科学研究費助成事業　　研究成果報告書

平成　２６　年　　６　月　１０　日現在

機関番号：　３４５０４
研究種目：　若手研究(A)
研究期間：　2010～2013
課題番号：　２２６８０００７

研究課題名（和文）表構造の異なる複数の時区間履歴データからの時系列分析多次元データベースの構築手法

研究課題名（英文）Development of Multidimensional Databases for Analyzing Time Interval Data in Heterogeneous Schemas

研究代表者
猪口　明博（Akihiro, Inokuchi）

関西学院大学・理工学部・准教授

研究者番号：７０４５２４５６

交付決定額（研究期間全体）：（直接経費）　　17,600,000円、（間接経費）　　5,280,000円

研究成果の概要（和文）：既存のOLAPシステムでは，時区間からなるイベント時系列データを解析することは困難である．本研究では，医療データ，Web閲覧データ，購買データ，単語の系列である文書など，時区間からなるデータを対話的に分析するために，HealthCubeと呼ばれるOLAPシステムを構築した．HealthCubeでは，垂直レイアウトの表にデータを格納し，時区間のためのOLAP演算を用いることでデータを多次元に分析することができる．さらに，HealthCubeに格納されたデータを多数の計算機に分割し，並列して処理を実行することで，線形速度向上を実現した．また，大規模なデータに対して本手法を適用し，実用性を検証した

研究成果の概要（英文）：The current OLAP systems are incapable of supporting to analyze event time series with temporal intervals. In the last years, we have developed an OLAP system, called HealthCube, to interactively analyze event time series with temporal intervals such as medical data, buying history data, click stream data for Web browsing, sequences of words and so on. It is to analyze the medical data in a multidimensional manner by storing the data in tables of vertical layout and by designing OLAP operations for temporal intervals. In addition, we achieved a linear speedup by partitioning the data into multiple computers and by paralleling the operations with these computers, and we confirmed the practicality of our system to huge databases.

研究分野：　情報学

科研費の分科・細目：　メディア情報学・データベース

キーワード：　多次元データベース　OLAP　データマイニング

１．研究開始当初の背景

1990 年代より多次元データベース(DB)に関する研究は盛んに行われてきた．典型的な応用例は，大型小売業界で収集される購買履歴データを対象として，いつ，どこで，どのような商品がどのくらい購買されたかの集計(統計)情報を計算することである．分析の対象となるデータは数テラバイト規模のデータで，分析者は対話的に試行錯誤しながら分析するために，1 度の分析(問合せ)に対して，10 秒以下の応答時間であることが望まれる．既存多次元 DB の研究では，分析対象データを格納するファクト表(履歴情報)の属性(変数)間の集計情報を求めることが，主な目的であった．2008 年に S-OLAP[Eric Lo, et. al. OLAP on Sequence Data. SIGMOD Conference 2008, pp.649-660]が提案され，ファクト表の履歴(事象)間の集計情報を求めることが可能となった．例えば，S-OLAP により同一の顧客がどのような順番で商品を購入したかを示す事象系列を高速に集計することが可能となった．しかし，S-OLAP の枠組みでは単一のファクト表から事象系列の集計を対象としているため，複数のファクト表から事象系列を集計することは困難である．また，S-OLAP では事象間に時間的全順序関係が成り立つ事象系列しか集計できないため，全順序関係ではなく半順序関係のみが成り立つ事象系列の集計を行うことは困難である．更に，購買などの事象の履歴を対象としているため，時間幅を持つ時区間データを扱うことができない．例えば，診療履歴データでは，入退院履歴，病歴，診察履歴，投薬履歴，手術履歴，検体検査履歴などは，構造が異なる表に別々に格納され，入退院情報は入院時間から退院時間までの時区間データ，病歴は発症から治癒(死亡)までの時区間により表される．また，疾病に至る要因は複数あるが，その要因には必ず時間的全順序関係が成り立つわけではないので，複雑な分析を可能にするには，事象間に半順序関係のみが成り立つ事象系列の集計も必要となる．

２．研究の目的

本提案では，表構造の異なる複数の時区間履歴データを対象として，多次元データベースを構築するための枠組みを考案することを第 1 の目的とする．また，大規模時区間履歴データから多次元データベースを構築するための並列化手法を考案することを第 2 の目的とする．さらに，考案した手法の研究プロトタイプを作成し，産業界の実データにおいて実用性に関する検証実験を行うことを第 3 の目的とする．

３．研究の方法

分析の対象データ D を D={(fi, {pi1,pi2,…, pim})} (i=1,2,…,n)と定義する．{fi |i=1,2,…, n}はデータ ID の集合であり，pij は区間情報であるとする．(fi,{pi1,pi2,…,pim})は各患者 fi が区間に関する情報 pij の集合をもつことを意味する．区間を pij=(ts,te,c:v)と定義する．ここで，ts，及び te は区間の開始時刻，及び終了時刻とする．特に，ts=te であるとき，区間 pij をイベントと呼ぶ. v は区間を説明する値であり, c は値 v の属するカテゴリとする．また c は階層を持つデータのある節点である．

分析の対象とするデータを患者データであるとした場合の具体例は，{pij}が入退院に関する区間(期間)であれば，c:v は，入院中病名，主治医などからなる．また，病名のカテゴリは階層構造をもつ国際疾病分類（ICD：International Classification of Diseases）とする．ICD のカテゴリ”C00.1 唇の悪性新生物，外側上唇”は中分類”C00 唇の悪性新生物”に属している小分類の 1 つであり, 中分類 C00 から D48 まで大分類”新生物”に属している．また入院期間中の病名は 1 つとは限らないので, 同一 c であるが異なる v をもつ c:v が存在する可能性もある．また pij が手術であれば，c:v は術式，手術部位，執刀医などであり，執刀医のカテゴリは所属する診療科などで階層化されている．また従来の OLAP システムであれば，c と v を区別することなく用いるが，提案手法では c を検体検査における白血球数の検査項目，v をその検査値のように区別して扱う．また c はカテゴリ階層の最下位の節点である必要はなく，内部節点でもよいものとする．

階層の集合 $D=\{T_k\}$が与えられたとき, スキーマは $S=(F,D)$と定義される．ここで，F はファクトタイプ，C_l はカテゴリタイプ，T_k は階層タイプ $T_k=(C_l,\leq_{T_k})$である．タイプ T_k の階層インスタンス T_k は，$T_k=(C_k,\leq_{T_k})$である．ここで，C_k はカテゴリ c_j の集合，$\leq_{T_k}$ は C_k 間の半順序関係である. 本提案手法で用いる階層は従来の OLAP のシステムの多くが採用するバランス木である必要はなく，有向非循環グラフを想定している．各カテゴリ $c \in C$ は定義域 dom(c)を持ち，前述のように dom(c)の各要素は，{c:v}と表される．

集計演算の計算速度をあげるために，階層を以下のように索引付けする．人工のルートノード c_{root} を考え, C のうち上位の概念を持たない c_j の親節点とする. c_{root} からはじめて，各節点に前順，後順，深さを割り当てながら深さ優先に探索していく．ただし，内部節点ではバックトラックせずに，葉節点でのみバックトラックする．入力であるカテゴリ c とデータのカテゴリが子孫関係にあるかを判定するために，以下の条件で容易に判定することが可能である. 節点 A が節点 B の祖先であるなら，以下が成り立つ.

A の preorder1(=A の前順)<B の前順
　≤A の preorder2=A の後順+A の深さ

階層関係と区間情報をストアするために，テーブル CATEGORY T と INTERVAL I を

以下のように定義する．

CATEGORY (CATENAME CHARCTER,
PATH CHARCTER,
PREORDER1 INTEGER,
PREORDER2 INTEGER,
PARENT INTEGER)
INTERVAL (ID INTEGER,
START TIMESTAMP,
END TIMESTAMP,
PREORDER INTEGER,
VALUE CHARCTER,
INTERVALID INTEGER)

T の各レコードは階層の各節点に相当し，CATENAME，PATH，PREORDER1，PREORDER2，PARENT は，それぞれ節点のカテゴリ名，ルート節点からのパス，前順，後順と深さの和，親節点の前順である．I の各レコードは，(ts,te,c:v)を|{c:v}|個に分割した情報に相当し，ID，START，END，PREORDER，VALUE，INTERVALID は，それぞれ患者 ID，区間の開始時間，区間の終了時間，カテゴリ c の前順，dom(c)内の値 v，区間の識別子である．区間識別子 INTERVALID を用いる理由は，(ts,te,{c:v})を|{c:v}|個に分割したためである．

以上の 2 つのテーブルを用いて，集計演算を以下のように定義する．以下の定義において，T_c は入力カテゴリに対して，テーブル T の 1 タプルを返す SQL 文である“$\sigma_{P(T)}$ FETCH FIRST 1 ROWS ONLY”を意味する．

集計演算 α：テーブル A($v_1,v_2,\ldots,v_n$, count(distinct id))に対して，$\alpha=\chi_{v_1,v_2,\ldots,v_n,\mathrm{count(distinct\ id)}}$を返す集計演算をα(A)と定義する．ただし，$a\chi_{a,\mathrm{sum(b)}}$ は SQL”SELECT a, SUM(b) FROM ... GROUP BY a"に相当する演算とする．演算αはテーブル A から，n 次元の多次元キューブを生成する関数である．

結合演算 β：結合演算 β を $\beta(I'_1,I'_2,\ldots,I'_n\},O,W)=\pi_O(I'_1\diamondsuit I'_2\diamondsuit\ldots\diamondsuit I'_n)$と定義する．ここで，各テーブル I'_i は区間 I'(id,start,end,value,interval_id)とする．また W は結合の条件式の集合であり，$I'_i\diamondsuit I'_j$ は，W の条件式，及び I'_i.id=I'_j.id に従って結合される．O は出力されるカラムの集合であるとする．

区間選択演算 g:区間選択演算 $g(T,I_c)$を区間を表すテーブル I'(id,start,end,value, interval_id)を返す演算と定義する．関数 g は分析の意図に応じて定義されるユーザ定義関数であり，例えば図 1 に示される．$g^{(1)}$ は指定したカテゴリ c とその子孫カテゴリに属する v を有する区間を選択する演算である．$g^{(2)}$は指定したカテゴリ c に属する v を有する区間を選択する演算である．$g^{(3)}$は $g^{(1)}$と同様の区間を選択する演算であるが，v をテーブル T の CATENAME に置き換えて区間を選択する演算である．$g^{(4)}$もまた $g^{(1)}$と同様の区間を選択する演算であるが，指定されたカテゴリ c の子カテゴリの CATENAME に置き換えて区間を選択する演算である．$g^{(5)}$は $g^{(1)}$と同様の区間を選択する演算であるが，v を区間の開始時間に置き換えて区間を選択する演算である．

$g^{(1)}(T,I,c)=\pi_{\mathrm{id,start,end,value,interval\_id}}(I\diamondsuit_{\mathrm{preorder1}\le\mathrm{preorder}\le\mathrm{preorder2}}T_c)$
$g^{(2)}(T,I,c)=\pi_{\mathrm{id,start,end,value,interval\_id}}(I\diamondsuit_{\mathrm{preorder1=preorder}}T_c)$
$g^{(3)}(T,I,c)=\pi_{\mathrm{id,start,end,T.catename,interval\_id}}(I\diamondsuit_{\mathrm{preorder1}\le\mathrm{preorder}\le\mathrm{preorder2}}T_c)$
$g^{(4)}(T,I,c)=\pi_{\mathrm{id,start,end,T.catename,interval\_id}}(T_c\diamondsuit_{\mathrm{T.parent=T\_c.preorder1}}T\diamondsuit_{\mathrm{T.preorder1}\le\mathrm{I.preorder}\le\mathrm{T.preorder2}}I)$
$g^{(5)}(T,I,c)=\pi_{\mathrm{id,start,end,start,interval\_id}\}}(I\diamondsuit_{\mathrm{preorder1}\le\mathrm{preorder}\le\mathrm{preorder2}\}}T_c)$
$g^{(6)}(T,I,c)=\pi_{\mathrm{id,start,end,end,interval\_id}\}}(I\diamondsuit_{\mathrm{preorder1}\le\mathrm{preorder}\le\mathrm{preorder2}\}}T_c)$

図 1: ユーザ定義関数 g の例

これらの基本演算を組み合わせることで実現される多次元データベースを HealthCube と呼ぶ．HealthCube により主に処理されるクエリは α(β({g(T,I_i,c1), g(T,I_i,c2),…}, O,W),distinct id))で表される．

上記の処理手順を複数の計算機を使って応答時間を短くすることを考える．このためには効率よく各計算機にデータを分散させないといけない．そこで，HealthCube の処理手順に沿って，各データ操作演算におけるデータ分散に必要な要件を述べる．まず，区間選択演算 g では，各計算機になるべく均一にデータを分散させる必要がある．もし過度の偏りがあれば，データ量が多く保存されている計算機が区間選択をする際に負荷がかかってしまい，効率的でない．次に結合演算では，同一 ID のタプルの結合処理が行われるので，同一 ID のタプルが同一計算機に保存されている必要がある．もし，異なる計算機に同一 ID のタプルが分散されていた場合，タプルを結合するために計算機間の通信が必要になり，非効率である．

この条件を満たすために，データ ID をランダムに割り当てる．さらに，k 台の計算機に分散させる場合，テーブル INTERVAL I のタプルを，以下の条件を満たすように複数のテーブル I_i (i=1,⋯,k)に分割し，各計算機に割り当てる．

・$I=\cup_{i=1,\cdots,k}I_i$
・$\mathrm{id}(I_i)\cap\mathrm{id}(I_j)=\varnothing$ $(i\neq j)$

ここで，id(I)は，テーブル INTERVAL I の ID の集合を返す関数とする．1 つ目の条件は，テーブル INTERVAL I の情報が分割により失われないことを表している．また 2 つ目の条件により，同一 ID のタプルが異なる計算機に保存されていないので，上記の要件を満たす．例えば，mod(ID,k)の値に応じて ID を分散させることで条件を満たす．

HealthCube で計算されるクエリは並列化

によって次式になる．

$\chi(\cup_i \alpha(\beta$
$(\{g(T,I_i,c1),g(T,I_i,c2),\ldots\},O,W),\text{distinct id}))$

ここで，I_i はテーブル INTERVAL I を各計算機に分散させたときのテーブルである．また，χ は，各テーブル I_i に対して得られた結果を，属性値が同じ集計結果を足し合わせ，最終結果を導く演算である．上記の式の $\alpha(\beta(\{g(T,I_i,c1), \ldots\},O,W),\text{distinct id})$ を q_i とすると，q_i は i 台目の計算機で処理されるので，テーブル I_i と I_j $(i \neq j)$ を結合する処理は発生しない．すなわち，計算機間の通信は発生しない．χ の演算により，計算機間の通信が発生するが，テーブル I_i に比べ q_i は小さいので，通信負荷が小さくて済む．

４．研究成果

本研究成果を示すために，計算機を用いた実験結果を示す．本実験では，適用するデータを人工的に作成し，テーブル INTERVAL I，CATEGORY T についてそれぞれランダムに生成した．その結果，テーブル INTERVAL I は 7,331,455 タプルで，ID の数が 27,207 個であった．テーブル CATEGORY T は 3,098 タプルであった．つまり，今回作成した人工データは患者数 27,207 人で，カテゴリ数 3098 個を付与された診療履歴データ数が 7,331,455 個存在するデータと解釈することが出来る．

実験するに当たり，HealthCube の処理の違いからクエリを以下の式のように 4 つのタイプに分けた．

・type 11:
$\alpha(\beta(\{g^{(1)}(T,I,c1),O,W)\},\text{count(distinct id)})$

・type 21:
$\alpha(\beta(\{g^{(1)}(T,I,c1),O,W)\},\text{count(distinct id)})$

・type 12:
$\alpha(\beta(\{g^{(1)}(T,I,c1),\ g^{(1)}(T,I,c1)\},\ O,\ W),$
$\text{count(distinct id)})$

・type 22:
$\alpha(\beta(\{g^{(4)}(T,I,c1),\ g^{(1)}(T,I,c1)\},\ O,\ W),$
$\text{count(distinct id)})$

4 つのタイプは大きく 2 つに分けることができ，type 11，type 21 はテーブル INTERVAL I から一区間のみの選択，type 12，type 21 は二区間の選択を行い，より複雑なクエリである．type 11，type 21 の違いは type 21 が区間選択演算 $g^{(4)}$ を用いることにより，type 11 では必要のなかった選択した区間とテーブル CATEGORY T との結合が必要な点である．type 12，type 22 の違いも同様に，選択した区間とテーブル CATEGORY T との結合が type 22 において必要な点である．以下に，それぞれのタイプにおいて考えられるクエリを 1 つずつ挙げる．

・type 11:手術カテゴリにおいて，術式ごとの集計人数は何人か．

・type 21:手術カテゴリにおいて，診療科ごとの集計人数は何人か．

・type 12:手術カテゴリ，検査カテゴリ両方に属する患者で，術式，検査内容ごとの集計人数は何人か．

・type 22:手術カテゴリ，検査カテゴリ両方に属する患者で，診療科，検査内容ごとの集計人数は何人か．

type 11 のクエリは区間選択演算 $g^{(1)}$ により，テーブル INTERVAL I から手術カテゴリの子カテゴリの値を選択し，その値ごとに集計している．type 21 のクエリは区間選択演算 $g^{(4)}$ により，テーブル INTERVAL I から手術カテゴリの子カテゴリの値を，属するカテゴリ名に置き換えたものを選択し，そのカテゴリ名ごとに集計している．子カテゴリの値を，属するカテゴリ名に置き換える操作を行うときに，選択した区間と CATEGORY T との結合が必要になる．type 12 のクエリは手術カテゴリ，検査カテゴリそれぞれの子カテゴリの値を選択し，同一の ID で結合する．その後，それぞれの値ごとに集計する．type 22 のクエリは手術カテゴリの子カテゴリの値を属するカテゴリ名に置き換えたものと検査カテゴリの子カテゴリの値を選択し，同一の ID で結合する．その後，それぞれの値ごとに集計する．

本研究では，提案した手法を Java で実装した．各計算機で並列に処理を行う過程は，Java のマルチスレッドプログラミングを用いて実装した．マルチスレッドプログラミングとは，プロセス内の実行の流れであるスレッドを複数生成し，非常に短時間でスレッドを切り替えることによってほぼ並列的に処理を行うことの出来るプログラム方法である．また，集計した結果を同期化させる手法は，1 つの n 次元配列の更新作業を行うことで実現した． また，他のタイプのクエリを実行するときは，クエリ q の値を変更することで並列処理をすることが出来る．

本実験では，上記のスペックの計算機 2 台を用いて並列化している．計算機の数を k 台としたとき，ID を mod(ID, k)の値に応じて分散させた．具体的には対象データの ID を奇数と偶数に分け，それぞれの計算機に分散した．その結果，ID が奇数のデータには 3,616,174 タプルが，偶数のデータには 3,715,281 タプルが保存されている．またテーブル INTERVAL I の各属性に対し，索引を作成した．テーブル，及び索引を保持するために要したハードディスク上のスペースは，データを分散させなかった場合の計算機，ID が奇数のデータを保持した場合の計算機，ID が偶数のデータを場合の計算機で，それぞれ 1,148.82MB，535.93MB，538.04MB であった．

本実験では，前述のデータの前処理を行った計算機を用い，それぞれのタイプのクエリに対して，HealthCube を並列化している時と並列化していない時での，応答時間を比較した．具体的には，それぞれのタイプのクエリの c1 を全てのカテゴリについて応答時間を計測する．なお，type 12，type 22 のもう一

方のカテゴリは固定させる．
また，応答時間の計測を，集計演算を行う前の時間(time1)と，集計演算を行い，同期化もさせた最終的な応答時間(time2)の 2 つのタイミングに分けた．本実験では集計演算は count(distinct id)に従っており，RDB において count(distinct id)は計算コストが大きいと予想される.その為，集計演算を実行する前までの処理と全体の処理で，それぞれ並列化の効果を把握するために 2 つのタイミングに分けた．
type 11 のクエリを並列化していない場合と，並列化した場合に用いた．図 2，図 3 はそれぞれ，time1，time2 について応答時間を計測した散布図である．このグラフの横軸の対象件数とは，区間選択演算 g により選択される件数である．縦軸はクエリを発行してから結果が返ってくるまでの応答時間である．また，図 4 は提案手法が time1，time2 それぞれについて，従来の HealthCube よりもどれほど早く応答するかを示した図である．縦軸は従来の応答時間=並列化した応答時間の値で，横軸は図 2，図 3 同様，区間選択演算 g により得られる件数である．

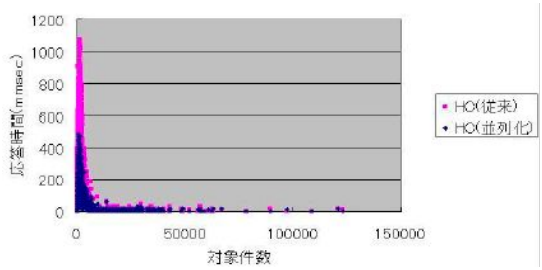

図 2: time1 の結果(type11)

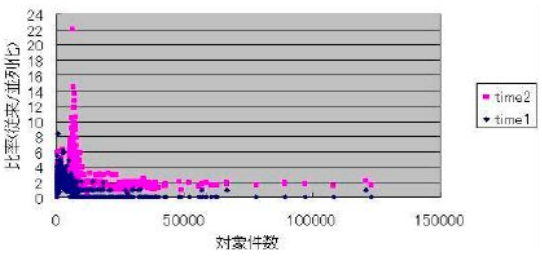

図 3: time2 の結果(type11)

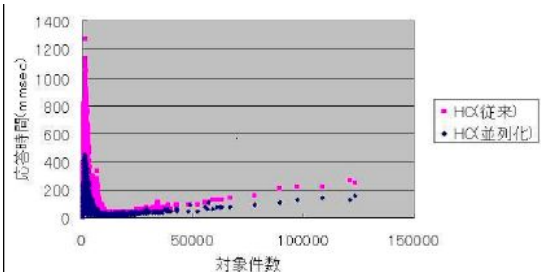

図 4: 応答時間の比(type11)

図 2 から，集計演算を行う前の応答時間である time1 は，対象件数が少ないときに大きな差が見られ，多くなるにつれて差が見られなかった．また，time1 は図 4 から，並列化処理をした方が遅い場合も多くあった．しかし，この場合のクエリは図 2 から，1 つのクエリに対する応答時間がどちらも 0.2 秒に満たないものがほとんどであったので，誤差と考えることが出来る．一方，time2 は図 3 から，time1 の時と同様の結果が得られたが，図 4 からほとんどのクエリについて，2 倍以上早く結果を得ることが出来た．このことから，並列化によって対象件数が増えても最低 2 倍以上早く結果を得ることが出来ると考えられる．type 12, type 21, type 22 に対する実験についても，type 11 と同様の結果が得られた．
これらの実験結果に基づいて，HealthCube と呼ばれるアプリケーションソフトウェアを作成した. HealthCube は, GUI を完備し，SQL に不慣れなユーザであっても, そのインターフェースを介してクエリを作成でき，集計結果を得ることができる．このソフトウェアは経済産業省プロジェクト情報大航海における日本航空インターナショナル社の新総合安全運航支援システムで活用された．また，国内大手の広告代理店とライセンス契約を結んだ．

５．主な発表論文等
（研究代表者、研究分担者及び連携研究者には下線）

〔雑誌論文〕（計 4 件）

① 猪口 明博: 頻出パターンマイニングのグラフ系列への適用，人工知能学会誌, 特集「系列パターンマイニングの最近の動向」, Vol. 27, No. 2, pp 120-127, 2012. 査読有.
http://www.ai-gakkai.or.jp/vol27_no2/

② Akihiro Inokuchi, Hiroaki Ikuta, and Takashi Washio: Efficient Graph Sequence Mining using Reverse Search, IEICE Transactions, Vol.95-D, No.7, pp.1947-1958, 2012. 査読有. DOI:10.1587/transinf.E95.D.1947

③ Akihiro Inokuchi and Takashi Washio: FRISSMiner: Mining Frequent Graph Sequence Patterns Induced by Vertices, IEICE Transactions, Vol.E95-D, No.6, pp.1590-1602, 2012. 査読有. DOI: 10.1587/transinf.E95.D.1590

④ Akihiro Inokuchi and Takashi Washio: GTRACE: Mining Frequent Subsequences from Graph Sequences, IEICE Transactions, Vol.93-D, No.10,

pp.2792-2804, 2010. 査読有. DOI: 10.1587/transinf.E93.D.2792

〔学会発表〕（計 15 件）

① 前田光貴，猪口明博: 動的オラクルを用いた日本語の係り受け解析, 第 76 回情報処理学会全国大会, 2014 年 3 月 11 日, 東京電機大学
② 小西哲生，猪口明博: スペクトラルクラスタリングに基づいた動的に変化するグラフのクラスタリング, 第 76 回情報処理学会全国大会, 2014 年 3 月 11 日, 東京電機大学
③ Akihiro Inokuchi, and Ayumu Yamaoka, Mining Rules for Rewriting States in a Transition-based Dependency Parser for English. The 24th International Conference on Computational Linguistics, 2012 年 12 月 13 日, Mumbai, India
④ Akihiro Inokuchi, Ayumu Yamaoka, Takashi Washio, Yuji Matsumoto, Masayuki Asahara, Masakazu Iwatate, and Hideto Kazawa: Mining Rules for Rewriting States in a Transition-based Dependency Parser, The 12th Pacific Rim International Conference on Artificial Intelligence, 2012 年 9 月 5 日, Kuching, Malaysia
⑤ Nguyen Duy Vinh, Akihiro Inokuchi, and Takashi Washio: Graph Classification based on Optimizing Graph Spectra, The International Conference on Discovery Science, 2010 年 9 月 7 日, Canberra, Australia
⑥ Akihiro Inokuchi and Takashi Washio: GTRACE2: Improving Performance Using Labeled Union Graphs, The 14th Pacific-Asia Conference on Knowledge Discovery and Data Mining, 2010 年 6 月 23 日, Hyderabad, India
⑦ Akihiro Inokuchi and Takashi Washio: Mining Frequent Graph Sequence Patterns Induced by Vertices, The Tenth SIAM International Conference on Data Mining, 2010 年 4 月 29 日, Columbus, Ohio, USA

〔図書〕（計 0 件）

〔産業財産権〕
○出願状況（計 0 件）

○取得状況（計 1 件）
名称：多次元データ分析方法，多次元データ分析装置，及びプログラム
発明者：猪口 明博，高林 健登，鷲尾 隆
権利者：国立大学法人 大阪大学
種類：特許
番号：特許第 5252388 号
取得年月日：平成 25 年 4 月 26 日
国内外の別： 国内

〔その他〕
ホームページ等
http://ist.ksc.kwansei.ac.jp/~inokuchi/index_ja.html

６．研究組織
(1)研究代表者
猪口 明博（INOKUCHI AKIHIRO）
関西学院大学・理工学部・准教授
研究者番号：70452456