [p=] [g=] [top=] [T=] [--help]
```

- `i=` key 型トランザクションデータファイル名【必須】
- `c=` クラスファイル名【オプション】
- `x=` 階層分類データファイル名【オプション】
- `O=` 出力パス名【オプション:default=./take_#{日付時刻}】
- `tid=` トランザクション ID 項目名【必須】
- `item=` アイテム項目名（i=,x=上の項目名）【オプション:default="item"】
- `class=` クラス項目名（c=上の項目名）【オプション】
  クラス項目名を指定すると顕在パターンが列挙される。
- `taxo=` 分類項目名【条件付き必須:x=】

表 2.12 顕在パターンにおける patterns.csv の内容例。class 項目は、どのクラスに特徴的な顕在パターンか、その対象クラスを示している。pid,pattern,size,total は表 2.10 と同様である。pos は対象クラスのトランザクションに出現した件数で、neg はそれ以外のクラスのトランザクションに出現した件数である。posTotal,negTotal は、対象クラスとそれ以外のクラスのトランザクション件数である。support は、対象クラスにおける出現確率で、pos/posTotal で計算される。growthRate は増加率で、support/(neg/negTotal) で計算される値である。分母が 0 の場合は inf と表示される。この値が大きいほど、対象クラスに特徴的であることを意味する。postProb は、パターンを条件とした時の対象クラスの事後確率で、growthRate と同様、この値が大きいほど対象クラスに特徴的であることを意味する。詳細な定義は資料 1 を参照のこと。

class	pid	pattern	size	pos	neg	posTotal	negTotal	total	support	growthRate	postProb
cls2	13	A E	2	2	0	2	4	6	1	inf	1
cls2	15	A B E	3	2	0	2	4	6	1	inf	1
cls2	10	A B D E	4	2	0	2	4	6	1	inf	1
cls2	14	B E	2	2	0	2	4	6	1	inf	1
cls2	17	A D E	3	2	0	2	4	6	1	inf	1
cls2	18	B D E	3	2	0	2	4	6	1	inf	1
cls2	12	A B D	3	2	1	2	4	6	1	4	0.6666666667
cls2	11	A D	2	2	1	2	4	6	1	4	0.6666666667
cls2	16	D E	2	2	1	2	4	6	1	4	0.6666666667

`type=` **アイテム集合のタイプ【オプション: default=F】**
F:頻出アイテム集合，C:飽和アイテム集合，M:極大アイテム集合
`s=` **最小サポート（確率）【選択必須:s=, S=】**
`S=` **最小サポート（件数）【選択必須:s=, S=】**
`sx=` **最大サポート（確率）【オプション】**
`SX=` **最大サポート（件数）【オプション】**
`l=` **最小アイテム集合サイズ【オプション】**
`u=` **最大アイテム集合サイズ【オプション】**
`p=` **顕在パターン用最小事後確率【オプション:default=0.5】**
`g=` **顕在パターン用最小増加率【オプション】**
`top=` **列挙するパターン数の上限【オプション:default:制限なし】**
`T=` **ワークディレクトリ【オプション】**
`--help` **ヘルプの表示**

2.1.2 利用例

例 1: 基本例

3 件以上で出現する頻出アイテム集合を列挙する。

```
$ more dat1.csv
tid,item
T1,C
T1,E
T2,D
T2,E
T2,F
T3,A
T3,B
T3,D
T3,F
T4,B
T4,D
T4,F
```

```
T5,A
T5,B
T5,D
T5,E
T6,A
T6,B
T6,D
T6,E
T6,F
$ mitemset.rb  S=3 tid=tid item=item i=dat1.csv O=result1
#MSG# lcm_20140215 FIf /tmp/__MTEMP_96064_70346946698540_0 3 /tmp/__MTEMP_96064_7034694669
7820_0
trsact: /tmp/__MTEMP_96064_70346946698540_0 ,#transactions 6 ,#items 6 ,size 21 extracted
database: #transactions 6 ,#items 5 ,size 20
 output to: /tmp/__MTEMP_96064_70346946697820_0
separated at 0
iters=8
14
1
5
6
2
trsact: /tmp/__MTEMP_96064_70346946698540_0 ,#transactions 6 ,#items 6 ,size 21 extracted
database: #transactions 6 ,#items 6 ,size 21
 output to: /tmp/__MTEMP_96064_70346946697820_1
separated at 0
iters=7
6
0
6
#MSG# output patterns to CSV file ...
#MSG# the number of patterns enumerated is 13
#MSG# output tid-patterns ...
#MSG# The final results are in the directory ‘result1’
#END# /usr/bin/mitemset.rb S=3 tid=tid item=item i=dat1.csv O=result1
$ more result1/patterns.csv
pid,size,count,total,support,lift,pattern
1,1,5,6,0.8333333333,1,D
7,2,4,6,0.6666666667,1.2,D F
6,1,4,6,0.6666666667,1,F
4,1,4,6,0.6666666667,1,E
2,1,4,6,0.6666666667,1,B
3,2,4,6,0.6666666667,1.2,B D
8,2,3,6,0.5,1.125,B F
13,2,3,6,0.5,1.2,A D
5,2,3,6,0.5,0.9,D E
12,3,3,6,0.5,1.8,A B D
11,2,3,6,0.5,1.5,A B
10,1,3,6,0.5,1,A
9,3,3,6,0.5,1.35,B D F
$ more result1/tid_pats.csv
tid,pid
T1,4
T2,1
T2,4
T2,7
T2,6
T2,5
T3,10
T3,6
T3,13
T3,7
T3,11
T3,8
```

```
T3,3
T3,12
T3,1
T3,2
T3,9
T4,6
T4,7
T4,8
T4,2
T4,1
T4,3
T4,9
T5,11
T5,13
T5,3
T5,1
T5,4
T5,10
T5,5
T5,2
T5,12
T6,2
T6,11
T6,6
T6,7
T6,5
T6,10
T6,1
T6,8
T6,12
T6,4
T6,9
T6,13
T6,3
```

例 2: **アイテム集合のサイズに制限を加えた例**

出現頻度が 3 以上で、アイテム集合のサイズ 3 のパターンを列挙する。

```
$ mitemset.rb S=3 l=3 u=3 tid=tid item=item i=dat1.csv O=result2
#MSG# lcm_20140215 FIf -l 3 -u 3 /tmp/__MTEMP_96148_70251320761080_0 3 /tmp/__MTEMP_96148_
70251320760380_0
trsact: /tmp/__MTEMP_96148_70251320761080_0 ,#transactions 6 ,#items 6 ,size 21 extracted
database: #transactions 6 ,#items 5 ,size 20
 output to: /tmp/__MTEMP_96148_70251320760380_0
separated at 0
iters=8
2
0
0
0
2
trsact: /tmp/__MTEMP_96148_70251320761080_0 ,#transactions 6 ,#items 6 ,size 21 extracted
database: #transactions 6 ,#items 6 ,size 21
 output to: /tmp/__MTEMP_96148_70251320760380_1
separated at 0
iters=7
6
0
6
#MSG# output patterns to CSV file ...
#MSG# the number of patterns enumerated is 2
#MSG# output tid-patterns ...
```

```
#MSG# The final results are in the directory ‘result2’
#END# /usr/bin/mitemset.rb S=3 l=3 u=3 tid=tid item=item i=dat1.csv O=result2
$ more result2/patterns.csv
pid,size,count,total,support,lift,pattern
0,3,3,6,0.5,1.35,B D F
1,3,3,6,0.5,1.8,A B D
```

例 3: **飽和集合の列挙例**

```
$ mitemset.rb S=3 type=C tid=tid item=item i=dat1.csv O=result3
#MSG# lcm_20140215 CIf /tmp/__MTEMP_96231_70281540721720_0 3 /tmp/__MTEMP_96231_7028154072
1000_0
trsact: /tmp/__MTEMP_96231_70281540721720_0 ,#transactions 6 ,#items 6 ,size 21 extracted
database: #transactions 6 ,#items 5 ,size 20
 output to: /tmp/__MTEMP_96231_70281540721000_0
separated at 0
iters=8
8
1
2
3
2
trsact: /tmp/__MTEMP_96231_70281540721720_0 ,#transactions 6 ,#items 6 ,size 21 extracted
database: #transactions 6 ,#items 6 ,size 21
 output to: /tmp/__MTEMP_96231_70281540721000_1
separated at 0
iters=7
6
0
6
#MSG# output patterns to CSV file ...
#MSG# the number of patterns enumerated is 7
#MSG# output tid-patterns ...
#MSG# The final results are in the directory ‘result3’
#END# /usr/bin/mitemset.rb S=3 type=C tid=tid item=item i=dat1.csv O=result3
$ more result3/patterns.csv
pid,size,count,total,support,lift,pattern
1,1,5,6,0.8333333333,1,D
2,2,4,6,0.6666666667,1.2,B D
3,1,4,6,0.6666666667,1,E
5,2,4,6,0.6666666667,1.2,D F
4,2,3,6,0.5,0.9,D E
6,3,3,6,0.5,1.35,B D F
7,3,3,6,0.5,1.8,A B D
```

例 4: **極大集合の列挙例**

```
$ mitemset.rb S=3 type=M tid=tid item=item i=dat1.csv O=result4
#MSG# lcm_20140215 MIf /tmp/__MTEMP_96314_70152347761620_0 3 /tmp/__MTEMP_96314_7015234776
0900_0
trsact: /tmp/__MTEMP_96314_70152347761620_0 ,#transactions 6 ,#items 6 ,size 21 extracted
database: #transactions 6 ,#items 5 ,size 20
 output to: /tmp/__MTEMP_96314_70152347760900_0
separated at 0
iters=8
3
0
0
1
2
trsact: /tmp/__MTEMP_96314_70152347761620_0 ,#transactions 6 ,#items 6 ,size 21 extracted
database: #transactions 6 ,#items 6 ,size 21
```

```
 output to: /tmp/__MTEMP_96314_70152347760900_1
separated at 0
iters=7
6
0
6
#MSG# output patterns to CSV file ...
#MSG# the number of patterns enumerated is 3
#MSG# output tid-patterns ...
#MSG# The final results are in the directory ‘result4’
#END# /usr/bin/mitemset.rb S=3 type=M tid=tid item=item i=dat1.csv O=result4
$ more result4/patterns.csv
pid,size,count,total,support,lift,pattern
0,2,3,6,0.5,0.9,D E
1,3,3,6,0.5,1.35,B D F
2,3,3,6,0.5,1.8,A B D
```

例 5: **アイテムの階層分類を使った例**

```
$ more taxo.csv
item,taxonomy
A,X
B,X
C,Y
D,Z
E,Z
F,Z
$ mitemset.rb S=4 tid=tid item=item i=dat1.csv x=taxo.csv taxo=taxonomy O=result5
#MSG# lcm_20140215 FIf /tmp/__MTEMP_96398_70213488525420_0 4 /tmp/__MTEMP_96398_7021348851
6480_0
trsact: /tmp/__MTEMP_96398_70213488525420_0 ,#transactions 6 ,#items 10 ,size 32 extracted
 database: #transactions 6 ,#items 6 ,size 27
 output to: /tmp/__MTEMP_96398_70213488516480_0
separated at 0
iters=6
22
1
6
9
5
1
trsact: /tmp/__MTEMP_96398_70213488525420_0 ,#transactions 6 ,#items 10 ,size 32 extracted
 database: #transactions 6 ,#items 9 ,size 32
 output to: /tmp/__MTEMP_96398_70213488516480_1
separated at 0
iters=9
9
0
9
#MSG# output patterns to CSV file ...
#MSG# reducing redundant rules in terms of taxonomy ...
#MSG# the number of patterns enumerated is 11
#MSG# output tid-patterns ...
#MSG# The final results are in the directory ‘result5’
#END# /usr/bin/mitemset.rb S=4 tid=tid item=item i=dat1.csv x=taxo.csv taxo=taxonomy O=res
ult5
$ more result5/patterns.csv
pid,size,count,total,support,lift,pattern
1,1,6,6,1,1,Z
2,1,5,6,0.8333333333,1,D
19,2,4,6,0.6666666667,1.2,D X
13,2,4,6,0.6666666667,1,B Z
14,1,4,6,0.6666666667,1,X
```

```
6,1,4,6,0.6666666667,1,F
11,2,4,6,0.6666666667,1.2,B D
21,2,4,6,0.6666666667,1,X Z
4,1,4,6,0.6666666667,1,E
10,1,4,6,0.6666666667,1,B
7,2,4,6,0.6666666667,1.2,D F
```

例 6: **オリジナルアイテムを階層分類で置換する例**

```
$ more taxo.csv
item,taxonomy
A,X
B,X
C,Y
D,Z
E,Z
F,Z
$ mitemset.rb S=4 tid=tid item=item i=dat1.csv x=taxo.csv taxo=taxonomy -replaceTaxo O=resul
t6
#MSG# lcm_20140215 FIf /tmp/__MTEMP_96532_70167229154680_0 4 /tmp/__MTEMP_96532_7016722915
3720_0
trsact: /tmp/__MTEMP_96532_70167229154680_0 ,#transactions 6 ,#items 4 ,size 11 extracted
database: #transactions 6 ,#items 2 ,size 10
 output to: /tmp/__MTEMP_96532_70167229153720_0
separated at 0
iters=2
4
1
2
1
trsact: /tmp/__MTEMP_96532_70167229154680_0 ,#transactions 6 ,#items 4 ,size 11 extracted
database: #transactions 6 ,#items 3 ,size 11
 output to: /tmp/__MTEMP_96532_70167229153720_1
separated at 0
iters=3
3
0
3
#MSG# output patterns to CSV file ...
#MSG# the number of patterns enumerated is 3
#MSG# output tid-patterns ...
#MSG# The final results are in the directory ‘result6’
#END# /usr/bin/mitemset.rb S=4 tid=tid item=item i=dat1.csv x=taxo.csv taxo=taxonomy -repl
aceTaxo O=result6
$ more result6/patterns.csv
pid,size,count,total,support,lift,pattern
1,1,6,6,1,1,Z
2,1,4,6,0.6666666667,1,X
3,2,4,6,0.6666666667,1,X Z
```

例 7: **顕在パターンの列挙例**

```
$ more dat2.csv
tid,item,class
T1,C,cls1
T1,E,cls1
T2,D,cls1
T2,E,cls1
T2,F,cls1
T3,A,cls1
T3,B,cls1
T3,D,cls1
```

```
T3,F,cls1
T4,B,cls1
T4,D,cls1
T4,F,cls1
T5,A,cls2
T5,B,cls2
T5,D,cls2
T5,E,cls2
T6,A,cls2
T6,B,cls2
T6,D,cls2
T6,E,cls2
T6,F,cls2
$ mitemset.rb S=2 tid=tid item=item class=class i=dat2.csv p=0.6 O=result7
#MSG# lcm_20140215 FIA -w /tmp/__MTEMP_96648_70348154647100_1 /tmp/__MTEMP_96648_703481546
47100_0 1073741817 /tmp/__MTEMP_96648_70348154644460_0
trsact: /tmp/__MTEMP_96648_70348154647100_0 ,#transactions 6 ,#items 6 ,size 21 extracted
database: #transactions 6 ,#items 4 ,size 17 ,weightfile /tmp/__MTEMP_96648_70348154647100
_1
 output to: /tmp/__MTEMP_96648_70348154644460_0
separated at 0
iters=6
9
1
4
3
1
#MSG# output patterns to CSV file ...
#MSG# the number of contrast patterns on class ‘cls1’ enumerated is 8
#MSG# output tid-patterns ...
#MSG# lcm_20140215 FIA -w /tmp/__MTEMP_96648_70348154647100_2 /tmp/__MTEMP_96648_703481546
47100_0 2147483645 /tmp/__MTEMP_96648_70348154644460_2
trsact: /tmp/__MTEMP_96648_70348154647100_0 ,#transactions 6 ,#items 6 ,size 21 extracted
database: #transactions 6 ,#items 4 ,size 16 ,weightfile /tmp/__MTEMP_96648_70348154647100
_2
 output to: /tmp/__MTEMP_96648_70348154644460_2
separated at 0
iters=14
11
0
1
5
4
1
#MSG# output patterns to CSV file ...
#MSG# the number of contrast patterns on class ‘cls2’ enumerated is 11
#MSG# output tid-patterns ...
#MSG# the number of emerging patterns enumerated is 16
#MSG# The final results are in the directory ‘result7’
#END# /usr/bin/mitemset.rb S=2 tid=tid item=item class=class i=dat2.csv p=0.6 O=result7
$ more result7/patterns.csv
class,pid,pattern,size,pos,neg,posTotal,negTotal,total,support,growthRate,postProb
cls2,18,B D E,3,2,0,2,4,6,1,inf,1
cls2,14,B E,2,2,0,2,4,6,1,inf,1
cls2,13,A E,2,2,0,2,4,6,1,inf,1
cls2,17,A D E,3,2,0,2,4,6,1,inf,1
cls2,15,A B E,3,2,0,2,4,6,1,inf,1
cls2,10,A B D E,4,2,0,2,4,6,1,inf,1
cls1,2,D F,2,3,1,4,2,6,0.75,1.5,0.75
cls1,1,F,1,3,1,4,2,6,0.75,1.5,0.75
cls2,9,A B,2,2,1,2,4,6,1,4,0.6666666667
cls2,8,A,1,2,1,2,4,6,1,4,0.6666666667
cls1,6,B D F,3,2,1,4,2,6,0.5,1,0.6666666667
cls2,12,A B D,3,2,1,2,4,6,1,4,0.6666666667
```

```
cls1,5,B F,2,2,1,4,2,6,0.5,1,0.6666666667
cls2,16,D E,2,2,1,2,4,6,1,4,0.6666666667
cls2,11,A D,2,2,1,2,4,6,1,4,0.6666666667
cls1,0,D,1,3,2,4,2,6,0.75,0.75,0.6
```

2.1.3 資料 1: パラメータ g=,p=,-uniform について

クラス集合 $C = \{c_1, c_2, \cdots, c_m\}$ があり、各トランザクションは、いずれか一つのクラスに属しているものとする。ここで我々が興味のある顕在パターンとは、ある対象クラスに頻出し、その他のクラスに頻出すしないようなアイテム集合である。例えば、対象クラス c_1 に頻出し、その他のクラス $c_2, c_3, \cdots, c_m$ には頻出しないようなアイテム集合である。以降、対象クラスを c_t、その他のクラスを c_o で表すものとする。

本コマンドでは顕在パターンを以下の 3 種類の方法により定義できる。

1. 増加率 (クラス間の出現確率の比) の閾値を指定
2. 事後確率の閾値を指定 (事前確率はデータ上の分布から推定)
3. 事後確率の閾値を指定 (事前確率は各クラス一様)

1. 増加率

対象クラス c_t におけるアイテム集合 I の増加率 $GR_t(I)$ は式 (2.1) で表され、対象クラスとその他クラスにおけるアイテム集合の出現確率の比として定義される。そして、顕在パターンとは、ユーザが与えた最小増加率 γ 以上の増加率を持つようなアイテム集合のことである。γ は、パラメータ g=によって指定する。

$$GR_t(I) = \frac{\Pr(I|c_t)}{\Pr(I|c_o)} \geq \gamma \tag{2.1}$$

2. 事後確率

クラス未知のあるトランザクションについて、アイテム集合 I を観測したとき、そのトランザクションがクラス c_t に属する確率は、ベイズの定理より式 (2.2) で表される。この式は、クラス c_t の事前確率 $\Pr(c_t)$ がアイテム集合 I を観測することで事後確率 $\Pr(c_t|I)$ に更新されたことを意味する。事前確率 $\Pr(c_t)$ は、与えられたデータにおけるクラス分布に基づいて推定する。ここで、顕在パターンとは、ユーザが与えた最小事後確率 π 以上の事後確率を持つようなアイテム集合のことである。π は、パラメータ p=によって指定する。

$$\Pr(c_t|I) = \frac{\Pr(I|c_t)\Pr(c_t)}{\Pr(I|c_t)\Pr(c_t) + \Pr(I|c_o)\Pr(c_o)} \geq \pi \tag{2.2}$$

3. 事後確率 (事前確率一様)

全てのクラスの事前確率は一様であると仮定して上記の事後確率を計算する。$\Pr(c_t) = \frac{1}{m}$,$\Pr(c_o) = \frac{m-1}{m}$ を式 (2.2) に代入すると式 (2.3) が得られる。そして、顕在パターンとは、事前確率が一様であると仮定のもと、ユーザが与えた最小事後確率 π_u 以上の事後確率を持つようなアイテム集合のことである。π_u は、パラメータ p=によって指定し、かつ-uniform オプションを指定する。

$$\Pr(c_t|I) = \frac{\Pr(I|c_t)}{\Pr(I|c_t) + (m-1)\Pr(I|c_o)} \geq \pi_u \tag{2.3}$$

$GR_t(I)$ と $Pr(c_t|I)$ の関係

式 (2.1) と式 (2.2) より、$GR_t(I)$ と $Pr(c_t|I)$ の関係は式 (2.4) で表される。最小事後確率 π を指定して顕在パターンを列挙する場合、内部的には式 (2.4) に従い、π を最小増加率 γ に変換して実行している。

$$GR_t(I) = \frac{\Pr(c_o)}{\Pr(c_t)} \cdot \frac{\Pr(c_t|I)}{1 - \Pr(c_t|I)} \tag{2.4}$$

2.1.4 資料 2: LCM による顕在パターン列挙について

正例，負例のデータ集合をそれぞれ D_t, D_o とし，それらのサイズは $|D_t| = W|D_o|$ の関係にあるとする．いま，あるアイテム集合 I(以下ではパターン I と呼ぶ) について，D_t における増加率 $GR_t(I)$ および出現ゲイン $Gain_t(I)$ をそれぞれ式 (2.5),(2.6) の通り定義する．

$$GR_t(I) = \frac{sup(I, D_t)/|D_t|}{sup(I, D_o)/|D_o|} = W\frac{sup(I, D_t)}{sup(I, D_o)} \tag{2.5}$$

$$Gain_p(I) = \omega sup(I, D_t) - sup(I, D_o) \tag{2.6}$$

ここで，$sup(I, D_t), sup(I, D_o)$ はパターン I の D_t, D_o における出現件数で，ω は正例の出現数に与えるウェイトを表している．パターン I について，$GR_t(I)$ がユーザによって与えられた最小増加率 γ 以上の場合，そのパターンを顕在パターンと呼び，$Gain_p(I)$ がユーザによって与えられた最小サポート σ 以上の場合，そのパターンをコントラストパターンと呼ぶことにする．顕在パターンおよびコントラストパターンを正例，負例における出現数の関係で示すと式 (2.7),(2.8) の通りとなる．

$$sup(I, D_o) \leq \frac{W}{\gamma} sup(I, D_t) \tag{2.7}$$

$$sup(I, D_o) \leq \omega sup(I, D_t) - \sigma \tag{2.8}$$

そして、負例における出現数 $sup(I, D_o)$ を y 軸に，正例における出現数 $sup(I, D_t)$ を x 軸としたとき，顕在パターンは図 2.1 の網かけで示された領域に属するパターンであり，コントラストパターンは図 2.2 の網かけで示された領域に属するパターンである．

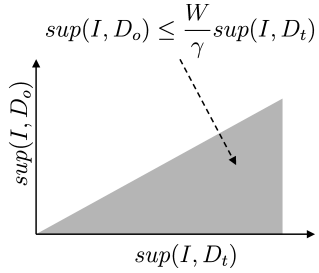

図 2.1 顕在パターン

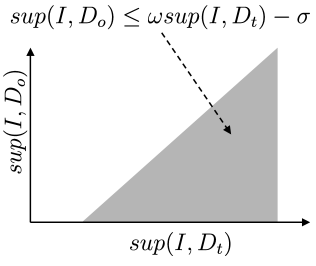

図 2.2 コントラストパターン

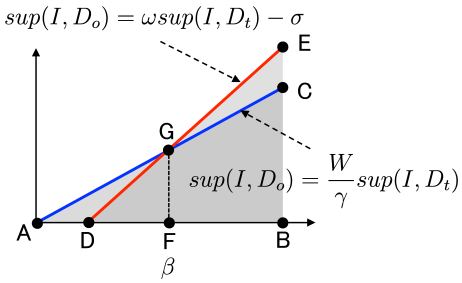

図 2.3 2 つのパターンの関係

LCM では，ユーザがパラメータ ω, σ を与えることでコントラストパターンを高速に列挙することができる．コントラストパターンの列挙においては、$sup(I, D_t)$ が大きくなればなるほど、$sup(I, D_o)$ との差が相対的に小さいパターンが列挙されることもあり、そのようなパターンはクラス c_t に特徴的なパターンとは言えなくなる。この欠点を回避するために、本コマンドでは顕在パターンの列挙を採用している。

ここで問題となるのは、コントラストパターンを列挙する LCM を使って、いかに顕在パターンを列挙するかであり、以下に本コマンドで採用している方法を示す。

図 2.3 は、図 2.1 と図 2.2 を重ねた図である。ここでは顕在パターンの領域 ABC を全て列挙するのではなく、$sup(I, D_t) \geq \beta$(β は本コマンド S=で指定するパラメータ) を満たす領域 GFCB に属する顕在パターンの列挙を考える。LCM によって列挙されるパターンは、$\triangle DEB$ に属するパターンであるが、直線 DE は、σ と ω を定めることによって決まる。まず σ は、直線 AC と直線 DE の交点 G の x 座標が、ちょうど β になるように設定する (式 (2.9):ω

の決め方は後述)。そうすることで、LCM が列挙した全パターンから、$\triangle DFG$ および $\triangle EGC$ に属するパターンを削除すれば、目的とする顕在パターンを列挙することが可能となる。

$$\sigma = \beta(\omega - \frac{W}{\gamma}) \tag{2.9}$$

次に ω であるが、これは直線 DE の傾きを決めることに対応する。一般的に $\triangle EGC$ に属するパターンより、$\triangle DGF$ に属するパターンの方が遥かに多いので、ω をできる限り大きくする方が効率的である。

しかしながら、ω はトランザクションの重みであり、計算機上で件数をカウントする変数の型の最大値に制約される。その最大値を $maxInt$ とすると、$\omega sup(I, D_t) \leq maxInt$ の制約を満たしていなければならない。$sup(I, D_t)$ は $|D_p|$ を超えることはないので、ω を式 (2.10) のとおり指定することで、$\triangle DFG$ の列挙数を最小化できる。

$$\omega = \frac{maxInt}{|D_t|} \tag{2.10}$$

2.2 msequence.rb 頻出系列パターンの列挙

アイテム系列データから頻出系列パターンを列挙する。アイテム系列データとは、順序付けられたアイテム列の集合であり、本コマンドは、アイテム系列データに多頻度に出現する部分系列をパターンとして列挙する。系列パターンが系列データに出現 (もしくはマッチ) するとは、パターンを構成するアイテム全てが、その順序で系列データ上に現れることを意味する。列挙のコアアルゴリズムには LCMseq(LCM algorithm for enumerating all frequently appearing sequences) を用いている [3]。本コマンドは以下のような特徴を持つ。

- gap 長と window 幅の制約条件 (上限値) を与えることができる。
- gap 長と window 幅の制約条件 (上限値) を時間制約として与えることも可能。
- アイテムの分類階層 (taxonomy) を用いることができる。
- 分類クラスを指定することで、あるクラスに特徴的なパターン (顕在系列パターン) を列挙することができる。3 つ以上のクラスにも対応している。
- アイテム集合のシーケンス (同じ時刻に異なる複数のアイテム) は扱うことができない。

表 2.13 に、本コマンドが扱う入力データ例を示す。tid 項目によってひとつのシーケンスを識別し、time 項目で item 項目の順序を表す。同じ時刻に複数の item を含めることはできない (同時刻に複数の item が存在した場合の動作は不定である)。ただし、time 項目は基本的には item の順序を決めるためのみに利用される。`-padding` オプションを指定した時のみ、整数で与えられた time 項目の値に応じた gap 長や window 幅の設定が可能となる (後述)。また、表 2.13 のデータを、わかりやすさのためにアイテムの順序、及び時刻別の順序として表したデータを、それぞれ表 2.14 および表 2.15 に示す。

表 2.13 系列データ

tid	time	item
T1	0	C
T1	2	B
T1	3	A
T1	7	C
T2	2	D
	:	

表 2.14 ベクトル型で表示

	シーケンス
T1	C B A C
T2	D A B C
T3	C B D E
T4	A C B
T5	B A D D C C
T6	A B D B C

表 2.15 時刻を考慮した表示

	0	1	2	3	4	5	6	7	8	9
T1	C		B	A				C		
T2			D	A		B	C			
T3		C	B		D				E	
T4			A				C			B
T5	B	A	D	D				C		C
T6	A					B	D		B	C

このデータについて 2 件以上出現する系列パターンは (A C),(B C),(D B C) など 20 件存在する (例 1 を参照)。系列パターン (B C) は、tid が T1,T2,T5,T6 のレコードに出現している。T3、T4 はアイテム B と C の両方を含んでいるが、出現順序が違うために出現したことにはならない。

頻出系列パターン

頻出系列パターンとは、出現頻度 (サポートと呼ぶ) がユーザの与えた最小サポート以上であるような系列パターンのことを言う。最小サポートが 3 件とすると、系列パターン (B D) は、系列データ T3,T5,T6 の 3 件に出現しているので頻出であるが順序を逆にした系列パターン (D B) は、系列データ T2,T6 の 2 件にしか出現しないので頻出ではない。ここで最小サポート 3 件を満たす全ての頻出系列パターンと、その出現件数を表 2.16 に示す。

顕在系列パターン

各データが属する「クラス」を導入し、あるクラスに特徴的な系列パターンを列挙する。ここで特徴的とは、あるクラスには多頻度で、他のクラスでは多頻度でないことである。例えば、スーパーマーケットでは、男性と女性で購買されるアイテムの順序の違いを識別したい時などに使われる。顕在系列パターンの列挙例は例 5 を、また、顕在系列パターンの評価指標として用いられる増加率や事後確率については、mitemset.rb コマンドの資料 1 を参照のこと。

表 2.16　表 2.14 において最小サポート 3 件を満たす全頻出系列パターンとその出現トランザクション

系列パターン	出現件数	出現トランザクション
C	6	T1,T2,T3,T4,T5,T6
B	6	T1,T2,T3,T4,T5,T6
A	5	T1,T2,T4,T5,T6
A C	5	T1,T2,T4,T5,T6
B C	4	T1,T2,T5,T6
D	4	T2,T3,T5,T6
A B	3	T2,T4,T6
B D	3	T3,T5,T6
C B	3	T1,T3,T4
D C	3	T2,T5,T6

階層分類

アイテムの階層分類を反映させることができる。詳細は mitemset.rb コマンドを参照のこと。

gap 長上限

gap 長とは、系列パターンの隣接する任意の 2 アイテムについて、系列データ上でマッチした部分系列の距離 (2 アイテム間のアイテム数 −1) として定義される。例えば、系列パターン (A B C) と系列データ (A D D D B D C) では、パターン上で隣接する 2 アイテム AB 間の系列データ上での gap 長は 4 で、BC 間の gap 長は 2 である。gap 長上限を指定することによって、系列パターンの「出現」の定義を、パターン上の任意の gap 長が指定した上限以下であるように制約する。なお、複数のマッチが存在する場合は、いずれかのマッチが制約を満たしていれば出現したと考える。gap 長上限を 1 に設定すると、データ上で隣接する頻出系列パターンを列挙することになる。gap 長の計算例を表 2.17 に示す。

window 幅上限

window 幅とは、マッチした系列データ上の部分系列について、その始点から終点までの長さ (アイテム数) である。例えば、パターン (A B C) と系列データ (C A D C B D C) を考えると、データ上のマッチした始点が 2 番目のアイテムで、終点が 7 番目目のアイテムとなり、window 幅は 6 となる。window 幅上限を指定することによって、パターンの出現の定義を、データ上のマッチ幅が指定した上限以下であるように制約する。なお、複数のマッチが存在する場合は、いずれかのマッチが制約を満たしていれば出現したと考える。window 幅の計算例を表 2.17 に示す。

時間制約

LCMseq では、アイテムの出現時刻を直接指定して gap 長制約や window 幅制約を指定することができない。そこで、前処理でアイテムが存在しない時刻に架空のアイテム (”!”:エクスクラメーションマーク) を導入することで時間制約を実現する。[*1] 例えば、表 2.15 に示された系列データは、表 2.18 に示されるようなデータに変換される。そして、架空アイテムを挿入した系列データに対して gap 長制約や window 制約を指定して系列パターンを列挙する。そして最終の出力時に架空アイテムを含む系列パターンの出力を抑制する。

出力

本コマンドが出力するデータは、大きく分けて 2 つあり、一つは、列挙された系列パターンデータ (`patterns.csv`)、そして他方は、それらのパターンをどのトランザクションが含むかについての情報 (`tid_pats.csv` である。パターン

[*1] そのため、アイテムの文字列として”!”を利用することはできない。

表 2.17 パターン (A B C) がマッチする位置と、その gap 長と window 幅。下の 4 行は、系列データ AAABCC について 4 通りのマッチが存在し、それら全てのマッチについての gap 長と window 幅を示している。これらの gap 長もしくは window 幅の一つでも条件にマッチすれば出現したとみなされる。例えば、window 上限を 3 に設定した場合は、パターン ABC は出現したことになるが、上限を 2 にすると出現したことにはならない。

系列データ	A-B 間 gap 長	B-C 間 gap 長	window 幅
A D D D D **B** D **C** D	5	2	8
A **B** **C** D	1	1	3
C **A** A C **B** A **C** C	3	2	6
C **A** C **B** B A **C** B C	2	3	6
A A **B** **C** C	2	1	4
A **A** **B** **C** C	1	1	3
A A **B** C **C**	2	2	5
A **A** **B** C **C**	1	2	4

表 2.18 時間を考慮した gap 長制約と window 制約を実現するために、アイテムの存在しない時刻に架空のアイテム"!"を挿入する。

	シーケンス
T1	C ! B A ! ! ! C
T2	D A ! B C
T3	C B ! D ! ! ! E
T4	A ! ! ! C ! ! B
T5	B A D D ! ! ! C ! C
T6	A ! ! ! ! B D B ! C

データは、顕在系列パターンの場合異なる CSV 項目を出力する。表 2.19〜表 2.21 にそれらのサンプルを示す。

表 2.19 patterns.csv のデータ例。pid 項目は一つの系列パターンを識別するための ID で、size はパターンとしてのアイテム集合を構成するアイテム数、count はそのパターンが出現した系列データ数、そして total は全系列データ数である。support は出現確率で、count/total で計算される。最後に pattern が系列パターンで、アイテムは半角スペースで区切られている。

pid	pattern	size	count	total	support
1	C	1	6	6	1
4	B	1	6	6	1
11	A C	2	5	6	0.8333333333
10	A	1	5	6	0.8333333333
16	D	1	4	6	0.6666666667
7	B C	2	4	6	0.6666666667
12	A B	2	3	6	0.5
2	C B	2	3	6	0.5
19	D C	2	3	6	0.5
3	C C	2	2	6	0.3333333333
:	:	:	:	:	:

表 2.20 tidpats.csv の内容例。tid は系列データ ID で、tid=パラメータで指定した入力データ項目に対応している。そしてそれぞれの系列データに含まれる系列パターンの ID が pid で示されている。

tid	pid
T1	0
T1	1
T1	10
T1	2
T2	0
T2	1
T2	10
T2	11
T3	0
T3	1
:	:

2.2.1 書式

```
msequence.rb i= [x=] [O=] [tid=] [item=] [class=] [taxo=] [s=|S=] [sx=|SX=] [l=] [u=]
```

表 2.21　顕在パターンにおける patterns.csv の内容例。class 項目は、どのクラスに特徴的な顕在パターンか、その対象クラスを示している。pid,pattern,size,total は表 2.19 と同様である。pos は対象クラスのトランザクションに出現した件数で、neg はそれ以外のクラスのトランザクションに出現した件数である。posTotal,negTotal は、対象クラスとそれ以外のクラスのトランザクション件数である。support は、対象クラスにおける出現確率で、pos/posTotal で計算される。growthRate は増加率で、support/(neg/negTotal) で計算される値である。分母が 0 の場合は inf と表示される。この値が大きいほど、対象クラスに特徴的であることを意味する。postProb は、パターンを条件とした時の対象クラスの事後確率で、growthRate と同様、この値が大きいほど対象クラスに特徴的であることを意味する。詳細な定義は mitemset.rb コマンドの解説の資料 1 を参照のこと。

class	pid	pattern	size	pos	neg	posTotal	negTotal	total	support	growthRate	postProb
cls1	1	B C	2	3	0	4	2	6	0.75	inf	1
cls2	9	B C D	3	2	0	2	4	6	1	inf	1
cls2	10	A D	2	2	0	2	4	6	1	inf	1
cls2	11	A C D	3	2	0	2	4	6	1	inf	1
cls2	8	B D	2	2	1	2	4	6	1	4	0.6666666667
cls2	12	C D	2	2	1	2	4	6	1	4	0.6666666667

```
[gap=] [win=] [p=] [g=] [top=] [-padding] [T=] [--help]
```

`i=`　key **型トランザクションデータファイル名【必須】**

`c=`　**クラスファイル名【オプション】**

`x=`　**階層分類データファイル名【オプション】**

`O=`　**出力パス名【オプション**:default=./take_#{**日付時刻**}**】**

`tid=`　**トランザクション** ID **項目名【必須】**

`time=`　**トランザクション** ID **項目名【必須】**

`item=`　**時間項目名（**i=**上の項目名）【オプション**:default="time"**】**

`class=`　**クラス項目名（**c=**上の項目名）【オプション】**
クラス項目名を指定すると顕在パターンが列挙される。

`taxo=`　**分類項目名【条件付き必須**:x=**】**

`s=`　**最小サポート（確率）【選択必須**:s=, S=**】**

`S=`　**最小サポート（件数）【選択必須**:s=, S=**】**

`sx=`　**最大サポート（確率）【オプション】**

`SX=`　**最大サポート（件数）【オプション】**

`l=`　**最小アイテム集合サイズ【オプション】**

`u=`　**最大アイテム集合サイズ【オプション】**

`gap=`　**パターンのギャップ長の上限（**0 **以上の整数）【オプション**:0 **で制限無し**,default:0**】**

`win=`　**パターンの窓サイズの上限（**0 **以上の整数）【オプション**:0 **で制限無し**,default:0**】**

`p=`　**顕在パターン用最小事後確率【オプション**:default=0.5**】**

`g=`　**顕在パターン用最小増加率【オプション】**

`top=`　**列挙するパターン数の上限【オプション**:default:**制限なし】**

`-padding`　**時刻を整数とみなし、連続でない時刻に特殊なアイテムがあることを想定する。**
gap=**や** win=**の指定に影響する。**

`T=`　**ワークディレクトリ【オプション】**

`--help`　**ヘルプの表示**

2.2.2 利用例

例 1: **基本例**

2 件以上で出現する系列パターン。入力データの項目名は、全てデフォルトのものと同じなので省略していることに注意する。

```
$ more dat1.csv
tid,time,item
T1,0,C
T1,2,B
T1,3,A
T1,7,C
T2,2,D
T2,3,A
T2,5,B
T2,6,C
T3,1,C
T3,2,B
T3,4,D
T3,8,E
T4,2,A
T4,5,C
T4,6,B
T5,0,B
T5,1,A
T5,2,D
T5,3,D
T5,7,C
T5,9,C
T6,0,A
T6,5,B
T6,6,D
T6,8,B
T6,9,C
$ msequence.rb  O=result1 i=dat1.csv S=2
#MSG# lcm_seq_20140215 CIf /tmp/__MTEMP_96779_70117174398360_0 2 /tmp/__MTEMP_96779_701171
74386400_0
trsact: /tmp/__MTEMP_96779_70117174398360_0 ,#transactions 6 ,#items 5 ,size 26 extracted
database: #transactions 6 ,#items 4 ,size 25
 output to: /tmp/__MTEMP_96779_70117174386400_0
iters=20
20
1
4
10
5
#MSG# output tid-patterns ...
#MSG# the number of contrast patterns enumerated is 19
#MSG# The final results are in the directory ‘result1’
#END# /usr/bin/msequence.rb O=result1 i=dat1.csv S=2
$ more result1/patterns.csv
pid,pattern,size,count,total,support
1,C,1,6,6,1
4,B,1,6,6,1
11,A C,2,5,6,0.8333333333
10,A,1,5,6,0.8333333333
16,D,1,4,6,0.6666666667
7,B C,2,4,6,0.6666666667
12,A B,2,3,6,0.5
8,B D,2,3,6,0.5
2,C B,2,3,6,0.5
19,D C,2,3,6,0.5
```

```
3,C C,2,2,6,0.3333333333
18,D B C,3,2,6,0.3333333333
13,A B C,3,2,6,0.3333333333
14,A D,2,2,6,0.3333333333
15,A D C,3,2,6,0.3333333333
9,B D C,3,2,6,0.3333333333
17,D B,2,2,6,0.3333333333
6,B A C,3,2,6,0.3333333333
5,B A,2,2,6,0.3333333333
```

例 2: **パターン長の制限**

出現頻度が 3 以上の長さが 2 の系列パターンのみを列挙する。

```
$ msequence.rb  O=result2 i=dat1.csv S=3 l=2 u=2
#MSG# lcm_seq_20140215 CIf -l 2 -u 2 /tmp/__MTEMP_96838_70227485417240_0 3 /tmp/__MTEMP_96
838_70227485416360_0
trsact: /tmp/__MTEMP_96838_70227485417240_0 ,#transactions 6 ,#items 5 ,size 26 extracted
database: #transactions 6 ,#items 4 ,size 25
 output to: /tmp/__MTEMP_96838_70227485416360_0
iters=11
6
0
0
6
#MSG# output tid-patterns ...
#MSG# the number of contrast patterns enumerated is 6
#MSG# The final results are in the directory ‘result2’
#END# /usr/bin/msequence.rb O=result2 i=dat1.csv S=3 l=2 u=2
$ more result2/patterns.csv
pid,pattern,size,count,total,support
3,A C,2,5,6,0.8333333333
1,B C,2,4,6,0.6666666667
0,C B,2,3,6,0.5
2,B D,2,3,6,0.5
4,A B,2,3,6,0.5
5,D C,2,3,6,0.5
$ more result2/tid_pats.csv
tid,pid
T1,0
T1,1
T1,3
T2,1
T2,3
T2,4
T2,5
T3,0
T3,2
T4,0
T4,3
T4,4
T5,1
T5,2
T5,3
T5,5
T6,1
T6,2
T6,3
T6,4
T6,5
```

例 3: gap **長と** window **サイズの指定**

出現頻度が 2 以上、長さが 2 以上の系列パターンのうち、gap 長が 2、window サイズが 4 のパターンを列挙する。

```
$ msequence.rb  O=result3 i=dat1.csv S=2 l=2 gap=2 win=4
#MSG# lcm_seq_20140215 CIf -l 2 -g 2 -G 4 /tmp/__MTEMP_96898_70310565879140_0 2 /tmp/__MTE
MP_96898_70310565878100_0
trsact: /tmp/__MTEMP_96898_70310565879140_0 ,#transactions 6 ,#items 5 ,size 26 extracted
database: #transactions 6 ,#items 5 ,size 26
 output to: /tmp/__MTEMP_96898_70310565878100_0
iters=15
10
0
0
9
1
#MSG# output tid-patterns ...
#MSG# the number of contrast patterns enumerated is 10
#MSG# The final results are in the directory 'result3'
#END# /usr/bin/msequence.rb O=result3 i=dat1.csv S=2 l=2 gap=2 win=4
$ more result3/patterns.csv
pid,pattern,size,count,total,support
0,C B,2,3,6,0.5
2,B C,2,3,6,0.5
3,B D,2,3,6,0.5
4,A C,2,3,6,0.5
5,A B,2,3,6,0.5
1,B A,2,2,6,0.3333333333
6,A D,2,2,6,0.3333333333
7,D B,2,2,6,0.3333333333
8,D B C,3,2,6,0.3333333333
9,D C,2,2,6,0.3333333333
```

例 4: padding **により時刻を考慮する**

例 3 と同じ条件で、-padding を指定することで、時間を考慮した gap 長と window サイズ制約により系列パターンを列挙する。

```
$ msequence.rb  O=result4 i=dat1.csv S=2 l=2 gap=2 win=4 -padding
#MSG# lcm_seq_zero_20140215 CIf -l 2 -g 2 -G 4 /tmp/__MTEMP_96957_70107510988120_0 2 /tmp/
__MTEMP_96957_70107510987120_0
trsact: /tmp/__MTEMP_96957_70107510988120_0 ,#transactions 6 ,#items 6 ,size 46 extracted
database: #transactions 6 ,#items 6 ,size 46
 output to: /tmp/__MTEMP_96957_70107510987120_0
iters=33
4
0
0
4
#MSG# output tid-patterns ...
#MSG# the number of contrast patterns enumerated is 4
#MSG# The final results are in the directory 'result4'
#END# /usr/bin/msequence.rb O=result4 i=dat1.csv S=2 l=2 gap=2 win=4 -padding
$ more result4/patterns.csv
pid,pattern,size,count,total,support
0,C B,2,3,6,0.5
3,B D,2,3,6,0.5
1,B A,2,2,6,0.3333333333
2,B C,2,2,6,0.3333333333
```

例 5: **顕在系列パターンの列挙**

例 1 と同じ条件で、クラス項目を指定することで顕在パターンを列挙する。

```
$ more dat2.csv
tid,time,item,class
T1,0,C,cls1
T1,2,B,cls1
T1,3,A,cls1
T1,7,C,cls1
T2,2,D,cls1
T2,3,A,cls1
T2,5,B,cls1
T2,6,C,cls1
T3,1,C,cls1
T3,2,B,cls1
T3,4,D,cls1
T3,8,E,cls1
T4,2,A,cls1
T4,5,C,cls1
T4,6,B,cls1
T5,0,B,cls2
T5,1,A,cls2
T5,2,D,cls2
T5,3,D,cls2
T5,7,C,cls2
T5,9,C,cls2
T6,0,A,cls2
T6,5,B,cls2
T6,6,D,cls2
T6,8,B,cls2
T6,9,C,cls2
$ msequence.rb  O=result5 i=dat2.csv S=2 class=class -padding
#MSG# lcm_seq_zero_20140215 CIA -w /tmp/__MTEMP_97018_70131049887220_1 /tmp/__MTEMP_97018_
70131049887220_0 1073741815 /tmp/__MTEMP_97018_70131049885920_0
trsact: /tmp/__MTEMP_97018_70131049887220_0 ,#transactions 6 ,#items 6 ,size 46 extracted
database: #transactions 6 ,#items 5 ,size 45 ,weightfile /tmp/__MTEMP_97018_70131049887220
_1
 output to: /tmp/__MTEMP_97018_70131049885920_0
iters=33
9
1
4
4
#MSG# output patterns to CSV file ...
#MSG# the number of contrast patterns on class ‘cls1’ enumerated is 8
#MSG# output tid-patterns ...
#MSG# lcm_seq_zero_20140215 CIA -w /tmp/__MTEMP_97018_70131049887220_2 /tmp/__MTEMP_97018_
70131049887220_0 2147483645 /tmp/__MTEMP_97018_70131049885920_2
trsact: /tmp/__MTEMP_97018_70131049887220_0 ,#transactions 6 ,#items 6 ,size 46 extracted
database: #transactions 6 ,#items 5 ,size 45 ,weightfile /tmp/__MTEMP_97018_70131049887220
_2
 output to: /tmp/__MTEMP_97018_70131049885920_2
iters=36
5
0
0
3
2
#MSG# output patterns to CSV file ...
#MSG# the number of contrast patterns on class ‘cls2’ enumerated is 5
#MSG# output tid-patterns ...
#MSG# the number of emerging sequence patterns enumerated is 13
#MSG# The final results are in the directory ‘result5’
```

```
#END# /usr/bin/msequence.rb O=result5 i=dat2.csv S=2 class=class -padding
$ more result5/patterns.csv
class,pid,pattern,size,pos,neg,posTotal,negTotal,total,support,growthRate,postProb
cls1,1,B C,2,3,0,4,2,6,0.75,inf,1
cls2,10,A D,2,2,0,2,4,6,1,inf,1
cls2,9,B C D,3,2,0,2,4,6,1,inf,1
cls2,11,A C D,3,2,0,2,4,6,1,inf,1
cls1,0,C,1,4,2,4,2,6,1,1,0.6666666667
cls1,2,B,1,4,2,4,2,6,1,1,0.6666666667
cls1,6,A B,2,2,1,4,2,6,0.5,1,0.6666666667
cls2,8,B D,2,2,1,2,4,6,1,4,0.6666666667
cls2,12,C D,2,2,1,2,4,6,1,4,0.6666666667
cls1,5,A C,2,3,2,4,2,6,0.75,0.75,0.6
cls1,4,A,1,3,2,4,2,6,0.75,0.75,0.6
cls1,7,D,1,2,2,4,2,6,0.5,0.5,0.5
cls1,3,B C,2,2,2,4,2,6,0.5,0.5,0.5
```

2.3　mpolishing.rb 一般グラフの研磨

一般グラフデータに対して、データ研磨アルゴリズムを適用することで、ノイズを除去し、グラフの濃淡をより鮮明にした「研磨グラフ」を生成する。まずは、直感的な理解を得るために、図 2.4 と図 2.5 に研磨前のグラフと研磨後のグラフを示す。オリジナルのグラフについて、濃い部分構造はより濃く、そして薄い部分構造はより薄くなるように研磨される。結果として、中規模の極大クリーク (他の完全部分グラフに包含されない完全部分グラフ) が多く生成される。

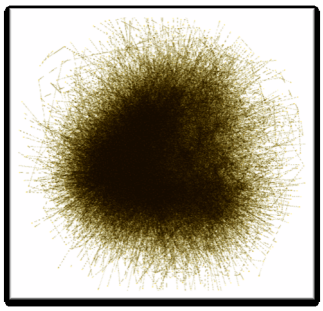

図 2.4　研磨前のグラフ

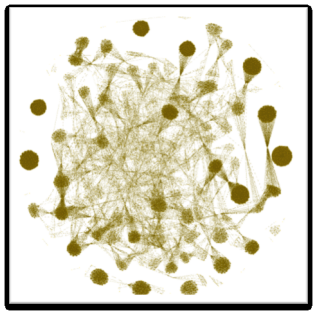

図 2.5　研磨後のグラフ

研磨のアルゴリズムを Algorithm 1 に示す。ここに示されたアルゴリズムは、非常に効率の悪い方法ではあるが、理解のし易さを優先させている。実際に内部で実装されているアルゴリズムについては参考文献 [2] に詳しい。研磨の方法は至ってシンプルで、全ての頂点ペアについて、その類似度 (後述) がユーザの指定した閾値以上であれば接続し、そうでなければ接続しないというルールに従って、新たなグラフを再構成する。

Algorithm 1 グラフ研磨アルゴリズム

```
1: function POLISHING(G = (V, E), σ)
2:     V : 頂点集合, E : 辺集合, σ : 類似度下限値
3:     E' = φ; V' = φ                          ▷ 研磨後の辺集合と頂点集合の初期化
4:     for all u ∈ V do
5:         for all v ∈ V do                    ▷ # 全頂点ペア u, v について調べる
6:             if sim(u, v) ≥ σ then           ▷ 頂点ペア u, v が似ていればには新たに辺として加え、似てなければ加えない
7:                 E' = E' ∪ (u, v)
8:                 V' = V' ∪ u
9:                 V' = V' ∪ v
10:            end if
11:        end for
12:    end for
13:    return(V', E')
14: end function
```

そして新たに構成されたグラフを入力として同様の研磨手法を繰り返し適用し、グラフの構成に変化がなくなるか、もしくはユーザの指定した最大繰り返し回数に達すれば終了する。最終的に得られたグラフが研磨グラフである。

次に 2 つの頂点間の類似度 (Algorithm 1 の 6 行目:$sim(u, v)$) の定義を示す。基本的な考え方は「共通の友達が多い友達は友達と考えよう」というものである。頂点 u, v に隣接する頂点が図 2.6 に示されるような構造を持っていたとしよう。u と v に直接の接続はないが、いずれかの頂点に接続された 5 つの頂点のうち、頂点 3,4 の 2 つの頂点が共通して接続されている。この情報を利用して類似度は定義される。

本コマンドでは、表 2.22 に示されるように、6 つの類似度を利用することができる。類似度の選択は表中に示された `sim=`パラメータを設定することによって行い、`th=`パラメータによって類似度の下限値を与える。

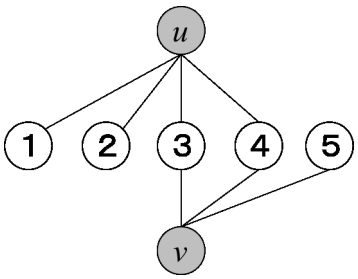

図 2.6 頂点 u, v の接続関係

表 2.22 グラフ $G = (V, E)$ における節点 u, v の類似度の定義

#	類似度	定義式	sim=パラメータ値	範囲
1	recemblance	$\frac{\|N(u)\cap N(v)\|}{\|N(u)\cup N(v)\|}$	R	0.0〜1.0
2	normalized PMI	$\log \frac{P(u,v)}{P(u)P(v)} / (-\log P(u,v))$ $= \frac{\|V\|\|N(u)\cap N(v)\|}{\|N(u)\|\|N(v)\|} / (-\log \frac{\|N(u)\cap N(v)\|}{\|V\|})$	P	-1.0〜1.0
3	intersection	$\|N(u) \cap N(v)\|$	T	0〜
4	cosine	$\frac{\|N(u)\cap N(v)\|}{\sqrt{\|N(u)\|}\sqrt{\|N(v)\|}}$	C	0.0〜1.0
5	max-confidence	$\frac{\|N(u)\cap N(v)\|}{\max(\|N(u)\|,\|N(v)\|)}$	S	0.0〜1.0
6	min-confidence	$\frac{\|N(u)\cap N(v)\|}{\min(\|N(u)\|,\|N(v)\|)}$	s	0.0〜1.0

$N(u)$ は頂点 u に隣接する頂点集合を表す。$P(u)$ は頂点 u に枝が張られる確率を表し、$P(u) = N(u)/|V|$ である。

2.3.1 例

本コマンドの入力データである一般グラフは、表 2.23 に示されるような、枝データを節点ペアとして表した CSV 形式で与える。グラフは無向グラフとして扱われ、島が複数あってもよい。枝を一つも持たない節点は、研磨の対象とならない (友達がいない) ために、節点データを別途入力することはない。表 2.23 のデータを視覚化したグラフを図 2.7 に示す。以下では、このデータを使いデータ研磨の例を示していく。

表 2.23 入力データ (枝データ)

node1	node2
a	b
a	c
a	e
b	c
b	e
b	g
c	d
c	g
d	e
e	f

簡単のために、類似度の定義を intersection としその下限値を 2 に設定したとしよう。すなわち、隣接する節点で共通の節点数 (共通する友達の数) が 2 以上である時に枝を張り、そうでなければ枝を張らない。例えば、図 2.8 は、図 2.7 から節点 a, b と連結のある節点を抜き出した部分グラフである。2 つの節点 e, c を共通に持つため、研磨グラフにおいて、節点 a, b に枝が張られる。一方で節点 e, g(図 2.9) は、共通節点は節点 b だけなので、研磨グラフにおいても

枝は張られない。

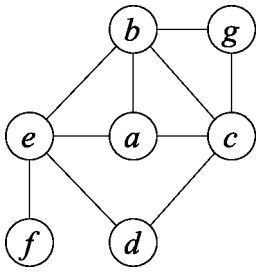

図 2.7　表 2.23 のグラフデータを視覚化

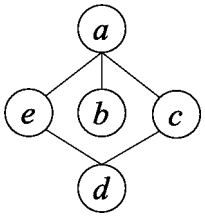

図 2.8　節点 a と節点 d の関係

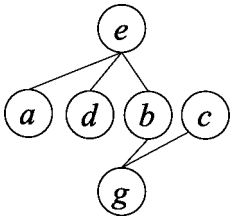

図 2.9　節点 e と節点 g の関係

図 2.10 は、節点 a, b に関連する部分グラフである。これまでの 2 つの例と異なる点は、節点 a, b に直接の接続があることである。このような場合、 2 つの考え方を取ることができる。ひとつは、図 2.11 のように、直接の関係は考慮せず、あくまでも共通の友達関係だけを考慮する方法である。もう一つの方法は、節点 a, b をお互いに共通の友達と考え、架空の節点 a', b' を追加し、a, b 双方からの接続を仮定する。よって、直接の接続がある場合、それだけで 2 人の友達を共通に持つと考える。

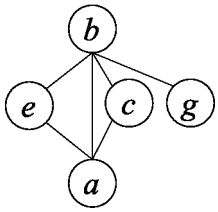

図 2.10　節点 a と節点 b の関係

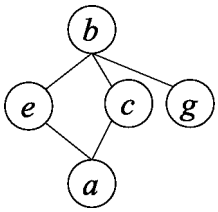

図 2.11　直接の接続を考慮しない例

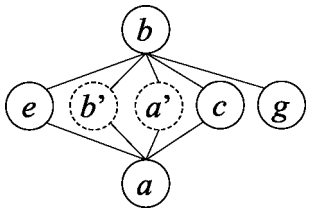

図 2.12　直接の接続を考慮する例

以上のようにすべての節点ペアについて新たな接続関係を更新すると、直接の関係を考慮しない場合は図 2.13 のような研磨グラフが得られ、直接の関係を考慮する場合は図 2.15 のような研磨グラフが得られる。そして、新たに得られたこれらの研磨グラフに繰り返しデータ研磨を付していくと、3 回でグラフ構造は変化しなくなり (収束し)、図 2.14 と図 2.16 が得られる。直接接続を考慮しない場合は、接続関係がすべて無くなる。一方で直接接続を考慮する場合は、6 つの節点 a, b, c, d, e, g が互いに接続され、極大クリークが形成される。また節点 f は e とのみ接続が残り、これも極大クリーク e, f を形成している。なお、データ研磨を繰り返し適用すると、グラフ構造は安定してくるが、収束しないケースもあることに注意する。

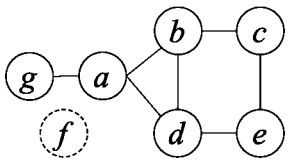

図 2.13　直接考慮しない 1 回研磨グラフ

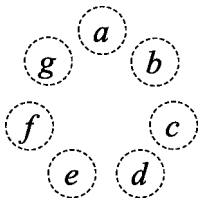

図 2.14　直接考慮しない 3 回研磨グラフ

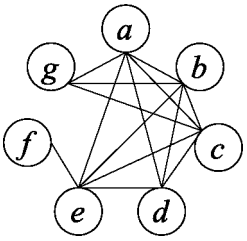

図 2.15　直接考慮した 1 回研磨グラフ

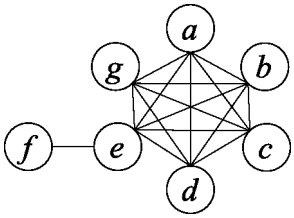

図 2.16　直接考慮した 3 回研磨グラフ

以上は理解のし易さのために、共通する枝の数を類似度と定義してデータ研磨を行ったが、グラフのサイズが大きくなると、思ったような研磨ができないことが多い。隣接節点の数が増えるに伴って巨大な次数の頂点が発生し、関係が薄いにも関わらず十分な数の共通接点を持つ節点ペアにも枝が張られてしまうからである。そこで、通常は、resemblance を始めとした、相対的な共通性を評価できる類似度を使う。類似度として resenmblance を用い、その下限値を 0.4,0.5 として収束するまで研磨したグラフを図 2.17,2.18 にそれぞれ示している。また図 2.19 は、類似度とし

て normalized PMI を、下限値を 0.2 とした結果である。

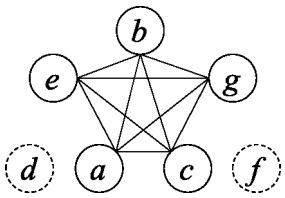

図 2.17 recemblance=0.4 による研磨グラフ

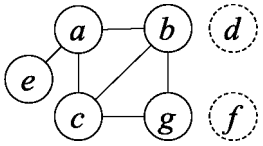

図 2.18 recemblance=0.5 による研磨グラフ

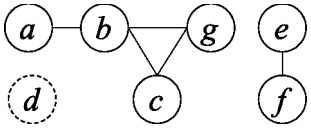

図 2.19 normalized PIM=0.2 による研磨グラフ

なお、出力としては研磨グラフ以外にも表 2.24 に示されるような各種統計を出力することができる (log=で指定したファイル)。

表 2.24 ログに出力される各種統計

key	内容
iter	データ研磨を適用した回数
time	実行時間
nSize0	データ研磨前のグラフの節点数
eSize0	データ研磨前のグラフの枝数
dens0	データ研磨前の枝密度
nSize#	#回目のデータ研磨後の節点数
eSize#	#回目のデータ研磨後の枝数
dens#	#回目のデータ研磨後の枝密度

以下にデータ研磨の特徴をまとめておく。

- 共通する隣接節点の情報にしたがって枝を張り直す。
- 6 つの類似度を選ぶことができ、利用する類似度によって結果は異なってくる。
- データ研磨を繰り返すことでグラフ構造は安定してくる (多くの場合は収束する)。
- オリジナルのグラフに比べ極大クリークの数が劇的に少なくなる。
- 類似度の下限値を低くすると、より大きな極大クリークが形成される。

2.3.2 書式

```
mpolishing.rb ei= ef= [ni=] [nf=] eo= [no=] [sim=R|P|C|s|S|T] th= [sup=] [-indirect] [iter=] [log=] [T=] [
```

ファイル名指定

`ei=` : **枝データファイル**

`ef=` : **枝データ上の 2 つの節点項目名 (省略時は**`"node1,node2"`**)**

`ni=` : **節点データファイル**

`nf=` : **節点データ上の節点項目名 (省略時は**`"node"`**)**

`eo=` : **データ研磨後の枝データファイル**

`no=` : **データ研磨後の節点データファイル**

`sim=` : **節点** `a,b` **と接続された枝集合を、それぞれ** `A,B` **とすると、節点** `a,b` **に枝を張るために用いる類似度。省略時は** `R` **が設定される。**

```
i: inclusion
```

```
            I: both-inclusion
            S: |A ∩ B|/max(|A|,|B|)
            s: |A ∩ B|/min(|A|,|B|)
            T (intersection): find pairs having common [threshld] items
            R (resemblance): find pairs s.t. |A\capB|/|A\cupB| >= [threshld]
            P (PMI): find pairs s.t. log (|A\capB|*|all| / (|A|*|B|)) >= [threshld]
            C (cosine distance): find pairs s.t. inner product of their normalized vectors >= [threshld]
```

th= : sim=で指定された類似度について、ここで指定された値以上の節点間に枝を張る。

sup= : 類似度計算において、|A ∩ B|>=sup の条件を加える。省略すれば sup=0。

-indirect: 上記類似度計算における隣接節点集合から直接の関係を除外する。
すなわち、A=A-b, B=B-a として類似度を計算する。

iter= : データ研磨の最大繰り返し数（デフォルト=30）

log= : パラメータの設定値や収束回数等を key-value 形式の CSV で保存するファイル名

O= : データ研磨過程のグラフを保存するディレクトリ名

その他

T= : ワークディレクトリ（default:/tmp）

--help : ヘルプの表示

2.3.3 利用例

例 1: 基本例

類似度を resemblance(sim=R) とし、th=0.4 で枝を張り直して得られた研磨グラフ。log1.csv を見ると研磨回数は 4 回で収束していることがわかる (iter,4)。出力結果は図 2.17 に示されるグラフ。

```
$ more dat1.csv
node1,node2
a,b
a,c
a,e
b,c
b,e
b,g
c,d
c,g
d,e
e,f
$ mpolishing.rb ei=dat1.csv ef=node1,node2 sim=R th=0.4 eo=result1.csv log=log1.csv
#MSG# converting graph files into a pair of numbered nodes ...
input-file /tmp/__MTEMP_95376_70244664277340_3, output-file /tmp/__MTEMP_95376_70244664277
340_4
degree threshold:
first & second scan end: /tmp/__MTEMP_95376_70244664277340_3
file read end: /tmp/__MTEMP_95376_70244664277340_3
iterative scan: #nodes=7, #edges = 20
forwardstar graph: /tmp/__MTEMP_95376_70244664277340_3 ,#nodes 7(7,7) ,#edges 20
#MSG# polishing iteration #0 (tra size=61
sspc_20140215 R -l 0 /tmp/__MTEMP_95376_70244664277340_4 0.4 /tmp/__MTEMP_95376_7024466427
7340_3
trsact: /tmp/__MTEMP_95376_70244664277340_4 ,#transactions 7 ,#items 7 ,size 27 extracted
database: #transactions 7 ,#items 7 ,size 27
 output to: /tmp/__MTEMP_95376_70244664277340_3
separated at 0
11 pairs are found
0,1,2,4, :1.000000  (0)
```

```
0,1,2,4,6, :1.000000  (0)
0,1,2,3,6, :1.000000  (0)
2,3,4, :1.000000  (0)
0,1,3,4,5, :1.000000  (0)
4,5, :1.000000  (0)
1,2,6, :1.000000  (0)
come
input-file /tmp/__MTEMP_95376_70244664277340_3, output-file /tmp/__MTEMP_95376_70244664277
340_4
degree threshold:
first & second scan end: /tmp/__MTEMP_95376_70244664277340_3
file read end: /tmp/__MTEMP_95376_70244664277340_3
iterative scan: #nodes=7, #edges = 22
forwardstar graph: /tmp/__MTEMP_95376_70244664277340_3 ,#nodes 7(7,7) ,#edges 22
#MSG# polishing iteration #1 (tra size=65
sspc_20140215 R -l 0 /tmp/__MTEMP_95376_70244664277340_4 0.4 /tmp/__MTEMP_95376_7024466427
7340_3
trsact: /tmp/__MTEMP_95376_70244664277340_4 ,#transactions 7 ,#items 7 ,size 29 extracted
database: #transactions 7 ,#items 7 ,size 29
 output to: /tmp/__MTEMP_95376_70244664277340_3
separated at 0
11 pairs are found
0,1,2,3,4,6, :1.000000  (0)
0,1,2,4,6, :1.000000  (0)
0,1,2,4,6, :1.000000  (0)
0,3, :1.000000  (0)
0,1,2,4,5, :1.000000  (0)
4,5, :1.000000  (0)
0,1,2,6, :1.000000  (0)
come
input-file /tmp/__MTEMP_95376_70244664277340_3, output-file /tmp/__MTEMP_95376_70244664277
340_4
degree threshold:
first & second scan end: /tmp/__MTEMP_95376_70244664277340_3
file read end: /tmp/__MTEMP_95376_70244664277340_3
iterative scan: #nodes=6, #edges = 22
forwardstar graph: /tmp/__MTEMP_95376_70244664277340_3 ,#nodes 7(7,7) ,#edges 22
#MSG# polishing iteration #2 (tra size=63
sspc_20140215 R -l 0 /tmp/__MTEMP_95376_70244664277340_4 0.4 /tmp/__MTEMP_95376_7024466427
7340_3
trsact: /tmp/__MTEMP_95376_70244664277340_4 ,#transactions 7 ,#items 7 ,size 28 extracted
database: #transactions 7 ,#items 7 ,size 28
 output to: /tmp/__MTEMP_95376_70244664277340_3
separated at 0
10 pairs are found
0,1,2,4,6, :1.000000  (0)
0,1,2,4,6, :1.000000  (0)
0,1,2,4,6, :1.000000  (0)
 :1.000000  (0)
0,1,2,4,5,6, :1.000000  (0)
4,5, :1.000000  (0)
0,1,2,4,6, :1.000000  (0)
come
input-file /tmp/__MTEMP_95376_70244664277340_3, output-file /tmp/__MTEMP_95376_70244664277
340_4
degree threshold:
first & second scan end: /tmp/__MTEMP_95376_70244664277340_3
file read end: /tmp/__MTEMP_95376_70244664277340_3
iterative scan: #nodes=5, #edges = 20
forwardstar graph: /tmp/__MTEMP_95376_70244664277340_3 ,#nodes 7(7,7) ,#edges 20
#MSG# polishing iteration #3 (tra size=57
sspc_20140215 R -l 0 /tmp/__MTEMP_95376_70244664277340_4 0.4 /tmp/__MTEMP_95376_7024466427
7340_3
trsact: /tmp/__MTEMP_95376_70244664277340_4 ,#transactions 7 ,#items 7 ,size 25 extracted
```

```
database: #transactions 7 ,#items 7 ,size 25
 output to: /tmp/__MTEMP_95376_70244664277340_3
separated at 0
10 pairs are found
0,1,2,4,6, :1.000000  (0)
0,1,2,4,6, :1.000000  (0)
0,1,2,4,6, :1.000000  (0)
 :1.000000  (0)
0,1,2,4,6, :1.000000  (0)
 :1.000000  (0)
0,1,2,4,6, :1.000000  (0)
come
input-file /tmp/__MTEMP_95376_70244664277340_3, output-file /tmp/__MTEMP_95376_70244664277
340_4
degree threshold:
first & second scan end: /tmp/__MTEMP_95376_70244664277340_3
file read end: /tmp/__MTEMP_95376_70244664277340_3
iterative scan: #nodes=5, #edges = 20
forwardstar graph: /tmp/__MTEMP_95376_70244664277340_3 ,#nodes 7(7,7) ,#edges 20
#MSG# converting the numbered nodes into original name ...
#END# /usr/bin/mpolishing.rb ei=dat1.csv ef=node1,node2 sim=R th=0.4 eo=result1.csv log=lo
g1.csv
$ more result1.csv
node1,node2
a,b
a,c
a,e
a,g
b,c
b,e
b,g
c,e
c,g
e,g
$ more log1.csv
key,value
ei=,dat1.csv
ef=,"node1,node2"
sim=,R
th=,0.4
eo=,result1.csv
log=,log1.csv
-indirect,false
iter,4
time,0.1502
nSize0,7
eSize0,10
dens0,0.4761904762
nSize1,7
eSize1,11
dens1,0.5238095238
nSize2,6
eSize2,11
dens2,0.7333333333
nSize3,5
eSize3,10
dens3,1
nSize4,5
eSize4,10
dens4,1
```

例 2: PMI **による研磨**

類似度を normalized PMI(sim=P) とし、th=0.2 で枝を張り直して得られた研磨グラフ。出力結果は図 2.19 に示されるグラフ。

```
$ mpolishing.rb ei=dat1.csv ef=node1,node2 sim=P th=0.2 eo=result2.csv
#MSG# converting graph files into a pair of numbered nodes ...
input-file /tmp/__MTEMP_95435_70301715200880_3, output-file /tmp/__MTEMP_95435_70301715200
880_4
degree threshold:
first & second scan end: /tmp/__MTEMP_95435_70301715200880_3
file read end: /tmp/__MTEMP_95435_70301715200880_3
iterative scan: #nodes=7, #edges = 20
forwardstar graph: /tmp/__MTEMP_95435_70301715200880_3 ,#nodes 7(7,7) ,#edges 20
#MSG# polishing iteration #0 (tra size=61
sspc_20140215 P -l 0 /tmp/__MTEMP_95435_70301715200880_4 0.2 /tmp/__MTEMP_95435_7030171520
0880_3
trsact: /tmp/__MTEMP_95435_70301715200880_4 ,#transactions 7 ,#items 7 ,size 27 extracted
database: #transactions 7 ,#items 7 ,size 27
 output to: /tmp/__MTEMP_95435_70301715200880_3
separated at 0
5 pairs are found
0,1,2,4, :1.000000  (0)
0,1,2,4,6, :1.000000  (0)
0,1,2,3,6, :1.000000  (0)
2,3,4, :1.000000  (0)
0,1,3,4,5, :1.000000  (0)
4,5, :1.000000  (0)
1,2,6, :1.000000  (0)
come
input-file /tmp/__MTEMP_95435_70301715200880_3, output-file /tmp/__MTEMP_95435_70301715200
880_4
degree threshold:
first & second scan end: /tmp/__MTEMP_95435_70301715200880_3
file read end: /tmp/__MTEMP_95435_70301715200880_3
iterative scan: #nodes=6, #edges = 10
forwardstar graph: /tmp/__MTEMP_95435_70301715200880_3 ,#nodes 7(7,7) ,#edges 10
#MSG# polishing iteration #1 (tra size=39
sspc_20140215 P -l 0 /tmp/__MTEMP_95435_70301715200880_4 0.2 /tmp/__MTEMP_95435_7030171520
0880_3
trsact: /tmp/__MTEMP_95435_70301715200880_4 ,#transactions 7 ,#items 7 ,size 16 extracted
database: #transactions 7 ,#items 7 ,size 16
 output to: /tmp/__MTEMP_95435_70301715200880_3
separated at 0
5 pairs are found
0,1, :1.000000  (0)
0,1,2,6, :1.000000  (0)
1,2,6, :1.000000  (0)
 :1.000000  (0)
4,5, :1.000000  (0)
4,5, :1.000000  (0)
1,2,6, :1.000000  (0)
come
input-file /tmp/__MTEMP_95435_70301715200880_3, output-file /tmp/__MTEMP_95435_70301715200
880_4
degree threshold:
first & second scan end: /tmp/__MTEMP_95435_70301715200880_3
file read end: /tmp/__MTEMP_95435_70301715200880_3
iterative scan: #nodes=6, #edges = 10
forwardstar graph: /tmp/__MTEMP_95435_70301715200880_3 ,#nodes 7(7,7) ,#edges 10
#MSG# converting the numbered nodes into original name ...
#END# /usr/bin/mpolishing.rb ei=dat1.csv ef=node1,node2 sim=P th=0.2 eo=result2.csv
$ more result2.csv
```

```
node1,node2
a,b
b,c
b,g
c,g
e,f
```

例 3: intersection で 1 回の研磨

前節の解説で用いてる例。類似度を intersection(sim=T) とし、2 件以上 (th=2) で枝を張り直し直接の接続を考慮に入れる例。研磨回数は 1 回のみ (iter=1)。出力結果は図 2.15 に示されるグラフ。

```
$ mpolishing.rb ei=dat1.csv ef=node1,node2 sim=T th=2 iter=1 eo=result3.csv
#MSG# converting graph files into a pair of numbered nodes ...
input-file /tmp/__MTEMP_95485_70155283659220_3, output-file /tmp/__MTEMP_95485_70155283659
220_4
degree threshold:
first & second scan end: /tmp/__MTEMP_95485_70155283659220_3
file read end: /tmp/__MTEMP_95485_70155283659220_3
iterative scan: #nodes=7, #edges = 20
forwardstar graph: /tmp/__MTEMP_95485_70155283659220_3 ,#nodes 7(7,7) ,#edges 20
#MSG# polishing iteration #0 (tra size=61
sspc_20140215 T -l 0 /tmp/__MTEMP_95485_70155283659220_4 2.0 /tmp/__MTEMP_95485_7015528365
9220_3
trsact: /tmp/__MTEMP_95485_70155283659220_4 ,#transactions 7 ,#items 7 ,size 27 extracted
database: #transactions 7 ,#items 7 ,size 27
 output to: /tmp/__MTEMP_95485_70155283659220_3
separated at 0
14 pairs are found
0,1,2,4, :1.000000  (0)
0,1,2,4,6, :1.000000  (0)
0,1,2,3,6, :1.000000  (0)
2,3,4, :1.000000  (0)
0,1,3,4,5, :1.000000  (0)
4,5, :1.000000  (0)
1,2,6, :1.000000  (0)
come
input-file /tmp/__MTEMP_95485_70155283659220_3, output-file /tmp/__MTEMP_95485_70155283659
220_4
degree threshold:
first & second scan end: /tmp/__MTEMP_95485_70155283659220_3
file read end: /tmp/__MTEMP_95485_70155283659220_3
iterative scan: #nodes=7, #edges = 28
forwardstar graph: /tmp/__MTEMP_95485_70155283659220_3 ,#nodes 7(7,7) ,#edges 28
#MSG# converting the numbered nodes into original name ...
#END# /usr/bin/mpolishing.rb ei=dat1.csv ef=node1,node2 sim=T th=2 iter=1 eo=result3.csv
$ more result3.csv
node1,node2
a,b
a,c
a,d
a,e
a,g
b,c
b,d
b,e
b,g
c,d
c,e
c,g
d,e
e,f
```

例 4: 直接の接続を考慮しない例

-indirect を指定することで、類似度の計算で直接の接続は無視される。出力結果は図 2.14 に示されるグラフであり、枝が全て消えるため、研磨後の枝データは出力されない。

```
$ mpolishing.rb ei=dat1.csv ef=node1,node2 sim=T th=2 eo=result4.csv -indirect
#MSG# converting graph files into a pair of numbered nodes ...
input-file /tmp/__MTEMP_95528_70362712963820_3, output-file /tmp/__MTEMP_95528_70362712963
820_4
degree threshold:
first & second scan end: /tmp/__MTEMP_95528_70362712963820_3
file read end: /tmp/__MTEMP_95528_70362712963820_3
iterative scan: #nodes=7, #edges = 20
forwardstar graph: /tmp/__MTEMP_95528_70362712963820_3 ,#nodes 7(7,7) ,#edges 20
#MSG# polishing iteration #0 (tra size=47
sspc_20140215 T -l 0 /tmp/__MTEMP_95528_70362712963820_4 2.0 /tmp/__MTEMP_95528_7036271296
3820_3
trsact: /tmp/__MTEMP_95528_70362712963820_4 ,#transactions 7 ,#items 7 ,size 20 extracted
database: #transactions 7 ,#items 7 ,size 20
 output to: /tmp/__MTEMP_95528_70362712963820_3
separated at 0
6 pairs are found
1,2,4, :1.000000  (0)
0,2,4,6, :1.000000  (0)
0,1,3,6, :1.000000  (0)
2,4, :1.000000  (0)
0,1,3,5, :1.000000  (0)
4, :1.000000  (0)
1,2, :1.000000  (0)
come
input-file /tmp/__MTEMP_95528_70362712963820_3, output-file /tmp/__MTEMP_95528_70362712963
820_4
degree threshold:
first & second scan end: /tmp/__MTEMP_95528_70362712963820_3
file read end: /tmp/__MTEMP_95528_70362712963820_3
iterative scan: #nodes=6, #edges = 12
forwardstar graph: /tmp/__MTEMP_95528_70362712963820_3 ,#nodes 7(7,7) ,#edges 12
#MSG# polishing iteration #1 (tra size=31
sspc_20140215 T -l 0 /tmp/__MTEMP_95528_70362712963820_4 2.0 /tmp/__MTEMP_95528_7036271296
3820_3
trsact: /tmp/__MTEMP_95528_70362712963820_4 ,#transactions 7 ,#items 7 ,size 12 extracted
database: #transactions 7 ,#items 7 ,size 12
 output to: /tmp/__MTEMP_95528_70362712963820_3
separated at 0
0 pairs are found
1,3,6, :1.000000  (0)
0,2,3, :1.000000  (0)
1,4, :1.000000  (0)
0,1, :1.000000  (0)
2, :1.000000  (0)
 :1.000000  (0)
0, :1.000000  (0)
come
input-file /tmp/__MTEMP_95528_70362712963820_3, output-file /tmp/__MTEMP_95528_70362712963
820_4
degree threshold:
first & second scan end: /tmp/__MTEMP_95528_70362712963820_3
file read end: /tmp/__MTEMP_95528_70362712963820_3
iterative scan: #nodes=0, #edges = 0
forwardstar graph: /tmp/__MTEMP_95528_70362712963820_3 ,#nodes 0(0,0) ,#edges 0
#MSG# polishing iteration #2 (tra size=6
sspc_20140215 T -l 0 /tmp/__MTEMP_95528_70362712963820_4 2.0 /tmp/__MTEMP_95528_7036271296
3820_3
```

```
trsact: /tmp/__MTEMP_95528_70362712963820_4 ,#transactions 2 ,#items 4 ,size 1 extracted d
atabase: #transactions 2 ,#items 4 ,size 1
 output to: /tmp/__MTEMP_95528_70362712963820_3
separated at 0
0 pairs are found
 :1.000000  (0)
3, :1.000000  (0)
come
input-file /tmp/__MTEMP_95528_70362712963820_3, output-file /tmp/__MTEMP_95528_70362712963
820_4
degree threshold:
first & second scan end: /tmp/__MTEMP_95528_70362712963820_3
file read end: /tmp/__MTEMP_95528_70362712963820_3
iterative scan: #nodes=0, #edges = 0
forwardstar graph: /tmp/__MTEMP_95528_70362712963820_3 ,#nodes 0(0,0) ,#edges 0
#MSG# converting the numbered nodes into original name ...
#END# /usr/bin/mpolishing.rb ei=dat1.csv ef=node1,node2 sim=T th=2 eo=result4.csv -indirec
t
$ more result4.csv
node1,node2
```

2.4　mtra2g.rb アイテム類似度グラフの構築

本コマンドは、トランザクションデータからアイテム間の類似構造を一般グラフ (以下「類似度グラフ」と呼ぶ) で出力する。内部では [3] の `lcm` コマンドを利用している。類似度は、2 アイテムの共起情報によって定義し、ユーザが指定した下限値より高い類似度を持つアイテム間に枝を張る。類似度としては、アイテムの出現確率 (support)、もしくは出現件数 (occurence) を指定する。また追加条件として resemblance、normalized PMI を用いることもできる。それぞれの定義は表 2.25 に示される通りである。

表 2.25　アイテム a と b の類似度の定義

類似度	定義式	パラメータ	範囲
support	$\frac{\lvert Occ(a,b)\rvert}{n}$	s=	0.0〜1.0
occurence	$\lvert Occ(a,b)\rvert$	S=	1〜
recemblance	$\frac{\lvert Occ(a)\cap Occ(b)\rvert}{\lvert Occ(a)\cup Occ(b)\rvert}$	sim=R th=	0.0〜1.0
normalized PMI	$\log \frac{P(a,b)}{P(a)P(b)} / (-\log P(a,b))$ $= \frac{n\lvert Occ(a)\cap Occ(b)\rvert}{\lvert Occ(a)\rvert\lvert Occ(b)\rvert} / (-\log \frac{\lvert Occ(a)\cap Occ(b)\rvert}{n})$	sim=P th=	-1.0〜1.0

n は全トランザクション数を表す。$Occ(a)$ はアイテム a が出現するトランザクション集合を表す。$P(a)$ はアイテム a の出現確率を表し、$P(a) = Occ(a)/n$ である。

本コマンドの入力データは、mitemset.rb コマンドと同様の key 型トランザクションファイルである (表 2.26)。表 2.27 に示されるような形式のデータは、MCMD パッケージの `mtra` コマンドによって key 型ファイルに変換すればよい。

このデータから類似条件として出現件数が 2 件以上とした場合に出力されるデータを表 2.28 に、そのグラフ構造を 2.20 に示す。

表 2.26　key 型データ

key	item
T1	C
T1	E
T2	D
T2	E
T2	F
:	:

表 2.27　tra 型データ

id	item
T1	C E
T2	D E F
T3	A B D F
T4	B D F
T5	A B D E
T6	A B D E F

表 2.28　出現件数が 2 件以上のアイテム類似度グラフ。出現件数は出現確率によって示されている。最後の項目 (void) は、sim=を指定した場合に、その値が出力される。

node1	node2	support	void
a	b	0.6	
a	d	0.4	
a	f	0.4	
d	b	0.6	
e	d	0.4	
f	b	0.6	
f	d	0.6	

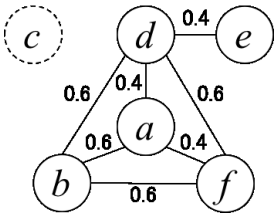

図 2.20　表 2.28 に対応する類似度グラフ。枝に示された数字は 2 つのアイテムの共起確率。

2.4.1 書式

書式）mtra2g.rb i= tid= item= [on=] eo= s= [sim=] [th=] [log=] [T=] [--help]

ファイル名指定

i= : **トランザクションデータファイル**

tid= : **トランザクション** ID **項目名**

item= : **アイテム項目名**

on= : **出力ファイル（節点）**

eo= : **出力ファイル（辺:節点ペア）**

s= : **最小支持度（全トランザクション数に対する割合による指定）:** 0 **以上** 1 **以下の実数】**

S= : **最小支持度（トランザクション数による指定）:** 1 **以上の整数】**

: s=,S=**共に指定しなければ、**S=1 **が指定されたとして動作する。**

sim= : **類似度**

R: resemblance

P: normalized PMI

省略時は s=**もしくは** S=**の条件によってのみアイテム間に枝が張られる。**

th= : sim=**で指定された類似度について、ここで指定された値以上のアイテム間に枝を張る。**

log= : **パラメータの設定値を** key-value **形式の** CSV **で保存するファイル名**

その他

T= : **ワークディレクトリ**（default:/tmp）

--help : **ヘルプの表示**

2.4.2 利用例

例 1: **基本例**

出現件数が 2 件以上の類似度グラフ。上述の解説の中で示した例。

```
$ more dat1.csv
tid,item
T1,C
T1,E
T2,D
T2,E
T2,F
T3,A
T3,B
T3,D
T3,F
T4,B
T4,D
T4,F
T5,A
T5,B
T5,D
T5,E
T6,A
T6,B
T6,D
T6,E
T6,F
```

```
$ mtra2g.rb  S=2 tid=tid item=item i=dat1.csv eo=edge1.csv
#MSG# converting a named item into a numbered item ...
#MSG# run lcm enumerating 2 itemset ...
#MSG# creating the edge file ...
#END# /usr/bin/mtra2g.rb S=2 tid=tid item=item i=dat1.csv eo=edge1.csv
$ more edge1.csv
node1,node2,support,void
A,B,0.5,
A,D,0.5,
A,E,0.3333333333,
A,F,0.3333333333,
B,D,0.6666666667,
E,B,0.3333333333,
E,D,0.5,
F,B,0.5,
F,D,0.6666666667,
F,E,0.3333333333,
```

例 2: resemblance **を追加**

例 1 に加えて resemblance が 0.4 以上を類似度条件に加える。また no=を指定することで、節点情報としてアイテム単独の出現頻度を出力する。

```
$ mtra2g.rb  S=2 sim=R th=0.4 tid=tid item=item i=dat1.csv eo=edge2.csv no=node2.csv
#MSG# converting a named item into a numbered item ...
#MSG# run lcm enumerating 2 itemset ...
#MSG# creating the edge file ...
#MSG# creating the node file ...
#END# /usr/bin/mtra2g.rb S=2 sim=R th=0.4 tid=tid item=item i=dat1.csv eo=edge2.csv no=nod
e2.csv
$ more node2.csv
node,support
A,0.5
B,0.6666666667
C,0.1666666667
D,0.8333333333
E,0.6666666667
F,0.6666666667
$ more edge2.csv
node1,node2,support,resemblance
A,B,0.5,0.75
A,D,0.5,0.6
A,E,0.3333333333,0.4
A,F,0.3333333333,0.4
B,D,0.6666666667,0.8
E,D,0.5,0.5
F,B,0.5,0.6
F,D,0.6666666667,0.8
```

2.5　mclique.rb 極大クリークの列挙

本コマンドは、一般グラフから極大クリークを列挙する。内部では [3] の mace コマンドを利用している。

$G = (V, E)$ を節点集合 V と枝集合 E を持つ無向グラフとすると、V の部分集合の任意の節点に枝があるような G の誘導部分グラフをクリークと呼ぶ。また、あるクリークが他のクリークの真部分集合でなければ、それは極大クリークと呼ばれる (図 2.21)。

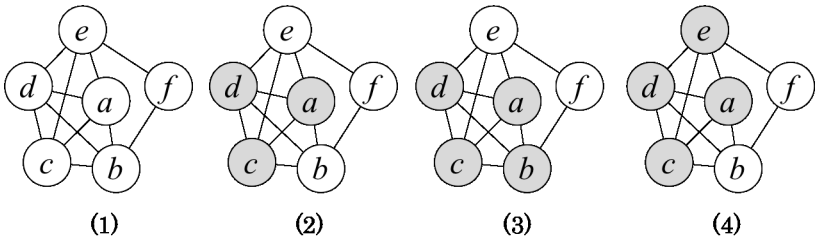

図 2.21　一般グラフ (1) について、網掛けで示された部分グラフ (2) はクリークではあるが、(3) や (4) に示されたクリークの真部分集合なので極大クリークではない。一方で (3) や (4) はその他のどのクリークの真部分集合にはならないので極大である。(3) と (4) は 4 つの頂点より構成されるが、そのうち 3 点が互いに共通している。

一般的なケースにおいて、極大クリークを列挙するとその数が膨大になることが多い。図 2.21 の (3) と (4) のように、節点が重複するような極大クリークが多数存在するためである。この問題を克服するアプローチがこれまでに幾つか提案されてきた。ひとつは、完全グラフでなくても、ある程度の密度であれば完全グラフと見なしてしまおうというもので、そのようなクリークは擬似クリークと呼ばれる。しかしながら、時に擬似クリークは極大クリーク以上に列挙されるケースもあり、根本的な解決にならない。他の方法としては、極大クリークを列挙した後で、似たクリークを併合していこうというアプローチであり、世に存在する多くのクラスタリングのアルゴリズムが利用できる。これは有望なアプローチではあるものの、列挙された極大クリークの数によっては実行時間が問題となる。そこで、第三のアプローチとして、列挙対象である一般グラフを事前にクリーニングする (このようなクリーニングを「データ研磨」と呼ぶ) ことで、そもそも列挙される極大クリークの数を大幅に減らしてしまおうという方法がある。この方法は宇野らによって最近提案された方法で [2]、データ研磨の数理的性質がより明らかになってくれば、クリーク列挙において有効な方法論となっていくであろう。第 2.3 節に示された mpolishing.rb コマンドは、この手法を実装したものであり、本コマンドと組み合わせて利用することで、グラフの本質をさほど変更することなく極大クリークの列挙数を劇的に少なくすることができる。

さて、本コマンドの入力データである一般グラフは、表 2.29 に示されるような、枝データを節点ペアとして表した CSV 形式で与える (図 2.21 (1) に対応)。グラフは無向グラフとして扱われ、島が複数あってもよい。

このグラフデータで極大クリークを列挙すると表 2.30 に示されるような形式でクリークが列挙される。id 項目がクリークの ID で、この項目が同じ行で一つのクリークが識別される。id=2 のクリークが図 2.21(4) に対応し、id=3 のクリークが図 2.21(3) に対応している。その他にも、2 つの節点から構成される極大クリーク $\{e, f\}$ と $\{b, f\}$ も列挙されている。最後の size 項目は、極大クリークを構成する節点数である。

2.5.1　書式

書式） `mclique.rb ei= ef= [ni=] [nf=] [o=] [l=] [u=] [-all] [-node] [-all] [-int] [log=] [T=] [--help]`

ファイル名指定

`ei=`　：**枝データファイル**

`ef=`　：**枝データ上の 2 つの節点項目名（`-int` を指定した場合は指定できない）**

`ni=`　：**節点データファイル**

`nf=`　：**節点データ上の節点項目名（省略時は"node"）**

表 2.29 入力データ (枝データ)

node1	node2
a	b
a	c
a	d
a	e
b	c
b	d
b	f
c	d
c	e
d	e
e	f

表 2.30 出力結果

id	node1	node2	size
0	e	f	2
1	b	f	2
2	a	c	4
2	a	d	4
2	a	e	4
2	c	d	4
2	c	e	4
2	d	e	4
3	a	b	4
3	a	c	4
3	a	d	4
3	b	c	4
3	b	d	4
3	c	d	4

`o=` : **出力ファイル（クリーク** `ID-枝:-node` **を指定することでクリーク** `ID-節点` **に変更可能）**

`l=` : **クリークを構成する最小節点数（ここで指定したサイズより小さいクリークは列挙されない）**

`u=` : **クリークを構成する最大節点数（ここで指定したサイズより大きいクリークは列挙されない）**

`-node` : **出力形式をクリーク** `ID-node` **名のペアとする。**

`-all` : **極大クリークだけでなく、全クリークを列挙する。**

`-int` : **アイテムを整数とみなして処理する。**

`log=` : **パラメータの設定値を** `key-value` **形式の** `CSV` **で保存するファイル名**

その他

`T=` : **ワークディレクトリ** (`default:/tmp`)

`--help` : **ヘルプの表示**

2.5.2 利用例

例 1: **基本例**

前節の解説で用いてる例。

```
$ more dat1.csv
node1,node2
a,b
a,c
a,d
a,e
b,c
b,d
b,f
c,d
c,e
d,e
e,f
$ mclique.rb ei=dat1.csv ef=node1,node2 o=result1.csv log=log1.csv
#MSG# converting graph files into a pair of numbered nodes ...
sgraph: /tmp/__MTEMP_95881_70169578268420_0 ,#nodes 6 ,#edges 11 ,#arcs 0
 output to: /tmp/__MTEMP_95881_70169578268420_2
iters=4
4
```

```
0
0
2
0
2
#MSG# converting the numbered nodes into original name ...
#END# /usr/bin/mclique.rb ei=dat1.csv ef=node1,node2 o=result1.csv log=log1.csv
$ more result1.csv
id,node,size
0,e,2
0,f,2
1,b,2
1,f,2
2,a,4
2,c,4
2,d,4
2,e,4
3,a,4
3,b,4
3,c,4
3,d,4
$ more cn1.csv
cn1.csv: No such file or directory
$ more ce1.csv
ce1.csv: No such file or directory
$ more log1.csv
key,value
ei=,dat1.csv
ef=,"node1,node2"
o=,result1.csv
log=,log1.csv
-all,false
time,0.085455
```

例 2: **サイズが** 4 **以上の極大クリークのみ列挙**

```
$ mclique.rb ei=dat1.csv ef=node1,node2 l=4 o=result2.csv
#MSG# converting graph files into a pair of numbered nodes ...
sgraph: /tmp/__MTEMP_95916_70352131156000_0 ,#nodes 6 ,#edges 11 ,#arcs 0
 output to: /tmp/__MTEMP_95916_70352131156000_2
iters=4
2
0
0
0
0
2
#MSG# converting the numbered nodes into original name ...
#END# /usr/bin/mclique.rb ei=dat1.csv ef=node1,node2 l=4 o=result2.csv
$ more result2.csv
id,node,size
0,a,4
0,c,4
0,d,4
0,e,4
1,a,4
1,b,4
1,c,4
1,d,4
```

例 3: **サイズが** 3 **の全クリークを列挙**

```
$ mclique.rb ei=dat1.csv ef=node1,node2 l=3 u=3 -all o=result3.csv
#MSG# converting graph files into a pair of numbered nodes ...
sgraph: /tmp/__MTEMP_95948_70136489403820_0 ,#nodes 6 ,#edges 11 ,#arcs 0
 output to: /tmp/__MTEMP_95948_70136489403820_2
iters=0
7
0
0
0
7
#MSG# converting the numbered nodes into original name ...
#END# /usr/bin/mclique.rb ei=dat1.csv ef=node1,node2 l=3 u=3 -all o=result3.csv
$ more result3.csv
id,node,size
0,a,3
0,b,3
0,c,3
1,a,3
1,b,3
1,d,3
2,a,3
2,c,3
2,d,3
3,b,3
3,c,3
3,d,3
4,a,3
4,c,3
4,e,3
5,a,3
5,d,3
5,e,3
6,c,3
6,d,3
6,e,3
```

2.6　mclique2g.rb 極大クリークグラフの生成

本コマンドは、一般グラフ (以下、「オリジナルグラフ」と呼ぶ) から列挙された各極大クリーク[*2]を節点とする新たなグラフ (「極大クリークグラフ」と呼ぶ) を生成する。クリーク間に共通節点があれば、対応する接点間に枝を張る。

以下に例を示す。図 2.22 のグラフは 4 つの極大クリークを含んでいるが、このグラフから図 2.23 に示すグラフが生成される。節点 0 は、図 2.21 のクリーク $\{a, b, c, d\}$ に、節点 1 は $\{d, e, f\}$ に、節点 2 は $\{e, f, g\}$ に、対応している。そして、節点 3 は $\{e, f, h\}$ に対応している。

極大クリークグラフの節点 0 と 1 は、オリジナルグラフにおいて共通節点を 1 つ (d) を持つので、節点 0 と 1 に枝が張られ、その重みが 1 となっている。同様に、1 と 2 は共通節点を 2 つ (e, f) を持つので、節点 1 と 2 に枝が張られ、その重みが 2 となっている。一方で、0 と 2 は共通節点を持たないので、節点 0 と 2 には枝が張られない。

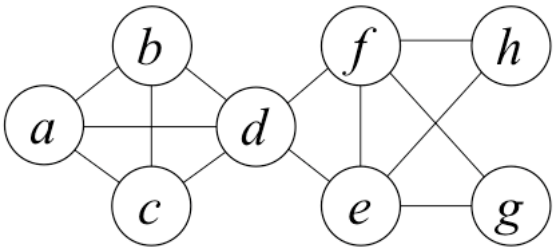

図 2.22　4 つの極大クリーク $\{a, b, c, d\}$ $\{d, e, f\}$ $\{e, f, g\}$ $\{e, f, h\}$ が含まれる。

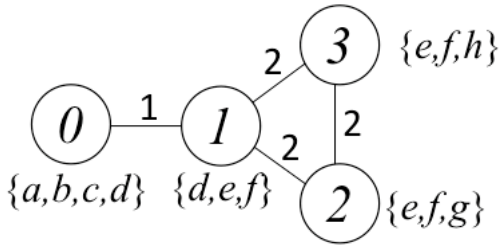

図 2.23　新たに生成される極大クリークグラフ

さて、本コマンドの入力データであるオリジナルグラフは、表 2.31 に示されるような、クリーク ID と節点の 2 項目から構成される CSV 形式データである。このデータは 2.5 節で示した mclique.rb コマンドで`-node` を指定して出力されたデータに対応している。

そして、出力結果である極大クリークグラフは、節点データ (表 2.32) と枝データ (表 2.33) を別々に出力する。節点データにおける `weight` 項目は、極大クリークを構成する節点数である。また枝データにおける `weight` 項目は、極大クリークを構成する節点数である。

2.6.1　書式

書式) `mclique2g.rb i= [id=] [f=] eo= no= [T=] [--help]`

ファイル名指定
`i=` ： **クリークファイル名**
`id=` ： **クリーク** `ID` **項目名 (デフォルト**:`"id"`)

[*2] 極大クリークの定義については 2.5 節を参照のこと。

表 2.31 オリジナルグラフ

id	node
0	a
0	b
0	c
0	d
1	d
1	e
1	f
2	e
2	f
2	g
3	e
3	f
3	h

表 2.32 極大クリークグラフ (節点データ)

node	weight
0	4
1	3
2	3
3	3

表 2.33 極大クリークグラフ (枝データ)

node1	node2	weight
0	1	1
1	2	2
1	3	2
2	3	2

`f=` : **クリークを構成する節点項目名**（**デフォルト**:`"node"`）

`eo=` : **出力枝ファイル名**

`no=` : **出力節点ファイル名**

その他

`T=` : **ワークディレクトリ**（`default:/tmp`）

`--help` : **ヘルプの表示**

2.6.2 利用例

例 1: **基本例**

前節で解説した例。

```
$ more clique.csv
id,node
0,a
0,b
0,c
0,d
1,d
1,e
1,f
2,e
2,f
2,g
3,e
3,f
3,h
$ mclique2g.rb i=clique.csv eo=edge.csv no=node.csv id=id f=node
#END# /usr/bin/mclique2g.rb i=clique.csv eo=edge.csv no=node.csv id=id f=node
$ more edge.csv
node1,node2,weight
0,1,1
1,2,2
1,3,2
2,3,2
$ more node.csv
node,weight
0,4
```

```
1,3
2,3
3,3
```

2.7　mbiclique.rb 極大二部クリークの列挙

本コマンドは、二部グラフから極大二部クリークを列挙する。内部では [3] の `lcm` コマンドを利用している。

$G = (V_1 \cup V_2, E)$ を 2 つの部 V_1, V_2 の節点集合と部間の枝に張られた枝集合 $E \subset V_1 \times V_2$ を持つ無向二部グラフとすると、V_1, V_2 の部分集合によって誘導される部分グラフの部間の任意の節点に枝があるような G の誘導部分グラフを二部クリークと呼ぶ。また、ある二部クリークが他の二部クリークの真部分集合でなければ、それは極大二部クリークと呼ばれる (図 2.24)。

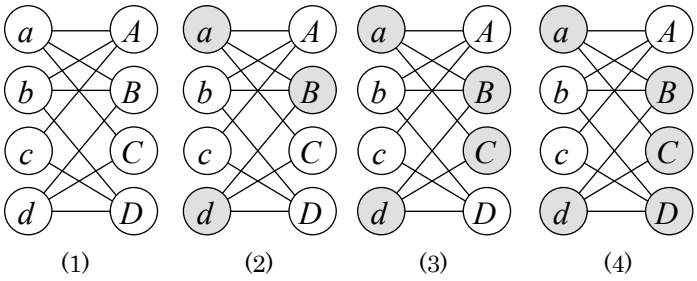

図 2.24　二部グラフ (1) について、網掛けで示された部分グラフ (2) は二部クリークではあるが、(3) に示された二部クリークの真部分集合なので極大二部クリークではない。一方で (3) はその他のどの二部クリークの真部分集合にはならないので極大である。(4) は a, D 間に枝がないので二部クリークではない。

極大二部クリークの列挙を使った応用例は多数考えられる。例えば、消費者の食品購入パネルデータにおいて、一つの部を商品集合、他方の部を店集合と見なし、ある閾値以上の購買のあった商品-店間に枝を張った二部グラフを構成できる。そこから極大二部クリークを列挙することで、互いに強い関係にある商品と店のグルーピングが可能となる。

またテキストマイニングにおいては、格助詞句と用言句を各部と見なし、用例のある格助詞句-用言句の間に枝を張った二部グラフを構成できる。そこから極大二部クリークを列挙することで、互いに強い関係にある格フレーム (格助詞句-用言句ペアのこと) のグルーピングが可能となり、ある種の概念を取得できる。

さらに、入札データにおいては、入札業者と案件をそれぞれ各部として考え、実際に入札があった入札業者-案件の間に枝を張ることで二部グラフが構成できよう。

さて、本コマンドの入力データである二部グラフは、表 2.34 に示されるような、枝データを節点ペアとして表した CSV 形式で与える (図 2.24 (1) に対応)。項目が部を表している。表 2.34 では、小文字のアルファベットを要素として持つ `node1` 項目が一方の部を表し、大文字のアルファベットを要素として持つ `node2` 項目が他方の部を表している。グラフは無向グラフとして扱われ、島が複数あってもよい。

この二部グラフから極大二部クリークを列挙すると表 2.35 に示されるような形式で出力される。一行が極大二部クリークに対応しており、各部を構成する要素が `node1,node2` 項目にベクトル形式で出力される。`size1,size2` には各部のサイズ (節点の数) が出力される。5 行目の極大二部クリーク ($V_1 = \{a, d\}, V_2 = \{B, C\}$) が図 2.24(3) に対応している。

2.7.1　書式

書式) `mbiclique.rb ei= ef= [o=] [l=] [u=] [T=] [-debug] [--help]`

ファイル名指定

`ei=` ： **辺データファイル**

`ef=` ： **辺データ上の 2 つの節点項目名**

`o=` ： **出力ファイル**

`l=` ： **極大二部クリークを構成する最小節点数**

表 2.34　入力データ (枝データ)

node1	node2
a	A
a	B
a	C
b	A
b	B
b	D
c	A
c	D
d	B
d	C
d	D

表 2.35　出力結果

node1	node2	size1	size2
a	A B C	1	3
a b	A B	2	2
a b c	A	3	1
a b d	B	3	1
a d	B C	2	2
b	A B D	1	3
b c	A D	2	2
b c d	D	3	1
b d	B D	2	2
d	B C D	1	3

: **ここで指定したサイズより小さい極大二部クリークは列挙されない。**
: **カンマで区切って 2 つの値を指定すると、各部のサイズを制限できる。**
: **カンマで区切った値の順番は** `ef=` **で指定した項目の順番と同じ。**
: **制限を設けない場合は**`"l=2,"`**や**`"l=,2"`**のように** `null` **文字を指定する。**
: **後ろの** `null` **は省略できる (**`"l=2,"`**と**`"l=2"`**は同じ意味)。**

`u=` : **極大二部クリークを構成する最大節点数**
: **指定の詳細は** `l=` **と同様である。**

その他

`T=` : **ワークディレクトリ (**`default:/tmp`**)**

`--help` : **ヘルプの表示**

2.7.2　注意点

本コマンドの二部クリークの出力形式がベクトル形式 (文字列を空白で区切った一つの CSV 項目) となるため、時に、一つの項目の長さが非常に長くなることがある。そのため、内部で使っている M コマンドの CSV 一行の最大長制約を超えてしまいエラーとなることがある。そのようなときは、`l=`もしくは `u=`によって二部クリークのサイズを制約することで回避する。

2.7.3　利用例

例 1: **基本例**

前節の解説で用いてる例。

```
$ more dat.csv
node1,node2
a,A
a,B
a,C
b,A
b,B
b,D
c,A
c,D
d,B
d,C
d,D
```

```
$ mbiclique.rb ei=dat.csv ef=node1,node2 o=result1.csv
#MSG# converting paired form into transaction form ...
#MSG# lcm_20140215 CIf /tmp/__MTEMP_95686_70245679198320_0 1 /tmp/__MTEMP_95686_7024567919
8320_3
trsact: /tmp/__MTEMP_95686_70245679198320_0 ,#transactions 4 ,#items 4 ,size 11 extracted
database: #transactions 4 ,#items 4 ,size 11
 output to: /tmp/__MTEMP_95686_70245679198320_3
separated at 0
iters=11
11
1
3
4
3
#END# /usr/bin/mbiclique.rb ei=dat.csv ef=node1,node2 o=result1.csv
$ more result1.csv
node1,node2,size1,size2
a,A B C,1,3
a b,A B,2,2
a b c,A,3,1
a b d,B,3,1
a d,B C,2,2
b,A B D,1,3
b c,A D,2,2
b c d,D,3,1
b d,B D,2,2
d,B C D,1,3
```

例 2: **サイズを制限する例**

項目 node1,node2 共にサイズが 2 の極大二部クリークを列挙する。

```
$ mbiclique.rb ei=dat.csv ef=node1,node2 o=result2.csv l=2,2 u=2,2
#MSG# converting paired form into transaction form ...
#MSG# lcm_20140215 CIf -l 2 -u 2 /tmp/__MTEMP_95739_70259512012180_0 1 /tmp/__MTEMP_95739_
70259512012180_3
trsact: /tmp/__MTEMP_95739_70259512012180_0 ,#transactions 4 ,#items 4 ,size 11 extracted
database: #transactions 4 ,#items 4 ,size 11
 output to: /tmp/__MTEMP_95739_70259512012180_3
separated at 0
iters=10
4
0
0
4
#END# /usr/bin/mbiclique.rb ei=dat.csv ef=node1,node2 o=result2.csv l=2,2 u=2,2
$ more result2.csv
node1,node2,size1,size2
a b,A B,2,2
a d,B C,2,2
b c,A D,2,2
b d,B D,2,2
```

例 3: **部分的にサイズを制限する例**

項目 node1 のサイズの下限を 1 に (デフォルトの下限が 1 なので実際には意味がないが指定例として)、項目 node2 のサイズの上限を 3 に制限した極大二部クリークを列挙する。u=パラメータの 1 番目の値が null になっているのは、項目 node1 の上限を設定しなためである。

```
$ mbiclique.rb ei=dat.csv ef=node1,node2 o=result3.csv l=1, u=,3
#MSG# converting paired form into transaction form ...
```

```
#MSG# lcm_20140215 CIf -u 3 /tmp/__MTEMP_95792_70093564374420_0 1 /tmp/__MTEMP_95792_70093
564374420_3
trsact: /tmp/__MTEMP_95792_70093564374420_0 ,#transactions 4 ,#items 4 ,size 11 extracted
database: #transactions 4 ,#items 4 ,size 11
 output to: /tmp/__MTEMP_95792_70093564374420_3
separated at 0
iters=11
11
1
3
4
3
#END# /usr/bin/mbiclique.rb ei=dat.csv ef=node1,node2 o=result3.csv l=1, u=,3
$ more result3.csv
node1,node2,size1,size2
a,A B C,1,3
a b,A B,2,2
a b c,A,3,1
a b d,B,3,1
a d,B C,2,2
b,A B D,1,3
b c,A D,2,2
b c d,D,3,1
b d,B D,2,2
d,B C D,1,3
```

2.8 mgdiff.rb グラフの差分

本コマンドは、2 つの一般グラフの差分情報を出力する。

2.8.1 書式

書式) mgdiff.rb ei= [ef=] eI= [eF=] [eo=] [no=] [T=] [--help]

ファイル名指定

ei= : **入力枝ファイル** 1

ef= : **枝を構成するの** 2 **つの節点項目名** (ei=**上の項目名**, **デフォルト**:node1,node2)

eI= : **入力枝ファイル** 2

eF= : eI=**上の** 2 **つの節点項目名** (ef=**と同じであれば省略できる**, **デフォルト**:ef=**で指定した項目名**)

ni= : **入力節点ファイル** 1(ei=**に対応**)

nf= : **枝を構成するの** 2 **つの節点項目名** (ni=**上の項目名**, **デフォルト**:node)

nI= : **入力節点ファイル** 2(eI=**に対応**)

nF= : nI=**上の** 2 **つの節点項目名** (nf=**と同じであれば省略できる**, **デフォルト**:nf=**で指定した項目名**)

eo= : **出力枝ファイル**

no= : **出力節点ファイル**

-dir : **有向グラフとして扱う**

その他

T= : **ワークディレクトリ** (default:/tmp)

--help : **ヘルプの表示**

注 1) **無向グラフとして扱う場合** (**デフォルト**)、ei=**ファイルと** eI=**ファイルとで、**
節点の並びが異なっていても、それは同じと見なす (ex. **枝** a,b **と枝** b,a **は同じ**)。
-dir **を指定すれば異なるものと見なす。**

注 2) **無向グラフとして扱う場合** (**デフォルト**)、
処理効率を重視し、ef=**で指定した節点の並びはアルファベット順に並べ替えるため、**
eo=**の項目の並びが** ei=**や** eI=**の並びと異なることがある。**

注 3) **同じ枝が複数ある場合、それらは単一化される。**

入力データ)

ei=,eI=: **節点ペアからなる** CSV **ファイル。**

ni=,nI=: **節点からなる** CSV **ファイル。**

枝出力データ)

枝ファイル 1 **と枝ファイル** 2 **のいずれかに出現する枝** (**節点ペア**) **について以下の値を出力する。**

項目名: **内容**

ei : ei=**で指定したグラフにその行の節点ペアがあれば、そのファイル名**

eI : eI=**で指定したグラフにその行の節点ペアがあれば、そのファイル名**

diff : **差分の区分**
- 1: ei=**のグラフにしか存在しない**
- 0: ei=,eI=**の両方に存在する**
- -1: eI=**のグラフにしか存在しない**

節点出力データ）

節点ファイル 1 **と節点ファイル** 2 **のいずれかに出現する節点について以下の値を出力する。**

項目名: **内容**

ni : ni=**で指定したグラフにその節点があれば、そのファイル名**

nI : nI=**で指定したグラフにその節点があれば、そのファイル名**

diff : **差分の区分**
- 1: ni=**のグラフにしか存在しない**
- 0: ni=,nI=**の両方に存在する**
- -1: nI=**のグラフにしか存在しない**

2.8.2 利用例

例 1: **基本例**

```
$ more g1.csv
node1,node2
a,b
b,c
c,d
$ more g2.csv
node1,node2
b,a
c,d
d,e
$ mgdiff.rb ei=g1.csv eI=g2.csv eo=result1.csv ef=node1,node2
#END# /usr/bin/mgdiff.rb ei=g1.csv eI=g2.csv eo=result1.csv ef=node1,node2
$ more result1.csv
node1,node2,ei,eI,diff
a,b,g1.csv,g2.csv,0
b,c,g1.csv,,1
c,d,g1.csv,g2.csv,0
d,e,,g2.csv,-1
```

例 2: **有向グラフとしての比較**

```
$ mgdiff.rb ei=g1.csv eI=g2.csv eo=result2.csv ef=node1,node2 -dir
#END# /usr/bin/mgdiff.rb ei=g1.csv eI=g2.csv eo=result2.csv ef=node1,node2 -dir
$ more result2.csv
node1,node2,ei,eI,diff
a,b,g1.csv,,1
b,a,,g2.csv,-1
b,c,g1.csv,,1
c,d,g1.csv,g2.csv,0
d,e,,g2.csv,-1
```

2.9　mcliqueInfo.rb 列挙されたクリークの各種情報

mclique.rb コマンドで列挙されたクリークについての各種情報を出力する。本コマンドの主な目的は、クリーク同士の接続関係についての各種情報を出力することにある。

mclique2g.rb コマンドの解説で用いたグラフを図 2.25 に再掲する。このグラフから得られる極大クリークは 4 つ存在する。これら 4 つのクリークそれぞれについて、表 2.36 に示す 5 つの情報を出力する。

表 2.36　本コマンドの出力結果

出力項目名	内容	id=1($\{d, e, f\}$ の例)
nSize	節点数	3 (節点 d, e, f の 3 つ)
eSize	枝数 (nSize∗(nSize-1)/2)	3 ($3 * 2/2 = 3$)
extNodes	外部接続節点数	5 (図 2.27 右)
extEdges	外部接続枝数	7 (図 2.27 左)
extCliques	外部接続クリーク数	3 (図 2.26)

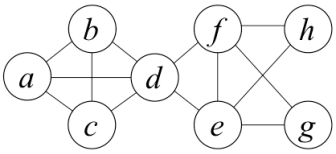

図 2.25　極大クリーク列挙の対象となるグラフ。4 つの極大クリーク {a,b,c,d}(id=0)、{d,e,f}(id=1)、{e,f,g}(id=2)、{e,f,h}(id=3) を含んでいる。

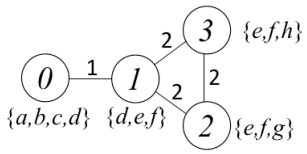

図 2.26　極大クリークグラフ。図 2.25 のクリーク id を節点としたグラフ。枝は共通の節点数を表している。id=1 のクリークは 3 つのクリークに直接の接続があることがわかる。なお、このグラフを出力したければ 2.6 節を参照のこと。

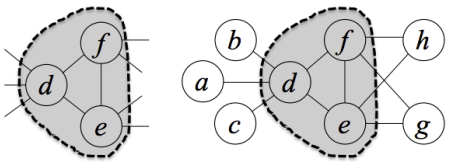

図 2.27　クリーク id=1(グレーで示された領域) から外の節点へと接続される枝は 7 本ある (左図)。クリーク id=1(グレーで示された領域) から接続される節点数は 5 である (右図)。

本コマンドの入力データは、2.6 節に示した mclique2g.rb コマンドと同様で、表 2.31 に示されるような、クリーク ID と節点の 2 項目から構成される CSV 形式データである。そして、出力結果は、表 2.38 に示されるように、極大クリーク毎に表 2.36 に示される情報が出力される。

2.9.1　書式

書式) `mcliqueInfo.rb i= [id=] [f=] [m=] [F=] o= [T=] [--help]`

`i=`　　: **クリークデータファイル名**

`id=`　　: **クリーク** `ID` **項目名 (デフォルト**:`"id"`**)**

`f=`　　: **クリークを構成する節点項目名 (デフォルト**:`"node"`**)**

表 2.37　入力データ

id	node
0	a
0	b
0	c
0	d
1	d
1	e
1	f
2	e
2	f
2	g
3	e
3	f
3	h

表 2.38　出力結果

id	nSize	eSize	extNodes	extEdges	extCliques
0	4	6	2	2	1
1	3	3	5	7	3
2	3	3	2	4	2
3	3	3	2	4	2

`T=` : **ワークディレクトリ** (`default:/tmp`)

`--help` : **ヘルプの表示**

2.9.2　利用例

例 1: **基本例**

前節で解説した例。

```
$ more clique.csv
id,node
0,a
0,b
0,c
0,d
1,d
1,e
1,f
2,e
2,f
2,g
3,e
3,f
3,h
$ mcliqueInfo.rb i=clique.csv o=result1.csv id=id f=node
#END# /usr/bin/mcliqueInfo.rb i=clique.csv o=result1.csv id=id f=node
$ more result1.csv
id,nSize,eSize,extNodes,extEdges,extCliques
0,4,6,2,2,1
1,3,3,5,7,3
2,3,3,2,4,2
3,3,3,2,4,2
```

参考文献

[1] Takeaki Uno, Tatsuya Asai, Yuzo Uchida, Hiroki Arimura, ”An Efficient Algorithm for Enumerating Closed Patterns in Transaction Databases”, *Discovery Science 2004*, LNAI 3245, pp.16-31.

[2] 宇野毅明, 中原孝信, 前川浩基, 羽室行信「データ研磨によるクリーク列挙クラスタリング」情報処理学会アルゴリズム研究会報告書, 2014-AL-146(2), pp. 1-8, 2014.

[3] http://research.nii.ac.jp/ uno/codes-j.htm