ビッグデータ解析ツールNYSOL
—性能評価，並列処理，ビジネス応用ケース—

中原　孝信，中元　政一，羽室　行信

NYSOL（にそる）はデータ分析のためのオープンソースソフトウェアで，2003年にリリースされたMUSASHIの後継となる．NYSOLの特徴は，大規模なCSVデータに対して単一の処理に特化したコマンド群を組み合わせることで，データの加工・前処理から，マイニングアルゴリズムの適用まで，KDD (Knowledge Discovery in Databases) の全プロセスを効率よく実現できる．そして1億件以上の大規模データをPCで処理することが可能である．

本稿では，NYSOLの利用方法を簡単に紹介し，より効率的な処理を行うために実現された並列処理に関する利用方法と計算実験の結果を紹介する．そして次にNYSOLの応用研究としてマイクロクラスタリングの利用方法について説明し，NYSOLが実際のビジネスにどのように利用されているのかを紹介する．

キーワード：スーパーコンピュータ，ビッグデータ，並列処理，データ解析，データマイニング

1. はじめに

スマートフォンやSNSの普及，レシートクーポン，センサー技術の向上など情報通信技術の急速な発展により，Webサイトの閲覧履歴，小売店の購買履歴，ユーザの位置情報，移動経路など，さまざまなデータが蓄積されており，ビッグデータブームとともに企業でもより一層データの収集とその利用に注目が集まっている．

NYSOL[1]は大規模なCSVデータを効率よく処理するために開発されたデータ分析プラットフォームであり，単一の処理に特化したコマンド群を組み合わせることで，効率的に柔軟な処理が行える．一つひとつのコマンドは高速に動作し，1億件以上の大規模データをPCで処理することができる．そして，前処理からマイニング手法，可視化まで知識発見プロセス (KDD) のすべてを扱えることが特徴である．また，シェルやRubyスクリプトと組み合わせて利用することで，コマンドを一つの関数のように扱うことができたり，出力結果のCSVファイルをR，EXCEL，WEKAなど他の統計パッケージや解析ツールの入力として与えたりすることで，他のプログラミング言語やパッケージとの連携ができる．NYSOLにはMCMD（「Mコマンド」と呼ぶ），Mining，Take，ZDD，View，そしてFumiといった六つのパッケージが提供されており，各パッケージではマイニングアルゴリズム，パターン列挙，ZDD，可視化，そしてテキストマイニングなどのコマンドが利用できる．

なかはら　たかのぶ
専修大学商学部
〒214–8580 神奈川県川崎市多摩区東三田2–1–1
nakapara@isc.senshu-u.ac.jp
なかもと　まさかず，はむろ　ゆきのぶ
関西学院大学経営戦略研究科
〒662–8501 兵庫県西宮市上ヶ原一番町1–155
nain@kwansei.ac.jp
hamuro@kwansei.ac.jp

2. Mコマンド：データ前処理ツール

本節では，データの前処理に威力を発揮するMコマンドについて紹介する．Mコマンドは C++ によって実装された80種類以上のコマンドから構成されている．表1に，それらの一部を抜粋してその機能を示している．たとえばmcutはデータから指定した列を抜き出すコマンドで，msumは指定したレコードを合計するコマンドである．個々のコマンドは単一の処理機能に特化しており，これらのコマンドをUNIX OSの機能であるパイプを利用して組み合わせることで柔軟な処理を可能にする．パイプとは，FIFO (First In First Out) キューを用いたプロセス間通信の機構のことで，パイプで接続されたプロセスは並列処理され，MISD (Multiple Instruction Single Data) 型の並列処理を容易に実現できる．このようにOSのコマンドとして実装することで，OSに元々備わっている機能を容易に利用できる．これは他の多くのデータ解析パッケージには見られないNYSOLの大きな特徴である．

図1のスクリプトは顧客ごとの平均来店間隔を計算するスクリプトを示している．一つ目のコマンド`mcut`で図2の入力データから分析に必要な列の項目を選択

[1] http://www.nysol.jp

表 1　M コマンド群の例

コマンド名	機能
mcut	項目の抜き出し
msum	レコードの合計
msortf	列の並べ替え
mcat	ファイルの併合
mjoin	単純結合
mrjoin	範囲結合
msel	行の条件選択
mcal	項目間の計算
muniq	重複行の単一化
mcount	行数のカウント
msetstr	文字列項目追加
mnumer	連番の作成
msed	正規表現による文字列置換
mtra	縦型データをベクトル項目に変換
mslide	指定した行数ずらす
msim	さまざまな類似度の計算

```
1 mcut f=顧客,日付 i=input.csv          |
2 muniq k=顧客,日付                     |
3 mslide k=顧客 f=日付:次回日付 s=日付   |tee tmp |
4 mcal c='$d{次回日付}-$d{日付}' a=間隔  |
5 mavg k=顧客 f=間隔:平均来店間隔        |
6 mcut f=顧客,平均来店間隔 o=result.csv
```

図 1　平均来店間隔を計算するスクリプト

している．二つ目のコマンドで顧客と日付をキーにして重複する行を単一にする．つまり入力データの 1, 2, 3 行目が単一行になり，5, 6 行目も単一行になる．三つ目のコマンド `mslide` で顧客ごとに日付を 1 行ずらすことで次回日付を作成している．ここまでの出力を UNIX コマンド `tee` でファイル `tmp`（図 3）に出力しており，四つ目のコマンド `mcal` で次回日付と日付の差を日数として計算している．そして五つ目のコマンドで顧客ごとに平均を計算し，六つ目のコマンドで顧客と平均来店間隔だけを抜き出し result.csv（図 4）に出力している．result.csv の項目名「顧客」に付加された「%0」の記号は，データの並べ替え条件を表しており，M コマンドが処理の効率化のために内部的に利用している．以上のように，NYSOL のアーキテクチャは，単一機能に特化したコマンド群をパイプを用いて組み合わせることによって多様な処理を実現するというものであり，この点はまさに UNIX の伝統的な考え方に基づくものである [1].

3. ベンチマークテスト

本節では NYSOL の処理性能を示すために，MCMD と同様の機能を有する他のソフトとのベンチマークテストの結果を紹介する．ベンチマークとして用いたのは表 2 に示した 4 種類のソフトウェアである．

処理対象データとしては，図 5 に例示されるように，一つの集計キー項目 (`key`)，二つの整数値項目 (`fld1,fld2`)，そして二つの実数値項目 (`fld3,fld4`) をもつ CSV ファイルを用意した．集計キーは 1 万種類の値をランダムに生成している．また，スケーラビリティを確かめるために，異なる三つのサイズのデー

```
顧客,店,日付,数量,金額,商品名
10001,101,20150102,2,400,パン
10001,101,20150102,1,200,カレー
10001,101,20150102,1,110,おにぎり
10001,101,20150410,1,120,牛乳
10002,101,20150102,2,300,お茶
10002,101,20150102,3,300,カップ麺
10002,101,20150115,1,105,おにぎり
10002,101,20150125,1,130,納豆
```

図 2　CSV の入力データ例

```
顧客,日付,次回日付
10001,20150102,20150410
10002,20150102,20150115
10002,20150115,20150125
```

図 3　途中の結果：`tmp`

```
顧客%0,平均来店間隔
10001,98
10002,11.5
```

図 4　M コマンドの出力結果

```
key,fld1,fld2,fld3,fld4
ID4171,187,210,2.102925087,0.2219931714
ID9972,137,114,-0.4419775337,0.05518012075
ID7204,105,241,-0.5888460571,0.8707323035
ID9326,286,267,-0.1800688891,0.83132784
ID2,209,158,1.938485447,0.206719154
```

図 5　ベンチマークテスト用のデータ例

表 2　ベンチマークに利用したソフトウェア

名称	内容	version
mcmd	NYSOL のデータ処理パッケージ MCMD	NYSOL 2.4
gsort	unix 標準のソーティングコマンド	GNU coreutils 5.93
dplyr	統計解析パッケージ R で利用可能な表構造データ処理ライブラリ	R: 3.2.2, dplyr: 0.4.3, data.table: 1.9.6
pandas	Python で利用可能な表構造データ処理ライブラリ	python: 2.7.9, pandas: 0.1.4
mysql	リレーショナル・データベース	Ver 14.14 Distrib 5.6.27

表 3　ベンチマークテストの結果（経過秒数）

処理	ソフト	1M	10M	100M
sel	mcmd	**0.12**	**1.03**	**10.20**
	dplyr	0.58	4.89	48.48
	pandas	0.85	8.48	85.16
	mysql	0.40	4.82	47.33
sum	mcmd	**0.46**	**4.22**	**42.00**
	dplyr	0.66	6.20	62.49
	pandas	0.92	9.34	95.17
	mysql	2.34	23.19	230.07
sort	mcmd	**0.56**	8.17	110.03
	gsort	20.02	242.18	—
	dplyr	0.74	**5.94**	**60.72**
	pandas	1.88	23.50	278.32
	mysql	4.05	40.21	688.79

実験で利用した PC: MacPro (2013)
CPU: 2.7 GHz 12-Core Intel Xeon E5
メモリ：64 GB 1866 MHz DDR3 ECC
ストレージ：SSD

タ，1M（100 万行 41MB），10M（1 千万行 410MB），100M（1 億行 4.1 GB）を用意した．

処理内容としては，集計キー項目 (`key`) から値 “`ID1`” を検索する処理 (sel)，集計キーで `fld1` から `fld4` を合計する処理 (sum)，そして集計キーを並べ替える処理 (sort) の 3 種類について実験した．実験では，各処理を 6 回実行し，最も悪い結果を省いたうえでの平均経過秒数を計算した．実験結果を表 3 に示す．各処理，各データサイズにおいて最も成績のよい結果が強調表示されている．

sel と sum の処理においては mcmd が最も高速で，sort においては dplyr が高速であった．また，sorting 以外の処理は，概ね線形のスケーラビリティがあることが確認できる．各コマンドの結果は 1,000 秒までで打ち切っており，gsort の 100M は打ち切られた．

ただし，ベンチマークテストでは，各ソフトウェアの実行条件をできる限り揃えたうえで実施したが，それぞれのソフトウェアは異なる利用目的の下で開発されたものであり，完全にその条件を揃えることは難しい．たとえば，R 上で動作する dplyr は処理結果はメモリ内に格納された状態で終了するが，一方で，その他のソフトウェアは処理結果をファイル出力しており，それらの条件を完全に揃えることは困難である．よって，上記のベンチマークテストは一つの参考情報として見ていただきたい．

いずれにせよ，データサイズが数億件（数十 GB）となると，単一の PC での処理には限界が出始めてくるため，並列処理の導入が欠かせなくなってくる．

表 4　並列処理の三つの方法

関数名	特徴
1) `meach`	1 台のマシンで fork と exec を利用することで複数のプロセスを並列実行する．
2) `meachi`	コンピュータクラスタを対象に MPI により並列実行する．
3) `meachc`	MPI が利用できない一般的なコンピュータネットワークを対象とした並列処理を実現している．
each	1 台のマシンでシングルプロセスによる逐次処理を行う．ベンチマークとして利用．

4. 並列処理

近年扱うべきデータ量は格段に増加しており，平成 27 年度情報通信白書 [2] によると国内データ通信量は 2014 年で 14.5 エクサバイトとなり 9 年間で 9 倍以上になっている．またデータ分析においてもユーザ同士の類似性を計算する際などでは，数 10 GB〜数 100 GB のデータを繰り返し処理する必要性も生じている．しかしながら，1 台のマシンの性能を向上させることは容易ではなく，単一のマシンによる計算速度の向上は難しい．一方で Amazon AWS などクラウドコンピューティングの利用環境は整備され，より身近なものとなってきており，複数のマシンを同時に実行することで，処理をスケールアウトさせ計算時間を短縮することは必要不可欠な技術である．

NYSOL では，前述のパイプによる MISD 型の並列処理以外にも，SIMD (Single Instruction Multipule Data) 型の並列処理の仕組みも提供しており，Ruby 拡張ライブラリによって三つの異なる関数を提供している．その概要を表 4 に示す．いずれの方法においても，入力データをあらかじめ分割しておき，それぞれのファイルに対して同一の処理を複数のプロセス上で並列実行するという非常に単純なものである．複数のファイル名は Ruby の配列に格納し，`meach` などのメソッドが，それぞれのファイル名を異なるプロセスに割り当てる．三つの関数は，このプロセスへの割り当て方法の違いであり，`meach` は単一の PC において Fork/Exec システムコールを発行することで実現し，`meachi` は MPI (Message Passing Instruction) ライブラリを用いることで，コンピュータ・クラスタ上の抽象化されたプロセスに割り当てられる．`meachc` は少し複雑で，あらかじめユーザが登録したネットワーク上の複数の PC に対して，データとプログラムの転送を独自に行い，さらには終了判定やエラー処理などの JOB 管理までも行っている．このことにより，MPI が利用でき

```
# CSV file
# v1,v2,v3,v4,v5,v6,...
# 257,258,257,256,256,259,...
# 322,317,321,321,325,321,...
# 130,129,130,129,130,129,...
# 127,130,130,129,130,126,...
#         :

# 別の実験では meach を meachc, meachi に置き換える
# だけでよい.
files.meach{|file| # files.size=500
  msim f=v1,v2,v3,v4 i=#{file} o=/dev/null
}
```

図 6　実験データと Ruby スクリプト

ない環境であってもコンピュータ・クラスタと同等の並列処理を実現可能としている.

さらに，これらの並列処理とパイプによる並列処理を組み合わせることで，MIMD (Multiple Instruction Multiple Data) 型の並列処理も容易に実現可能である.

以下では，表 4 に示した 3 種類の並列処理の比較実験の結果を紹介する．実験環境としては，統計数理研究所が保有するスーパーコンピュータ AIC を用いた. AIC は異なる三つのタイプのシステムで構成されており，`meachi` は，そのうちの一つであるコンピュータクラスタ HP XC4000 上で，また `meach` と `meachc` は共有クラウドシステム上で行った．前者は，高速ネットワーク InfiniBand で接続された 5 次元ハイパーキューブトポロジーで構成されており，データ転送が非常に高速である．また後者のクラウドは一般的なネットワーク構成でありデータ転送は比較的低速である.

実験で利用した入力ファイルの例と Ruby スクリプトを図 6 に示す．処理対象データは，CSV ファイルとして v1 から v100 までの 100 変数と 3,500 サンプルの行列データを 500 個用意し，そのファイル名を Ruby 配列に格納し計算を行った．並列処理は Ruby スクリプトで用いられる each 関数を `meach`，`meachc`，`meachi` に置き換えるだけで格納した Ruby 配列から順次ファイルが呼び出されて並列処理が実行できる．評価は，相関係数を計算する msim コマンドを逐次処理 (each) で実行した場合の処理時間と比較することで行った.

SIMD 型の並列処理においては，ある台数までは計算を担う CPU コア数が多いほどその効果は高くなるが，特にビッグデータの処理においては，複数のコアへのデータ転送がオーバヘッドとなる．処理にかかる時間が，データ転送にかかる時間に比べて大きくなると，並列処理の効率性は高まる．またデータ転送といっても，HDD やメモリのアクセス速度，I/O バス転送速度，PC 間のネットワーク速度など多様な経路を経由するため，そのシステム構成に大きく影響を受ける.

表 5　meachc の計算時間

v	$n=2$ $c=4$	$n=4$ $c=4$	$n=6$ $c=4$	$n=8$ $c=4$	meach $c=4$	each $c=1$
2	213.6	81.0	56.4	47.7	3.7	8.8
10	253.9	93.5	62.8	51.9	58.2	179.3
20	398.6	139.7	89.9	68.9	237.0	737.8

Machine spec.: Intel Xeon 2.8 Ghz 4 cores, 32 GB memory

このように，ビッグデータ処理のベンチマークを行うにあたっては，CPU のコア数に対する効率性評価以外にも，処理負荷とデータ転送量のバランスを考慮した評価も必要となる．そこで実験では，ノード数 n と CPU コア数 c を変化させることで CPU 総コア数 $p=n\times c$ の効果を測り，また，相関係数を計算する変数の数 v を変更することで，データ転送量に対する相対的な処理負荷を変化させ，その効果を測定した．例えば，$v=10$ の場合，45 ペア $({}_{10}C_2)$ の相関係数を計算することになる.

評価指標は並列プログラミングの性能評価で用いられる台数効果と並列効率化によって評価を行った [3]. プロセッサを p 台利用したときの台数効果 S_p は式 (1) で定義される.

$$S_p = T/T_p \tag{1}$$

ここで，T_p は p 台のプロセッサを利用した場合の処理時間を，そして T は逐次処理の時間を表しており，台数効果 S_p は，並列化によって，逐次と比較して何倍高速化ができているかを表す．そして，p 個のプロセッサを用いて $S_p=p$ になる場合は「理想的な速度向上」と呼ばれる．S_p は p の数に依存して決まるため，異なる p をもつシステム間での比較ができない．そこで，S_p を p で除することでプロセッサ当たりの台数効果を表したのが並列化効率 E_p である（式 (2)）．たとえば 100 個のプロセッサを使い，50 倍の台数効果が得られると並列化効率は 50%となる.

$$E_p = S_p/p * 100 \tag{2}$$

表 5 は `meachc` の計算時間，表 6 は台数効果および並列化効率を示している．$v=20$ の計算負荷に対して，$p=32(n=8,c=4)$ において台数効果は 10.7 倍（並列化効率で 33.45%），すなわち 32 台のプロセッサを使っても処理時間は 10 倍程度しか改善していない．これは，低速なネットワークを経由したデータ転送のオーバヘッドが原因と考えられる．一方で，`meach` で

表 6 meachc の台数効果と並列化効率．上段が台数効果で下段が並列化効率を表す

v	$n=2$ $c=4$	$n=4$ $c=4$	$n=6$ $c=4$	$n=8$ $c=4$	meach $c=4$
2	0.04	0.11	0.16	0.18	2.34
10	0.71	1.92	2.85	3.46	3.08
20	1.85	5.28	8.20	10.70	3.11
2	0.52%	0.68%	0.65%	0.58%	58.55%
10	8.83%	11.99%	11.89%	10.80%	76.93%
20	23.14%	32.99%	34.16%	33.45%	77.82%

表 7 meachi の計算時間

v	$n=2$ $c=24$	$n=4$ $c=24$	$n=6$ $c=24$	$n=8$ $c=24$	each $c=1$
2	0.77	0.78	0.73	1.37	14.25
10	2.31	1.41	1.40	1.32	91.91
20	7.94	4.59	3.49	2.88	347.12

Machine spec.: Intel Xeon 2.7Ghz 12 cores x 2, 128GB memory

表 8 meachi の並列化効率．上段が台数効果で下段が並列化効率を表す

v	$n=2$ $c=24$	$n=4$ $c=24$	$n=6$ $c=24$	$n=8$ $c=24$
2	10.37	10.24	10.84	5.82
10	37.28	61.17	61.58	65.29
20	42.99	74.26	97.64	118.38
2	21.60%	10.67%	7.53%	3.03%
10	77.68%	63.72%	42.76%	34.00%
20	89.56%	77.36%	67.81%	61.66%

は，$p=4(n=1,c=4)$ において台数効果は 3.11 で，4 台のプロセッサで約 3 倍の時間短縮を達成している．これは，データ転送は 1 台のノードに限定されるため HDD からの転送のみとなり，`meachc` で発生するネットワーク転送に比べてデータ転送のオーバヘッドが低く押さえられているためである．

次に `meachi` の実験結果を表 7 および表 8 に示している．$v=20$ の計算負荷に対して，$p=192$ $(n=8, c=24)$ において台数効果は約 120 倍，並列化効率は 61.66% であった．`meachc` に比べて台数効果が大幅に増加しているのは総コア数の大幅な増加に伴うものである．また，並列化効率も大幅に高まっているが，これは，コンピュータ・クラスタが有する高速なネットワークによりデータ転送のオーバヘッドが低下したことがその理由であろう．

また，すべての並列処理に共通した特徴として，処理負荷 (v) が増えるほど，データ転送のオーバヘッドが相対的に低くなり，並列化効率も高まることが確認された．

5. NYSOL を利用した応用事例

これまで著者らは多くのビジネスデータの解析に取り組んできており，特定のマーケティング課題やビジネス課題を設定し，その課題解決のために必要なデータ解析手法，分析フレームワーク，そしてモデル化の方法など多くを提案してきた．また，コンピュータサイエンスの研究者がもつ技術やアルゴリズムを NYSOL パッケージに組み込み[2]，研究成果を広く産業界に還元しようという試みを NYSOL プロジェクトとして行ってきた．したがって，NYSOL パッケージにはアカデミックにおけるこれまでのデータ解析のノウハウが詰まっており，エンドユーザーとして使いやすい分析ツールになるように改善が続けられている．ここでは NYSOL を利用した応用事例として，顧客の店舗選択モデル [4] を紹介し，そこで用いた Take パッケージの利用方法を示す．

顧客が店を選択する際に考慮する店舗選択要因の中でも商品に着目し，各商品と店舗選択との関係性を明らかにすることが分析の目的である．具体的には，消費者の複数店舗における購買状況を把握できるスキャンパネルデータを対象に，商品を購買した店舗とその商品のペアを一つのアイテムとして扱う．そして，アイテムの同時購買関係を利用し，ある閾値以上の共起関係の強さがある場合に，アイテム間に枝を張ることで一般グラフを作成する．そして，一般グラフからクリーク（完全部分グラフ）を抽出することでクラスタとして利用する．このクラスタを顧客の店舗選択の要因として利用し回帰モデルを構築する．以上の方法によって，ある商品を購入する際に顧客が想起する店舗が抽出可能であり，たとえば，「日用品を買うならダイエー」というような特徴が得られる．

一般グラフからクリークを列挙する際に，現実データにおいては多くの場合，類似したクリークが多数列挙されてしまう問題があり，この研究の特色は，グラフのクリーニング方法である「グラフ研磨」手法 [5] を適用することで，列挙されるクリークを大幅に削減できるという特徴がある．グラフ研磨を適用したグラフから列挙されたクリークを特にマイクロクラスタと呼ぶ．グラフ研磨の方法はシンプルで，すべての節点ペ

[2] ZDD パッケージ（北海道大学 湊真一氏が代表の ERATO で開発されたアルゴリズム），Take パッケージ（国立情報学研究所 宇野毅明氏が開発されたアルゴリズム）などがある．

```
1 mtra2g.rb i=tra.csv tid=顧客
2  item=商品名 sim=R th=0.4 eo=graph.csv
3
4 mpolishing.rb ei=graph.csv ef=node1,node2
5  sim=R th=0.2 eo=polished.csv
6
7 mclique.rb ei=polished.csv
8 ef=node1,node2 eo=microcluster.csv
```

図 7　マイクロクラスタリングの実行スクリプト

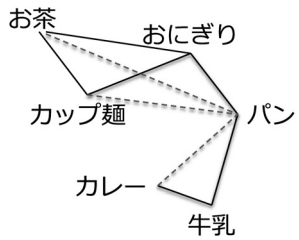

図 8　研磨後のグラフ

表 9　tra.csv

顧客, 商品
1, パン
1, カレー
1, おにぎり
1, 牛乳
2, お茶
2, カップ麺
2, おにぎり
3, お茶
3, パン
3, おにぎり
4, 牛乳
4, カップ麺
4, パン
5, お茶
5, カップ麺
5, おにぎり

表 10　graph.csv

node1,node2,resemblance
お茶, おにぎり, 0.75
カップ麺, おにぎり, 0.4
カップ麺, お茶, 0.5
カレー, 牛乳, 0.5
パン, おにぎり, 0.4
牛乳, パン, 0.667

表 11　研磨後のグラフ

node1,node2
おにぎり, お茶
おにぎり, カップ麺
おにぎり, パン
お茶, カップ麺
お茶, パン
カップ麺, パン
カレー, パン
カレー, 牛乳
パン, 牛乳

アについて，隣接関係に基づいて定義された類似度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．

図 7 はマイクロクラスタを得るための実行方法を示している．マイクロクラスタを得るためのアルゴリズムは NYSOL の Take パッケージでのみ利用可能な方法で，mpolishing.rb は学術の先端研究の成果を取り入れたコマンドである。商品の購買関係からグラフを生成するために Take パッケージでは，mtra2g.rb というコマンドが用意されている．入力データはサンプルデータとして表 9 の tra.csv で key-value 型のデータを用いる．ここでは key が顧客で value が商品となる．アイテムとしては店–商品ペアではなく説明を簡単にするために商品のみを利用している．枝による接続は，2 アイテムの共起情報によって定義する．sim=R は，ノード間の Jaccard 係数を類似度として利用することを意味している．th は閾値で 0.4 にしている．この出力が表 10 の graph.csv であり，Jaccard 係数が 0.4 以上のノード同士が枝で結ばれていることが確認できる．これはグラフで表すと図 8 の実線の接続関係からなるグラフとして表される．

次にこのグラフにグラフ研磨を適用する．表 10 の graph.csv を入力として与えて mpolishing.rb を実行する．その際，枝を構成する二つのノード項目を ef=で指定し，類似度 sim と閾値 th に基づき枝を追加または削除するかを判断する．この例では，類似度に Jaccard 係数，閾値を 0.2 として指定した．表 11 はグラフ研磨の結果得られた枝を示している．接続関係として削除された枝はなく，図 8 の点線の枝が追加された．

最後にこの研磨後のグラフを mclique.rb の入力として与え極大クリークを列挙し，microcluster.csv に出力する．出力は { お茶, おにぎり, カップ麺, パン } と { カレー, パン, 牛乳 } で二つのマイクロクラスタが抽出された．グラフ研磨を適用せずに，表 10 の graph.csv から極大クリークを列挙すると，{ おにぎり, お茶, カップ麺 }，{ パン, 牛乳 }，{ おにぎり, パン }，{ カレー, 牛乳 } の四つの極大クリークが列挙される．グラフ研磨によって類似した接続関係をもつノード同士がまとめられるため列挙されるクリーク数が大きく削減できる．

このマイクロクラスタをダミー変数として回帰モデルの説明変数に利用することでモデルが構築できる．回帰モデルには LASSO が実行できる mglmnet.rb が利用可能であり，R の glmnet を NYSOL コマンドとし


```
1 mglmnet.rb k=顧客 x=MC v=val y=obj
2   i=traX.csv c=traY.csv -sparse  O=odir
3
4 mglmnet.rb -predict lambda=min i=testX.csv
5   -sparse  I=odir o=odir/rsl_prd.csv
```

図 9　mglmnet による LASSO の実行スクリプト

表 12　説明変数 (traX)

顧客, MC,val
10001,mc1,1
10001,mc2,1
10001,mc3,1
10002,mc4,1
10002,mc2,1
:

表 13　目的変数 (traY)

顧客, obj
10001,1
10002,0
10003,0
10004,1
10005,1
:

て Ruby によりラッピングした．図 9 は mglmnet.rb の実行方法を示しており，入力データとしては，表 12 の説明変数と表 13 の目的変数をそれぞれをファイルとして与える．ここでは説明変数を各顧客がもつマイクロクラスタとして MC で与えており，val はその値を示している．-sparse オプションで入力データが疎行列形式であることを指定し，R の glmnet によって val 以外の値は 0 として扱われる．つまりこの例では顧客 10001 の mc4 は 0 になる．そして目的変数を健康志向なら 1，そうでなければ 0 として，生成モデルを説明変数の MC で構築する．最後に生成モデルに対してテストデータを当てはめ目的変数の予測が行われる．

6.　企業における NYSOL の導入事例

昨今のビッグデータブームの下，データに価値を見いだし始めた企業は，データ取得とデータ解析に力を入れ始めており，データ解析は新たな価値の創出や，経営上の意思決定を行ううえで必要不可欠なものとして，多くの資源を投入し始めている．しかしながら，中小企業ではデータを利用した客観的評価や意思決定の重要性は認識しているものの，日々の業務に追われてデータ分析に資源を投入する余裕がないのが現状である．

われわれは深いドメイン知識を有する企業の現場担当者がデータ分析を行うことが，価値のある施策提案や真の課題解決，そして業務改善に繋がると考えており，中小企業の現場担当者にデータ分析のための技術とデータ分析の知識を提供する機会としてソレイユデータ道場[3]を立ち上げている．ソレイユデータ道場では，課題解決を希望する企業がプロジェクトを立ち上げ，所有するデータをプロジェクト参加メンバーに提供し，データ解析から得られる知見と個人のもつ強みを発揮することで課題解決を行う．プロジェクト参加者は，分析技術や分析ノウハウを実践から学び，プロジェクト提案企業は，得られた結果や新たな施策を基にビジネスに繋げる．NYSOL はメインツールとしてプロジェクトで利用されており，現在進行中のプロジェクトとして美容院と珈琲の Web 販売の取り組みを紹介し，その後，株式会社朝日オリコミ大阪の商圏分析の取り組みの概要を紹介する．

美容院経営における NYSOL の利用

関西で 6 店舗の美容院を経営しているある企業の取り組みを紹介する．近年，美容院の店舗数は年々増加しており，平成 25 年度には約 23 万 4 千件になり [6]，コンビニエンスストアの約 4 倍の店舗数となっている．分析対象となる美容院の徒歩圏内にも 10 店舗以上の美容院があり，安定した収益を得るためには，一度来店した顧客と長期的な関係を築いていくことが必要不可欠である．一般的に美容院による顧客の離反率は高いことが知られている．この美容院でも顧客の離反率を下げることが重要であり，新規顧客を繰り返し来店させ最終的に定着させることが経営上の課題である．この美容院では，この課題を解決するためにソレイユデータ道場に参加し，所有データを利用して，現場スタッフの仮説とデータ解析による検証を繰り返し実施している．

Web 珈琲販売企業における NYSOL の利用

コーヒーには苦味成分が含まれており，苦味は生得的に毒物のシグナルとして扱われてきたため，苦味を伴うコーヒーに対する嗜好は，その飲用経験によって不快から快へと嗜好が獲得され習慣化されることが多いと言われている．そのためコーヒーの嗜好は個人差が大きく，飲用経験によっても嗜好が変化することから，コーヒーの嗜好を適切に捉えることは難しい．そこでこの会社ではソレイユデータ道場に参加し，人間が感じるコーヒーの美味しさとは何かを追求しており，飲用アンケートとして日々のコーヒーの飲用履歴を取得し，そのデータを解析することで，現在，嗜好を考慮した推薦システムの構築を行っている．

株式会社朝日オリコミ大阪における NYSOL の利用

エリアマーケティングによって各企業はターゲットとする地域や出店する地域を定めたりする．株式会社朝日オリコミ大阪はエリアマーケティングのノウハウを所有しており，近年はより詳細なスマートフォンの

[3] http://www.soleildatadojo.com/

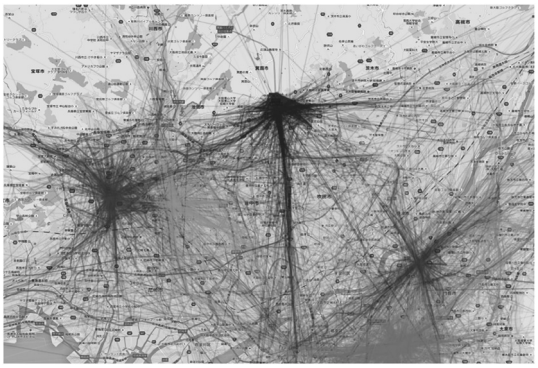

図 10　商業施設への移動経路（移動ログを利用）

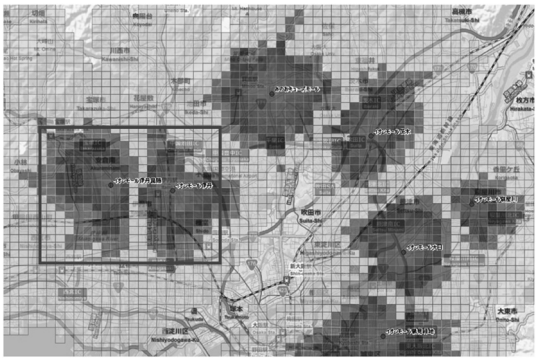

図 11　商業施設への移動経路（ハフモデルを利用）

移動ログデータを利用したエリアマーケティングの可能性を探っている．移動ログデータはある地域の 1 カ月のログで数億レコードにも及ぶ大規模なデータで，高速な処理と柔軟性が求められていることから，ログデータの分析に NYSOL を利用したトラジェクトリマイニングが行われている．

関西北摂エリアの大型ショッピングモールの商圏を可視化したものとして，移動ログデータを利用して可視化したもの（図 10）と，従来の商圏モデルであるハフモデルを可視化したもの（図 11）を比較している．商圏分析で用いられることの多いハフモデルでは，渋滞や，大きな道や鉄道による障害が考慮できておらず，四角で囲んでいる部分は，距離的に中間で商圏が分けられている．しかし移動ログでは同じ位置に属する左の商圏が広くなっており，JR と周辺の渋滞が障害になっていることが確認できる．したがって，移動ログによる商圏分析はより詳細に現状の把握が可能になるメリットがある．

7. おわりに

本稿は，データ分析のためのプラットフォームとして KDD の全プロセスを効率よく実現できるツールである NYSOL についてその利用方法を紹介した．そして，並列処理の実行方法とその性能を計算実験として示した．また応用研究として店舗選択モデルで利用したグラフ研磨手法を Take パッケージを用いて実行する方法を示し，企業での NYSOL を用いた取り組みを紹介した．

企業ではビッグデータの活用が重要視されており，今後ますますデータ分析のニーズは高まることが予想される．NYSOL は UNIX の設計思想に基づき一つのコマンドが一つの処理だけを行う非常にシンプルなもので，それらを組み合わせて柔軟で高速な処理を可能にしている．一度コマンドの使い方を理解すれば，手持ちの PC で 1 億件以上のビッグデータが扱える非常に強力なツールである．

謝辞　本研究の一部は，科学技術振興機構 CREST，および ERATO 湊離散構造処理系プロジェクトおよび，統計数理研究所平成 27 年度公募型共同利用「一般研究 1・27-共研-1031」の研究助成を受けている．株式会社朝日オリコミ大阪の福島孝志氏からは商圏分析の結果をご提供していただいた．ここに感謝の意を表します．

参考文献

[1] M. Gancarz, The UNIX Philosophy, Digital Press, 1994.（芳尾桂訳，『UNIX という考え方』，オーム社，2001.）
[2] 総務省，平成 27 年度情報通信白書，2015.
[3] 片桐孝洋，“1.1.5 性能評価指標，”『スパコンプログラミング入門』，p. 13, 東京大学出版会，2013.
[4] 中原孝信，羽室行信，宇野毅明，“グラフ研磨手法を用いた顧客の店舗選択モデルの構築，”オペレーションズ・リサーチ：経営の科学，**60**, pp. 89–95, 2015.
[5] 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.
[6] 厚生労働省，平成 25 年度衛生行政報告，2014.10.30 公表.

