JSPS-CNRS 二国間共同研究の枠組みでの学生派遣についての報告

浅井 大史 (panda@hongo.wide.ad.jp)

概要

JSPS-CNRS 二国間共同研究（代表：江崎浩，Patrice Abry）の枠組みでの学生派遣として，東京大学大学院江崎研究室博士 2 年の浅井大史が Ecole Normale Supérieure de Lyon (ENS-Lyon) で 1 ヶ月間 Patrice Abry らの下で研究活動を行った．研究内容はインターネットトラフィックによるネットワークアプリケーションのプロファイリングである．この課題に対するアプローチとして通信フローの時間的・空間的依存関係から通信パターンを有向グラフ（TCG: Traffic Causality Graph）として表し，グラフマイニング手法を適用した．本学生派遣では，グラフマイニング手法を TCG に適用することにより，各アプリケーションのトラフィックから特徴的な通信パターンを抽出することができた．また，様々なトラフィックデータに対する特徴量を計算し，アプリケーションと通信パターンの特徴量マトリクスを計算することで，TCG に対するグラフマイニング手法の適用によりアプリケーションプロファイリングが可能であることを示した．

1 はじめに

JSPS-CNRS 二国間共同研究においてフランス国立科学センター（CNRS: Centre National de la Recherche Scientifique）との協力活動があり，その一環として学生派遣を行っている．本年度は，東京大学大学院江崎研究室博士 2 年の浅井大史が対象者の 1 人として，2011 年 9 月 12 日から 2011 年 10 月 10 日までの 1 ヶ月間，フランスのリヨンにおいて研究活動を行った．受け入れ先は Ecole Normale Supérieure de Lyon (ENS-Lyon) の Patrice Abry らの研究室である．Patrice は，同大学の Pierre Borgnat とともに，WIDE プロジェクトメンバの江崎浩（東京大学），長健二郎（IIJ 技術研究所），福田健介（国立情報学研究所）との協力体制でインターネットトラフィック解析の研究を行っている．

本学生派遣における研究トピックは，インターネットトラフィックにおけるネットワークアプリケーションのプロファイリングに関する研究である．我々は通信フローの時間的・空間的依存関係から通信パターンを有向グラフ（TCG: Traffic Causality Graph）として表し，既存のグラフ解析手法を応用できるようなアプローチを提案した [1]．本学生派遣では，この有向グラフに対するグラフ解析手法として，グラフマイニングアルゴリズムである Subdue [2] を実装し，アプリケーションプロファイリングへの応用可能性を評価した．本研究の貢献は，TCG へのグラフマイニング手法の適用により，各アプリケーションのトラフィックから特徴的な通信パターンを TCG のサブストラクチャとして抽出することを可能としたことである．また，アプリケーションのトラフィックデータの抽出したサブストラクチャに対する特徴量を計算し，アプリケーションとサブストラクチャ（通信パターン）の特徴量マトリクスを計算することで，TCG によるアプリケーションプロファイリングにおけるグラフマイニング手法の応用可能性を示した．

2 TCG: Traffic Causality Graph

我々は，インターネットトラフィックによるネットワークアプリケーションのプロファイリングに対するアプローチとして，通信フローの時間的・空間的依存関係から通信パターンを有向グラフ（TCG: Traffic Causality Graph）として表す手法を提案した [1]．TCG では，従来の Iliofotou ら [3–5] や Jin ら [6] によるホスト間の通信相互作用をグラフとして表現する手法とは異なり，フロー間の時間的・空間的相互作用をグラフとして表現できるため，複数のプロトコルを用いるアプリケーションのプロファイリングが可能となっている．本研究では，各アプリケーションの特徴的な通信パターンの抽出およびアプリケーションプロファイリングの自動化を目指して，TCG に対してグラフマイニング手法を適用し評価する．本節では，インターネットトラフィックから TCG を構成するアルゴリズムについて概説する．

TCG の構成アルゴリズムは次の 3 つの要素からなる．

1. フローの取得：インターネットトラフィック（パケ

ットトレース）を取得し，5-tuple ⟨proto, srcIP, srcPort, dstIP, dstPort⟩ に基づきフローを得る．また，各フローの最初のパケットのタイムスタンプを各フローのタイムスタンプとして用いる．なお，本研究においては双方向通信における逆方向の通信はそれぞれ別々のフローとして扱うこととする．

2. TCG の構成：得られたフローからフロー間の時間的・空間的依存関係に基づき後述するアルゴリズムに従って TCG を構成する．
3. 辺の削減：得られた TCG から後述するヒューリスティクスに基づき，依存関係のない辺および解析に不必要な辺を取り除く．

TCG では，頂点および辺はそれぞれフローおよびフロー間の因果関係を表している．まず，TCG を構成するためのフロー間の因果関係として，次の 4 つを定義する（図 1）．

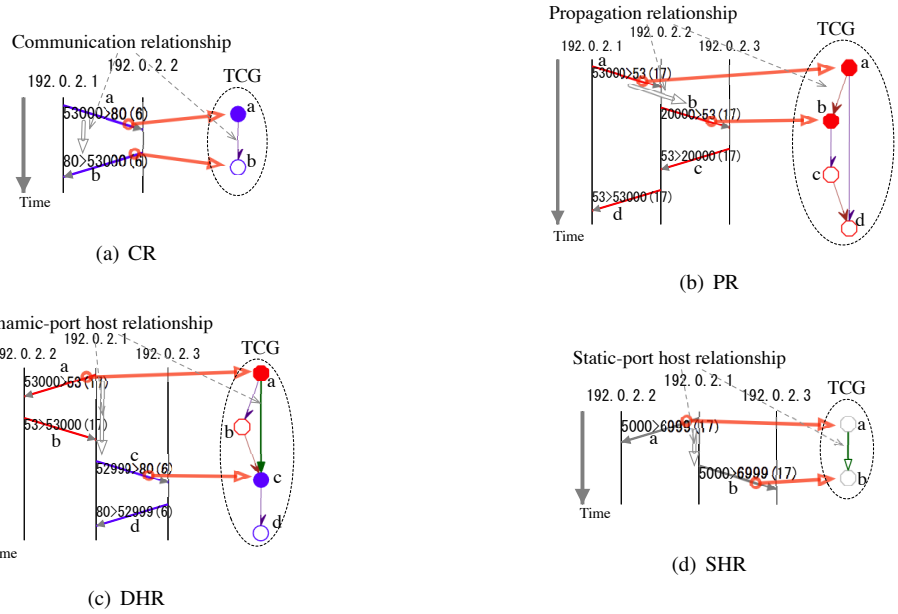
1. Communication relationship (CR)
2. Propagation relationship (PR)
3. Dynamic-port host relationship (DHR)
4. Static-port host relationship (SHR)

最初の 2 つの因果関係（CR および PR）はあるホスト（IP アドレス）が他のホストから受信したフローから他のホストへ送信するフローへの関係である．CR は TCP のような双方向通信におけるリクエストフローからレスポンスフローへの 1 対 1 の関係である．一方，PR はプロキシやリレーなどのように，あるホストが受信したフローから他のホストへのフローを誘引する多対多の関係である．残りの 2 つの因果関係（DHR および SHR）は，同一ホストから送出されるフロー間の関係であり多対多の関係となる．DHR は同一の srcIP で異なる srcPort のフロー間の関係であり，一般的な複数のコネクションを生成するクライアントアプリケーションの挙動である．一方，SHR は同一の srcIP かつ同一の srcPort のフロー間の関係であり，スキャン攻撃等の特殊な通信・アプリケーションや複数のコネクションを受け付けるサーバアプリケーションの挙動である．なお，以上の 4 つの因果関係以外にも何種類もの因果関係が定義可能ではあるが，本研究ではアプリケーションの挙動を特徴的に表しているこの 4 つに着目する．

TCG はアプリケーションの挙動を表すフローの依存関係を有向グラフとして可視化できる．本研究では，頂点の形は proto および計測点でのフローの向きを表す．三角形は出方向の ICMP，逆三角形は入方向の ICMP，二重円は出方向の TCP，単円は入方向の TCP，二重八角形は出方向の UDP，八角形は入方向の UDP を表す．辺の可視化については，CR はインディゴの半矢印，PR は茶色の通常矢印，DHR は緑色の塗りつぶし矢印，SHR は緑色の白抜き矢印で表す．

図 1 は本研究で定義した 4 つの因果関係と可視化の例を示している．ただし，図 1 の可視化ではフローの向きは全て入方向とした．なお，具体的な TCG の構成アルゴリズムは本節内で後述する．図 1(a) は HTTP によるサーバ・クライアント通信の挙動による CR の例である．クライアント 192.0.2.1 がサーバ 192.0.2.2 にリクエストを送り，その後，サーバがレスポンスを返すという通信である．レスポンスはリクエストにより引き起こされるため，このリクエストとレスポンスは因果関係があると言える．図 1(b) は DNS キャッシュサーバの挙動による PR の例である．クライアント 192.0.2.1 はキャッシュサーバ 192.0.2.2 に DNS リクエストを送り，その後，キャッシュサーバがそのリクエストを権威サーバ 192.0.2.3 へ伝達（再帰問い合わせ）する．キャッシュサーバから権威サーバへのフローはクライアントからのリクエストにより発生するものであるため，これら 2 つのフローには因果関係があると言える．図 1(c) は DNS と HTTP を用いた Web ページアクセスによる DHR の例である．クライアント 192.0.2.1 はまず DNS サーバ 192.0.2.2 により Web サーバの名前解決を行い，その後，解決された IP アドレス 192.0.2.3 に対して HTTP リクエストを送信する．DNS の名前解決は Web サーバへのアクセスの前に行われる必要があるため，クライアントから Web サーバへの HTTP リクエストフローはクライアントから DNS サーバへの DNS リクエストフローに依存している関係であり，これら 2 つのフローには因果関係があると言える．図 1(d) は静的な送信元ポート番号を使ったスキャン攻撃の挙動による SHR の例である．ホスト 192.0.2.1 は，順にホスト 192.0.2.2，ホスト 192.0.2.3 に対して同一の送信元ポート番号を用いてホストスキャンを行っている．

このように定義した 4 つの因果関係を元に各フローのタイムスタンプおよび 5-tuple の 6 つのパラメータを用いて TCG を構成する．関数 proto(f)，srcIP(f)，srcPort(f)，dstIP(f)，dstPort(f) がそれぞれフロー f の proto，srcIP，srcPort，dstIP，dstPort を返



(a) CR　(b) PR　(c) DHR　(d) SHR

図 1　4 つのフロー依存関係および TCG 可視化の例．TCG 可視化における頂点および辺はそれぞれフローおよび 2 つのフロー間の関係を表している．括弧で囲まれた数字はプロトコル番号（`proto`）を表す．すなわち，1 は ICMP，6 は TCP，17 は UDP である．

Algorithm 1 Get the type of relationship between flows f_1 and f_2

procedure getRelationship(f_1, f_2, τ):

1: **if** timestamp(f_2) − timestamp(f_1) > τ **then**
2:　return `Nil`
3: **end if**
4: **if** proto(f_1) = proto(f_2)
　　and srcIP(f_1) = dstIP(f_2) **and** srcPort(f_1) = dstPort(f_2)
　　and dstIP(f_1) = srcIP(f_2) **and** dstPort(f_1) = srcPort(f_2)
　then
5:　return `COMMUNICATION_RELATIONSHIP`
6: **else if** dstIP(f_1) = srcIP(f_2) **then**
7:　return `PROPAGATION_RELATIONSHIP`
8: **else if** srcIP(f_1) = srcIP(f_2) **and** srcPort(f_1) ≠ srcPort(f_2) **then**
9:　return `DYNAMIC_PORT_HOST_RELATIONSHIP`
10: **else if** srcIP(f_1) = srcIP(f_2) **and** srcPort(f_1) = srcPort(f_2) **then**
11:　return `STATIC_PORT_HOST_RELATIONSHIP`
12: **else**
13:　return `Nil`
14: **end if**

end procedure

し，関数 timestamp(f) がフロー f の最初のパケットの到着時刻を返すものとして，任意の 2 フロー間の因果関係を Algorithm 1 により決定する．連続しないフローについても同様に処理することに留意する．時間的に離れているフローは因果関係がないと考えることができるため，このアルゴリズムではまず閾値 τ により 2 フロー間の時間差を確認する（1–3 行目）．この閾値はアルゴリズム内で大域的に一意の値として設定される．また，この閾値は接続される辺の数を制限して計算量を削減するために設定されるが，後述する辺の削減とは独立に作用する．その後，このアルゴリズムは CR（4–5 行目），PR（6–7 行目），DHR（8–9 行目），SHR（10–11 行目）を順に確認する．2 フロー間に因果関係がない場合は，このアルゴリズムは `Nil` を返す（12–13 行目）．もしアルゴリズムが `Nil` ではない値を返した場合は，フロー f_1 から f_2 への辺を得られた因果関係によるラベル付けを行い TCG に追加する．

ここで，TCG の辺に関する用語を定義する．*CR-request* および *CR-response* はそれぞれ CR エッジのリンク元およびリンク先の頂点を示す．同様に，*PR-source*，*PR-destination*，*DHR-source*，*DHR-destination*，*SHR-source*，*SHR-destination* はそれぞれ PR エッジ，DHR エッジ，SHR エッジのリンク元およびリンク先の頂点を示す．

図 2 は，Microsoft Internet Explorer で `http://www.google.com/` にアクセスした際のトラフィックから Algorithm 1 に従って生成した TCG を可視化した図である．この図は辺を多く含んでいるためフロー間の因果関係を把握することは困難であるという問題点を明らかにしており，重要な辺以外を取り除く必要がある．Algorithm 1 は大きく次の二つの問題を抱えている．

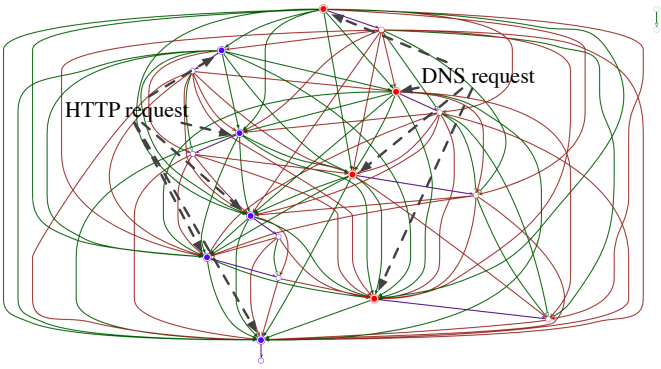


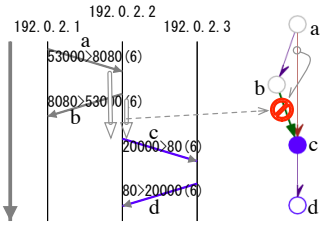
図 2　TCG の例（トラフィック：Microsoft Internet Explorer による http://www.google.com/ への Web アクセス; $\tau = 1$[s]; 頂点の色はプロトコル・ポート番号によって分けている（DNS (UDP/53) は赤色, HTTP (HTTP/80) は青色で表している））

1. PR，DHR，SHR エッジが大量に含まれており，そのほとんどは直接の因果関係ではない．
2. 単純なアルゴリズムであるため，因果関係のないフロー間にも辺が生成されてしまう．

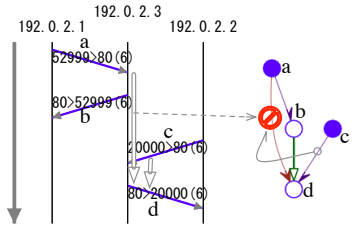
前者の問題は，PR，DHR，SHR が多対多の関係であり，各ホストは閾値 τ 時間内に複数のフローを生成しうるため生じる．本研究ではこれらの直接の因果関係のない辺を *tenuous edges* と呼ぶ．後者の問題は，Algorithm 1 が非常に単純なルールに基づくものであるため，直接因果関係のない辺を生成してしうるため生じる．例えば，このアルゴリズムに従うと CR-response は PR-destination にもなり得るが，CR-response は明らかに PR-source ではなく対応する CR-request により生じているものであるため，このような辺は取り除くべきである．本研究ではこれらの辺を *irrelative edges* と呼ぶ．これらの二つの問題に加え，プロファイリングの対象によって任意で辺を削除するルールを設定する．例えば，クライアントの挙動に着目した場合は，サーバの挙動（CR-response から別の CR-response への DHR/SHR）は任意で取り除くことができる．これを実現するために，以下で三つのヒューリスティクスに基づく辺の削減ルール（ER-Rule: Edge Reduction Rule）を導入する．

- ER-Rule 1. **tenuous edges の除去：**PR, DHR, SHR エッジを削減するために，それぞれの因果関係について時間的に最も近い辺以外の PR, DHR, SHR エッジを除去した．すなわち，それぞれの因果関係について各頂点からの最大出次数が 1 となるようにした．このルールは時間的に最も近いフローが同じアプリケーションから生成され直接的な因果関係を有するというヒューリスティクスに基づいている．
- ER-Rule 2. **irrelative edges の除去：**単純なアルゴリズムである Algorithm 1 を補完するために，近隣の辺を調べることにより irrelative edges を削除する．CR-response から CR-request の DHR/SHR エッジは，図 3(a) に示すように PR エッジが存在するべきであり，CR-response は CR-request を発生する要因ではないと考えられるため，これらを除去する．すなわち，PR-source は CR-request（この場合 PR-destination でもある）を発生する要因である．同様に，図 3(b) に示すように CR-response は対応する CR-request により発生するものであるため，CR-request から CR-response への DHR/SHR エッジおよび CR-response への PR エッジも除去する．CR-response から他の CR-response への DHR/SHR エッジについては，ただし，サーバの挙動として残し，ER-Rule 3(b) により任意で取り除くことができるものとする．
- ER-Rule 3（任意）. **(a) CR-response からの PR エッジの除去：**PR-destination が PR-source ではなく，CR-request により発生しているものと考える場合，図 4(a) に示すように，CR-response からの PR エッジを除去することができる．**(b) サーバアプリケーションの挙動の除去：**クライアントの挙動にのみ着目する場合，CR-response から CR-response への DHR/SHR エッジはサーバの挙動として図 4(b) に示すように任意で除去することができる．**(c) クライアントアプリケーションの挙動の除去**サーバアプリケーションの挙動の除去と同様に，サーバの挙動にのみ着目する場合，CR-request から CR-request への DHR/SHR エッジはクライアントの挙動として図 4(c) に示すように任意で除去することができる．ただし，クライアントの挙動はアプリケーションの挙動を表すのにより重要な情報であるため，ER-Rule 3(c) が適用されることは稀であると考える．

図 5 は図 2 で用いたものと同じトラフィックに対して ER-Rule 1, 2, and 3(a) を適用した TCG である．この図からは Web ブラウザが HTTP アクセスの直前にドメイン名を解決している挙動が明白に可視化されている．

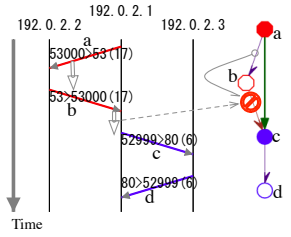


(a) CR-response から CR-request への DHR/SHR エッジ

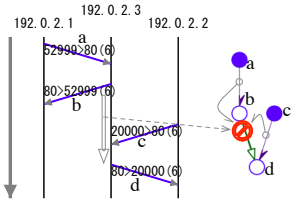


(b) CR-response への PR エッジおよび CR-request から CR-response への DHR/SHR エッジ

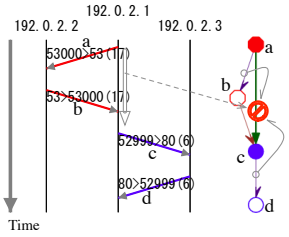
図 3　ER-Rule 2: irrelative edges の除去



(a) CR-response から PR エッジ



(b) サーバの挙動の除去



(c) クライアントの挙動の除去

図 4　ER-Rule 3: プロファイリング対象ごとに対応した任意の辺除去

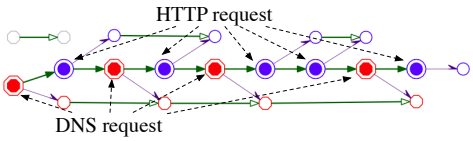


図 5　辺の除去を行った後の TCG （トラフィック：図 2 と同じもの; $\tau = 1$[s]; ER-Rule：1, 2, and 3(a)）

このように，辺の除去ルールを適用することで，可視化の点を含めてフロー間の因果関係の表現力を向上している．詳細な解析に際して，除去された辺はアプリケーションをプロファイリングを行う上で特徴的であるかもしれないが，残った辺と比較して重要ではないと考えられるため，グラフマイニングを用いた解析においてもこの辺の削減ルールを適用する．

3　成果

本学生派遣では，グラフマイニングアルゴリズムである Subdue [2] を実装し TCG への適用を行うことで，グラフマイニング手法のアプリケーションプロファイリングへの応用可能性を評価した．本節では TCG へのグラフマイニング手法の応用について報告する．

3.1　グラフマイニング手法

グラフマイニングにおいては，特徴的なサブストラクチャ（パターン）を抽出するための指標として，グラフをあるサブストラクチャで簡約化したグラフを記述するのに必要な情報量である Description Length (DL) を最小とするサブストラクチャを発見する手法（MDL: Minimum Description Length）が広く用いられている．本研究では，特徴的なサブストラクチャを抽出するアルゴリズムとして MDL に基づいた Subdue アルゴリズム [2] を用いる．

Subdue アルゴリズムでは，V を頂点の集合，E を辺の集合，L をラベル（頂点・辺の両方のラベル）の集合[*1]として，ラベル付き有向グラフ $G = (V, E)$ に対する記述長（DL: Description Length）を頂点，辺，隣接行列の行，

[*1] 論文 [2] では，頂点と辺のラベルを別々に扱っているが，著者らが公開しているソースコード [7] では同じ集合として扱っているため，本研究では後者のアルゴリズムに従った．

それぞれの記述長の和により以下のように定義する．

$$DL = h_{vertex} + h_{edge} + h_{row}$$

$$\text{s.t.} \begin{cases} h_{vertex} & = \lg(\|V\|) + \|V\| \lg(\|L\|) \\ h_{edge} & = \|E\| \lg(1 + \|L\|) + (K+1) \lg(m) \\ h_{row} & = (\|V\| + 1) \lg(b+1) + \sum_{i=1}^{\|V\|} \lg \binom{\|V\|}{k_i} \end{cases}$$

ここで，m はグラフの最大出次数（outdegree）とし，また，b および K は，グラフ G の隣接行列を $A = (a_{ij})$ として次の式により定義されるものとする．

$$b = \max_i k_i$$

$$K = \sum_{i=1}^{\|V\|} k_i$$

$$\text{s.t. } k_i = \sum_{j=1}^{\|V\|} a_{ij}$$

MDL では，サブストラクチャ S でグラフ G を簡約化したグラフの記述長 $DL(G|S)$ を最小とするサブストラクチャを探索するが，Subdue ではこの MDL を拡張し，サブストラクチャ S のグラフ G に対する特徴量 $f(S,G)$ を次のように定義し，この特徴量に基づき特徴的なサブストラクチャを抽出する．

$$f(S,G) = \frac{DL(G)}{DL(S) + DL(G|S)} \tag{1}$$

この特徴量 $f(S,G)$ が小さいほど，サブストラクチャ S はグラフ G をよりよく説明する特徴的なサブストラクチャであると言えるため，Subdue ではこの特徴量を最大化するサブストラクチャを探索する．

Subdue において，上述した特徴量 $f(S,G)$ を最大化するサブストラクチャの探索方法は 1 頂点のサブストラクチャから拡張して再帰的に探索するビーム探索法によるものであり，グラフ以外の入力パラメータとしては，ビーム探索における探索の幅，最大探索試行回数，抽出サブストラクチャの最大数，抽出するサブストラクチャの最小頂点数および最大頂点数の五つを与える．

3.2 頂点へのラベル付け

Subdue を TCG に適用するにあたって頂点へのラベル付けが必要となる．本研究では次の二つのラベル付け方法を用いて評価した．

1. プロトコル・ポート番号に基づくラベル付け
2. フローの統計量に基づくラベル付け

前者のプロトコル・ポート番号は従来のトラフィック分類手法 [8] でも用いられており，予備実験 [1] からもプロトコル・ポート番号に基づく特徴パターンはアプリケーションプロファイリングに有用であることが示されている．しかし，P2P などポート番号を動的に利用するアプリケーションも多く存在するため，後者のプロトコル・ポート番号に依らないフローの統計量に基づく頂点ラベル付けについても評価する．

3.2.1 プロトコル・ポート番号に基づくラベル付け

予備実験 [1] の結果から HTTP（TCP/80, TCP/443）フローおよび DNS（UDP/53）フローはアプリケーションプロファイリングに効果的であることが分かっているため，プロトコル・ポート番号に基づく頂点へのラベル付けでは以下の 7 種類のラベルを用いる．

1. DNS リクエスト
   （DNS/Req: `proto`=17, `dstPort`=53）
2. DNS レスポンス
   （DNS/Res: `proto`=17, `srcPort`=53）
3. HTTP リクエスト
   （HTTP/Req: `proto`=6, `dstPort`=80/443）
4. HTTP レスポンス
   （HTTP/Res: `proto`=6, `srcPort`=80/443）
5. その他 TCP（Other/TCP: `proto`=6）
6. その他 UDP（Other/UDP: `proto`=17）
7. その他（Other）

3.2.2 フローの統計量に基づくラベル付け

本研究におけるフローの統計量に基づくラベル付けでは，フローの統計量として各フローのサイズ（パケットサイズの総和）および各フローに含まれるパケット数で正規化したパケットサイズのエントロピーを用いる．正規化したパケットサイズのエントロピー H' は次の式で定義する．

$$H' = \frac{-\sum_i (p_i \log_2 p_i)}{\log_2 P} \tag{2}$$
$$(0 \le H' \le 1)$$

ここで，P はフローに含まれるパケット数，p_i はパケットサイズの確率分布とする．ただし，確率分布 p_i はビン幅 b で与えられるものとする．

フローサイズは 1 フローあたりの通信量を表しており，アプリケーション間で交換されるクエリやデータの特徴を良く表していると言える．しかし，アプリケーションの中には 1 フローの中で複数のメッセージをやりとりする場合もあるため，パケットサイズのエントロ

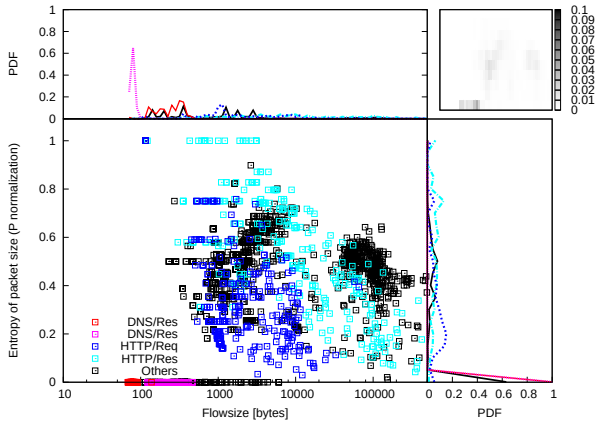


図 6　評価に用いるデータセットに含まれるフローのサイズおよびパケットサイズのエントロピーの分布

表 1　フローの統計量に基づくラベル付け

エントロピー \ フローサイズ [bytes]	< 700	< 20000	≥ 20000
$0 \leq H' < 0.07$	Class-1	Class-2	Class-3
$0.07 \leq H < 0.4$	Class-4	Class-5	Class-6
$0.4 \leq H' \leq 1$	Class-7	Class-8	Class-9

ピーによりフローの種類を分類する．パケットサイズのエントロピーが大きい場合は，大きさの異なるパケットを含むことから 1 フロー内で複数のクエリやメッセージ交換の特徴を表しており，逆にパケットサイズのエントロピーが小さい場合は，同じ大きさのパケットを多く含むことから 1 フロー内で大きなデータ（コンテンツ）を交換している特徴を表している．

本研究では，図 6 に示した評価に用いるデータセット（4 節）に含まれるフローのサイズおよびパケットサイズのエントロピーの分布から，フローのラベル付けに用いる二つの統計量についてそれぞれ確率密度分布（PDF）の谷となる点に閾値を設定して表 1 に示した通りに 9 個のクラスに分類するものとする．例えば，DNS などの非常に短いクエリ・レスポンス交換は Class-1 に分類され，1 フローでの大きなデータ配送は Class-3 に分類される．一方，1 フロー内で複数の通信が行われる HTTP の keep-alive フローなどはパケットサイズのエントロピーが大きい Class-5，Class-6，Class-8 などに分類される．なお，この統計量に基づくラベル付けについては今後改善の必要があると考えるが，本研究においてはプロトコル・ポート番号に依らないフローの統計量に基づくアプリケーションプロファイリングの可能性を評価するために非常に単純ではあるが表 1 を分類に用いた．

4　評価

本節では 3 節で述べたグラフマイニング手法の TCG によるアプリケーションプロファイリングへの応用可能性について評価する．本研究のアプリケーションプロファイリングにおいては，まず各アプリケーションの TCG から特徴的な通信パターン（サブストラクチャ）を抽出（学習）し，式 (1) によりプロファイリング対象のトラフィックの抽出したそれぞれのサブストラクチャに対する特徴量ベクトルを計算し，特徴量の距離によりプロファイリングを行う．

なお，本研究では評価用のトラフィックデータセットとして，5 種類の Web ブラウザ（Microsoft Internet Explorer, Mozilla Firefox, Google Chrome, Opera, Safari）の通常の Web ページ閲覧および動画サイト[*2]の閲覧（つまりそれぞれ 2 種類のトラフィックが得られる），5 種類の P2P ファイル共有アプリケーション（BitTorrent, Cabos, Winnyp, Share, Perfect Dark）および 1 種類の P2P ライブストリーミングアプリケーション（BBBroadcast）の 11 種類のアプリケーションをそれぞれ使用し，合計 16 種類の実トラフィックデータを取得した．なお，それぞれのアプリケーションは Windows 7 上で実行した．表 2 にそれぞれのトラフィックデータについてまとめた．

本研究のアプリケーションプロファイリングを評価するために，まず，3 節で説明した Subdue を用いて，評価用のトラフィックデータセット 16 種類のそれぞれから閾値 $\tau = 1$ として得られた TCG から特徴的な通信パターンを TCG のサブストラクチャとして抽出した．グラフ以外の入力パラメータである，ビーム探索における探索の幅，抽出サブストラクチャの最大数，抽出するサブストラクチャの最小頂点数および最大頂点数はそれぞれ 128，2，1，inf とし，最大探索試行回数はグラフに含まれる辺の数の 1/2 とした．

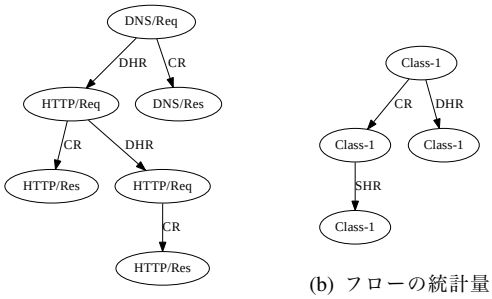
グラフマイニング手法の適用例として，図 7，図 8 に ie-youtube および bittorrent-debian それぞれのデータセットの TCG から Subdue により発見された特徴量 $f(G, S)$ を最大とするサブストラクチャを示す．図 7(a) は DNS による名前解決の後に HTTP によりコンテンツを取得する典型的な Web ブラウザの挙動を抽出できていることを示している．しかし，BitTorrent のトラフィックデータから得られた TCG の特徴的なサブストラクチャである図 8(a) からは送信元ポートを固定した同一ホスト

[*2] www.youtube.com

表 2　評価用トラフィックデータセット

プログラム	データセット略称	種別	トラフィック
Microsoft Internet Explorer	ie/ie-youtube	Web ブラウザ	Web ページ/動画サイトを閲覧
Mozilla Firefox	firefox/firefox-youtube	Web ブラウザ	Web ページ/動画サイトを閲覧
Google Chrome	chrome/chrome-youtube	Web ブラウザ	Web ページ/動画サイトを閲覧
Opera	opera/opera-youtube	Web ブラウザ	Web ページ/動画サイトを閲覧
Safari	safari/safari-youtube	Web ブラウザ	Web ページ/動画サイトを閲覧
BitTorrent	bittorrent-debian	P2P ファイル共有	起動後，ファイルをダウンロード
Cabos	cabos	P2P ファイル共有	起動後，検索・ファイルダウンロード
Winnyp	winny	P2P ファイル共有	起動後，検索・ファイルダウンロード
Share	share	P2P ファイル共有	起動後，検索・ファイルダウンロード
Perfect Dark	pd	P2P ファイル共有	起動後，検索・ファイルダウンロード
BBbroadcast	bbb	P2P ライブストリーミング	起動後，ストリーミング動画を試聴 †

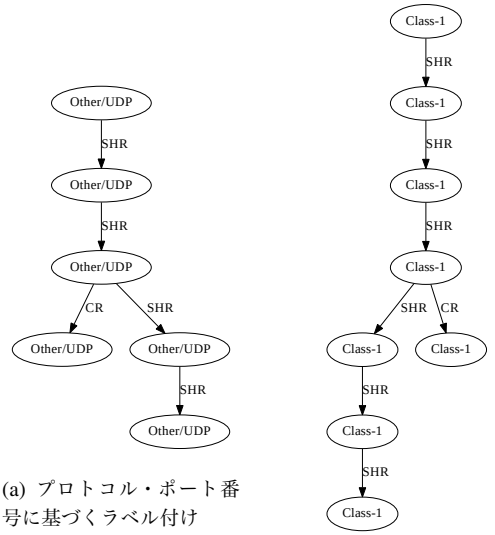
† Microsoft Internet Explorer のプラグインとして使用



(a) プロトコル・ポート番号に基づくラベル付け

(b) フローの統計量に基づくラベル付け

図 7　発見された最も特徴的なサブストラクチャ（データセット：ie-youtube）



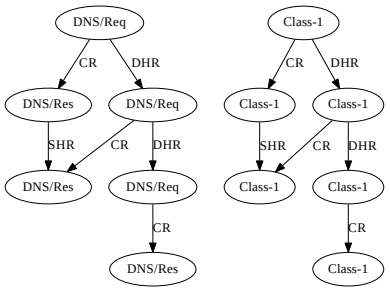
(a) プロトコル・ポート番号に基づくラベル付け

(b) フローの統計量に基づくラベル付け

図 8　発見された最も特徴的なサブストラクチャ（データセット：bittorrent-debian）

からの UDP フロー（SHR）とホスト間の通信（CR）を確認できるが，これからクエリ交換の挙動なのかコンテンツ交換の挙動なのかを区別することは困難である．一方，フローの統計量に基づくラベル付けを行った図 8(b) からは，Class-1 のフローが特徴的なサブストラクチャとして抽出されており，このサブストラクチャが小さなクエリ交換を表していることがわかる．

次に，このように 16 種類のトラフィックデータの TCG から二つずつ抽出された合計 32 個のサブストラクチャを用いたアプリケーションプロファイリングの可能性について評価する．式 (1) により抽出したそれぞれのサブストラクチャに対するプロファイリング対象のトラフィックデータの特徴量ベクトルを計算する．図 9 に抽出された 32 種類のサブストラクチャに対する 16 種類のアプリケーショントラフィックの特徴量マトリクスを示す．なお，サブストラクチャは#1 から#32 までの番号で表している．各要素の軸上の点はまでの特徴量の値を表しており，軸上は 0.5 から 1.5 の特徴量，軸の左端は 0.5 以下，軸の右端は 1.5 以上を意味する．なお，各要素の背景色は，特徴量 $f(G,S)$ が 1.1 以上の場合は緑色，特徴量 $f(G,S)$ が 0.9 以下の場合は赤色，その他は黄色で表している．また，軸上の青色の範囲は各トラフィックデータセットを 5 秒ごとに区切った場合のその特徴量 $f(G,S)$ の四分位数（25% 値，75% 値）を表している．この特徴量マトリクスからそれぞれのアプリケーションの特徴量ベクトルを比較することでアプリケーションプロファイリングを行うことができる．例えば，図 9(a) における#6 のサブストラクチャ（図 10(a)）は事前実験 [1] における連続した DNS リクエストを表しており，DNS プリフェッチを積極的に行う Firefox，Google Chrome および Safari でこのサブストラクチャに対する特徴量が大きくなっていることが確認できる．また，同様のサ



(a) 図 9(a) 中のサブストラクチャ#6　(b) 図 9(b) 中のサブストラクチャ#6

図 10　プロファイリングに効果的なサブストラクチャ

ブストラクチャは統計量による頂点へのラベル付けを行った際にも図 9(b) 中のサブストラクチャ#6（図 10(b)）として現れ，同様に DNS プリフェッチを積極的に行う Firefox，Google Chrome および Safari でこのサブストラクチャに対する特徴量が大きくなっている．このように，抽出されたサブストラクチャに対する特徴量を比較することによりアプリケーションのプロファイリングが可能である．

次に特徴量マトリクスからアプリケーションが分類可能かを評価するために，特徴量マトリクスからアプリケーション（トラフィックデータ）間の距離を計算し，階層的クラスタリングを行った．図 11 に図 9(a) の特徴量マトリクスから最遠隣法により階層的クラスタリングを行った結果を示す．この結果から，Web ブラウザと P2P ファイル共有アプリケーションと言ったようにマクロな分類は階層化クラスタリングにより達成できているが，前述した Web ブラウザにおける積極的な DNS プリフェッチの差異による分類やブラウザのプラグインとして動作するアプリケーション（BBBroadcast）の分類は達成できていないと言える．これは頂点へのラベル付け方法に依らず生じる問題であり，この原因については次の二つが考えられる．まず一つ目は，それぞれのサブストラクチャが独立なサブストラクチャではなく他のサブストラクチャの包含関係になっていることがあるため，階層化クラスタリングにおける特徴量ベクトル間の距離計算において一部のサブストラクチャの特徴量が結果に大きく影響することである．二つ目の原因としては，サブストラクチャによってはアプリケーション固有の挙動ではなく，コンテンツの違いや利用者の振る舞いを表しているためである．例えば，図 11 において同じアプリケーションでも Web ページの閲覧と動画サイトの閲覧では異なったクラスタに分類されてしまっている．つまり，アプリケーションプロファイリングにおいては，アプリケーションの挙動，ユーザの振る舞い，コンテンツの違い，それぞれの次元でプロファイリングに用いるサブストラクチャを決定し，さらにサブストラクチャの包含関係を考慮した重み付けが必要である．

5　今後の課題

頂点のラベル付け：本研究では，3.2 節で説明した 2 種類の頂点へのラベル付け方法を用いた．これは，DNS のように固定されたポート番号を使うプロトコルの場合はプロトコル・ポート番号によるラベル付けが非常に有効であると考えるが，HTTP のように同じポート番号を様々なアプリケーション（Web ブラウザ以外にカレンダーアプリケーションなど）が使い，多種多様なデータが交換される場合や P2P のように動的にポート番号を変更するアプリケーションに対してはフローの統計量に基づくラベル付けと組み合わせることが望ましいと考えたためである．しかし，図 9(a) および (b) から，P2P アプリケーションの TCG についてフローの統計量に基づくラベル付けではなくプロトコル・ポート番号に基づくラベル付けにより特徴量マトリクスを計算した場合においても，各サブストラクチャに対する特徴量に違いが生じていることがわかる．このことは，TCG の辺のラベルおよび TCG の構造がアプリケーションの挙動をよく表現しているとも言えるが，これが最適ではないと考える．様々なアプリケーションに対して適した頂点へのラベル付け手法については，今後，調査検討する．

サブストラクチャの分類：4 節で指摘したように，グラフマイニング手法で抽出したサブストラクチャの中には，アプリケーションの挙動ではなく，ユーザの振る舞いやコンテンツの違いを表しているものもある．また，それぞれのサブストラクチャが独立なサブストラクチャではなく他のサブストラクチャの包含関係になっていることがあるため，階層化クラスタリングにおける特徴量ベクトル間の距離計算において一部のサブストラクチャの特徴量が結果に大きく影響してしまう可能性がある．そのため，アプリケーションプロファイリングにおいては，アプリケーションの挙動，ユーザの振る舞い，コンテンツの違い，それぞれの次元でプロファイリングに用いるサブストラクチャを決定し，さらにサブストラクチャの包含関係を考慮した重み付けが必要である．

プロファイリングプロセスの自動化：本研究においては，グラフマイニング手法により特徴的なサブストラク

チャを抽出し，各アプリケーションのトラフィックデータについてそれらのサブストラクチャに対する特徴量ベクトル（マトリクス）を計算してアプリケーションプロファイリングが可能であることを示したが，プロファイリングプロセスの自動化については議論しなかった．まず特徴的なサブストラクチャの抽出であるが，上述したように抽出したサブストラクチャの分類と重み付けを行う必要がある．このプロセスの自動化についてはまず教師あり学習のアプローチを検討する．なお，サブストラクチャの分類と重み付けが実現できれば，各サブストラクチャに対する特徴量ベクトル（マトリクス）からアプリケーションの分類・プロファイリングは自動化できると考える．

多種多様なデータセットおよび実トラフィックへの応用：本研究では 16 種類のトラフィックデータを評価用のデータセットとして用いたが，今後，より多種多様なデータセットに対しても解析を行っていく．また，実トラフィックデータへの応用可能性についても評価する．

6 まとめ

JSPS-CNRS 二国間共同研究（代表：江崎浩，Patrice Abry）の枠組みでの学生派遣として，東京大学大学院江崎研究室博士 2 年の浅井大史が Ecole Normale Supérieure de Lyon (ENS-Lyon) で 1 ヶ月間 Patrice Abry らの下で研究活動を行った．本学生派遣において，ネットワークアプリケーションのプロファイリングを目的として，通信フローの時間的・空間的依存関係から通信パターンを有向グラフとして表した TCG に対して，グラフマイニングアルゴリズムを実装し適用した．本研究の貢献は，各アプリケーションのトラフィックの TCG からプロファイリングに有効な特徴的なサブストラクチャ（通信パターン）を抽出できることを示したことである．また，16 種類の評価用のトラフィックデータを用いて，特徴的なサブストラクチャに対する特徴量マトリクスを計算することで，アプリケーションプロファイリングへの応用可能性を示した．今後は，プロファイリングの自動化を目指して本研究活動で発見された課題に対して取り組む．

参考文献

[1] H. Asai, K. Fukuda, and H. Esaki, “Traffic Causality Graphs: Profiling network applications through temporal and spatial causality of flows,” in *Proceedings of the 23rd International Teletraffic Congress*, ITC ’11, pp. 95–102, ITCP, 2011.

[2] L. B. Holder, D. J. Cook, and S. Djoko, “Substructure discovery in the subdue system,” in *Workshop on Knowledge Discovery in Databases*, pp. 169 –180, 1994.

[3] M. Iliofotou, P. Pappu, M. Faloutsos, M. Mitzenmacher, S. Singh, and G. Varghese, “Network monitoring using traffic dispersion graphs (TDGs),” in *ACM IMC ’07*, pp. 315–320, 2007.

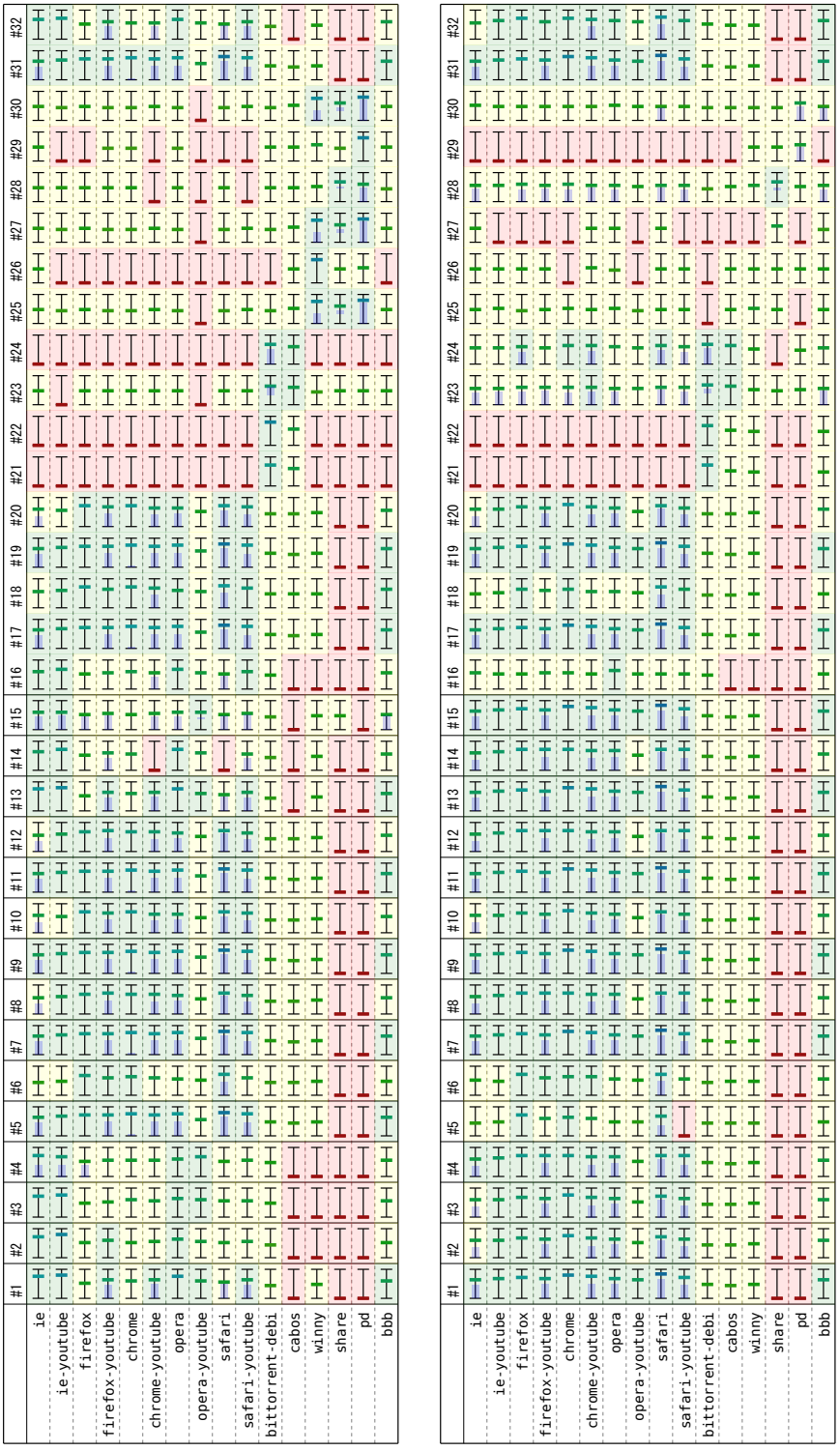
[4] M. Iliofotou, M. Faloutsos, and M. Mitzenmacher, “Exploiting dynamicity in graph-based traffic analysis: Techniques and applications,” in *ACM CoNEXT ’09*, pp. 241–252, 2009.

[5] M. Iliofotou, B. Gallagher, T. Eliassi-Rad, G. Xie, and M. Faloutsos, “Profiling-by-association: A resilient traffic profiling solution for the Internet backbone,” in *CoNEXT ’10*, p. 12, ACM, 2010.

[6] Y. Jin, E. Sharafuddin, and Z.-L. Zhang, “Unveiling core network-wide communication patterns through application traffic activity graph decomposition,” in *ACM SIGMETRICS ’09*, pp. 49–60, 2009.

[7] D. J. Cook and L. B. Holder, “SUBDUE – graph based knowledge discovery.” http://ailab.wsu.edu/subdue/.

[8] T. Karagiannis, A. Broido, M. Faloutsos, and K. claffy, “Transport layer identification of P2P traffic,” in *ACM IMC ’04*, pp. 121–134, 2004.



(a) プロトコル・ポート番号に基づくラベル付け　　(b) フローの統計量に基づくラベル付け

図 9　抽出された 32 種類のサブストラクチャに対する各アプリケーショントラフィックの特徴量マトリクス

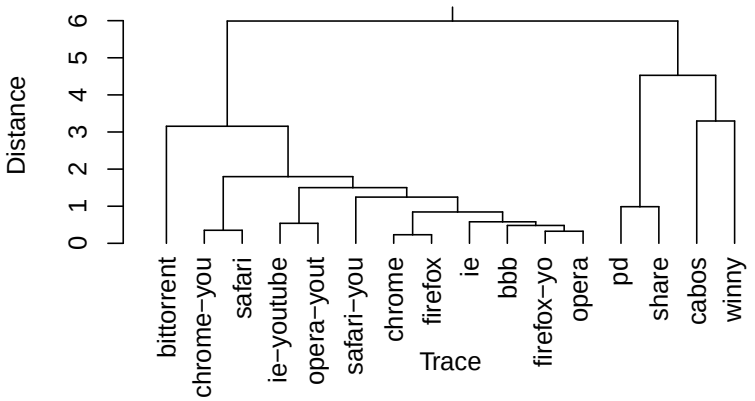


図 11　特徴量マトリクスからの階層的クラスタリング（最遠隣法）によるプロファイリング結果