Web構造分析を目的とした多次元データマイニング機構の効率化

栗原　大輔† 　大森　匡† 　星　守†

† 電気通信大学大学院情報システム学研究科　〒 182–8585 東京都調布市調布ヶ丘 1–5–1
E-mail: †{kurihara,omori}@hol.is.uec.ac.jp

あらまし　本研究室では，05 年から多次元データマイニングの考え方で，Web 空間構造分析を行ってきた [1], [2]．本稿では，Web 空間構造分析に対して，多次元制約下でデータマイニングを行う機構「アイテムセットキューブ」による効率的な計算方法を課題として，この Web 空間構造分析システムの評価を行うことが目的である．具体的には,「対象とする Web 空間を多次元制約下でどの程度の細かさで捉え，実体化しておくべきか」,「問い合わせとなる，多様な多次元制約の組合せに，ロールアップなどのデータキューブ演算でどう対応するか」を評価する．その上で，多次元データマイニング機構の効率化についての検討を述べる．

キーワード　Web マイニング，データウェアハウス，OLAP，DB アーキテクチャ

A Report on Implementation of a Multi-dimensional Web-Mining System

Daisuke KURIHARA†, Tadashi OHMORI†, and Mamoru HOSHI†

† The University of Electro-Communications, Graduate School of Information Systems, Chofugaoka 1–5–1, Chofu, Tokyo,182–8585 Japan
E-mail: †{kurihara,omori}@hol.is.uec.ac.jp

Abstract　In today's web researches, personalization of web structure mininig is one of hot topics. For this objective, the authors previously proposed a multi-dimensional mining system based on a new data cube model, termed the itemset cube. This paper reports an efficient implementation of the system. We firstly check how much degree of details the itemset cubes should be materialized with. We then propose two efficient algorithms for incrementally answering ad-hoc multi-dimensional queries in the itemset cube system.

Key words　Web mining, Data warehouse, OLAP, DB architecture

1. はじめに

近年 Web 上での仮想組織の活動が活発になっており，Web 空間でどのような活動が行われているかを調べることが重要となってきている [3]〜[6]．また，個人や状況に応じて情報をパーソナライズして調べることが重要である [8].

一方，本研究室では，データキューブの考えに基づいて，多次元制約の下でデータマイニングを行う機構であるアイテムセットキューブの試作を行っている [7]．アイテムセットキューブとは，各属性に指定されたセルに，条件を満たす高頻度アイテムセットを格納し，ロールアップやスライス等のデータキューブ演算を行うことで，アイテムセットの分析を効率良く行う機構である．

05 年からこの多次元データマイニング機構の考え方で Web 空間構造分析を行ってきた [1], [2]．それは，Web 空間構造分析でいうコア (完全 2 部グラフ) を高頻度アイテムセットとして求める (実体化と呼ぶ) と考え，事前に用意した多次元制約下でコアを求めた．そして，コアを基準としたコミュニティ間の関連性を表すグラフモデルを作成し，コミュニティをノードとしたランキングを行った．

図 1 は，本研究室の web 空間構造分析システムを使って，コミュニティのランキングを行うまでの流れである．

本稿では，Web 空間構造分析に対して，アイテムセットキューブによる効率的な計算方法を課題として，この Web 空間構造分析システムの評価を行うことを目的とした．具体的には，

- 「Web データの複雑さから決まる計算コストにどう対応するか」
- 「対象とする Web 空間を多次元制約下でどの程度の細かさで捉え，実体化しておくべきか」
- 「多様な多次元制約の組合せに，ロールアップなどのデータキューブ演算でどう対応するか」

を評価する．

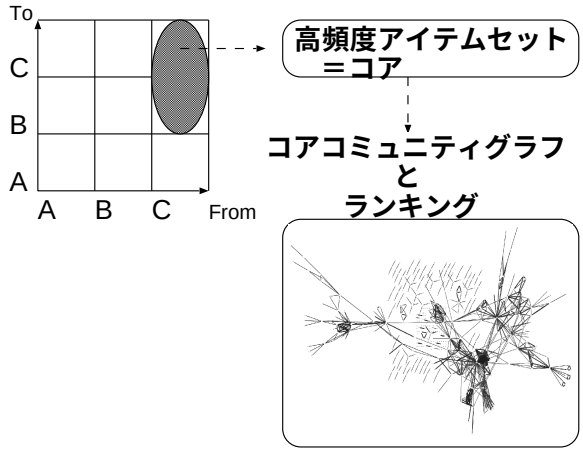

図 1　多次元データマイニングによる Web 空間構造分析

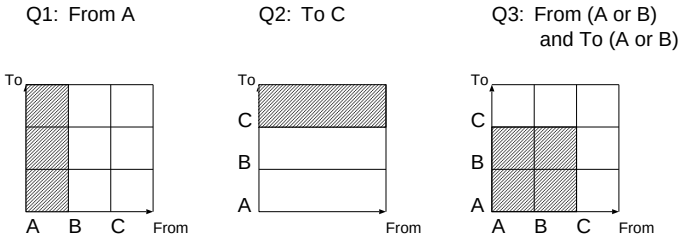

図 2　問い合わせ Q1, Q2, Q3

2. 関 連 研 究

これまでに様々な Web 構造解析の研究が行われてきている．その一つとして，あるトピックに関心を持ったページ集合をコミュニティとし，それを導出することで Web 空間の理解に役立てようとするコミュニティ研究 [3] がある．この Web コミュニティにおいて中心となるページ集合をコアと呼ぶ. [3] では，Web 空間におけるページとハイパーリンクをノードとエッジとし，この Web 空間全体を巨大なグラフと見たときに現れる完全 2 部グラフがコミュニティのコアであるとした．このようなコミュニティの研究としては，コミュニティに対してラベル付けをし，コミュニティの検索を行うもの [4] や，コミュニティ間の関連性を導出し，関連のあるコミュニティ同士を結ぶことでコミュニティ間の関係性を表す地図を作成するウェブコミュニティチャート [5] などがある．また，イントラネットにおける構造解析の研究としては [6] などがある．

3. 多次元制約機構

3.1 アイテムセットキューブ

本研究室では，いつ，どこで，誰がといった多次元制約の下で起きている事象の組であるアイテムセットを一定の閾値以上で高頻度アイテムセットとして求め，データキューブモデルのセルに格納したアイテムセットキューブシステムを提案している [7]．これを操作することで，多次元分析条件に応じた事象の組を効率良く調べることができる．

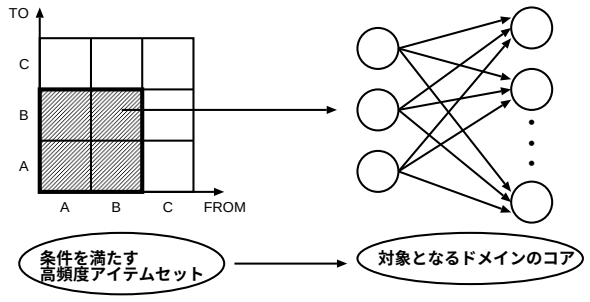

図 3　高頻度アイテムセット計算によるコア抽出

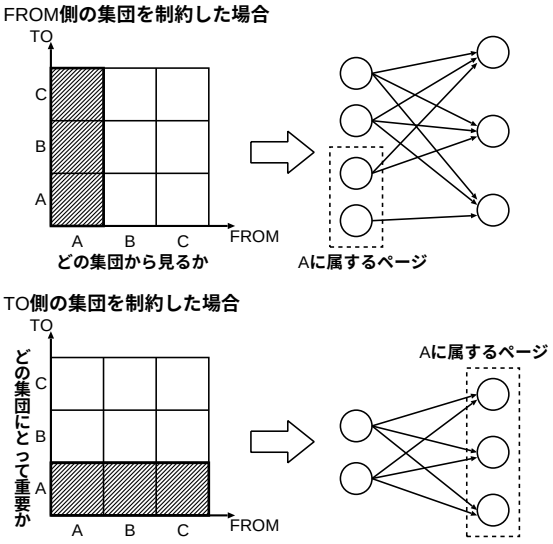

図 4　多次元制約のコアへの影響

3.2 Web 構造マイニングへの適用

現在 Web 構造マイニングの分野では，完全 2 部グラフをコアと呼び，コアに基づいた Web コミュニティ解析の研究が行われている．完全 2 部グラフ計算を，高頻度アイテムセット計算と対応づけることで，一定以上の大きさの完全 2 部グラフを高頻度アイテムセットとして求めることができる．そこで我々は，リンク構造データに対しアイテムセットキューブシステムを用いて，多次元データマイニングによる Web 構造計算を行った [1]．イントラネット型 Web 空間は階層構造を成している．例えば今回分析の対象とした UEC ドメインでは，トップである uec.ac.jp の下に，情報分野のドメイン（情報システム学研究科や情報通信工学科など）や，電気系のドメイン，事務室などのドメインがあり，さらにその下に各研究室ドメインなどが存在するという形になっている．このページの参照関係をもとに図 3，4 に示すように，どのドメインのページ集合からリンクが張られているかに着目した FROM 制約，どのドメインのページ集合に対しリンクを張っているかに着目した TO 制約を行うことで，どのドメインから見るか，どのドメインにとって重要かといった視点を用いて Web コミュニティ構造解析を行っている．

3.3 Web 構造マイニングにおける課題

元々，アイテムセットキューブシステムは Apriori に代表される取り引きログマイニングのための計算機構である．これに

対し，Web 構造マイニングでは，高頻度アイテムセットを Web グラフのリンクレコードから求められる完全 2 部グラフと対応づけることでアイテムセットキューブシステムを使っていた．そのため，既存のアイテムセットキューブシステムでは，データキューブモデル特有のロールアップ演算操作によるコア計算が対応できていないという欠点がある．

例えば，今まで行ってきた Web 空間構造分析では，FROM/TO 制約に関する問い合わせの度，実体化を行い応答してきた．

そこで，Web 空間構造分析におけるアイテムセットキューブシステムによる効率的な計算方法を課題とし，効率的な計算機構の再設計を行うこととした．具体的には，多次元制約機構を実現するにあたって，

1. 「Web データの複雑さから決まる計算コストにどう対応するか」
2. 「対象とする Web 空間を多次元制約下でどの程度の細かさで捉え，実体化しておくべきか」

の 2 点をまず評価する．ここでは，計算能力の改善を目的とした Prune 処理を導入し，実体化の範囲の確定について方針を立てる．次に，その結果を受けて，

3. 「多様な多次元制約の組合せが問い合わせ Q として与えられたとき，既に実体化されているアイテムセットキューブを使って，ロールアップ演算に対応する処理を使う，などのいくつかの方法により，効率的に処理できるか

を検討する．

本稿では，特に，項目 3 について，既に実体化されたアイテムセットキューブから図 2 のような問い合わせ (Q1〜Q3) に応じた結果 (極大コア集合) を生成する方法として，フィルタリングによる方法と，マージによる方法，の 2 つを提案する．

4. Prune 処理を導入した実体化のコスト

まず，実体化のコストを調べる．一般に，始点数 h，終点数 a の完全 2 部グラフをコアと呼ぶ．Ravi Kumar らは Pruning 法で枝切りした後にコアの列挙をする方法を使っている [3]．我々のコア計算は Apriori をそのまま使うが，今回は Pruning 法の枝切りを入れた上で Apriori による計算コストを調べる．

4.1 Prune 処理の目的

現在，求めている最小のコアは始点数 $h = 2$，終点数 $a = 4$ の $(2, 4)$ コアとなっている (図 5)．そこで，この最小コアには，なりえないリンク関係 (辺) を削除することで,「計算能力の改善」と「その改善による実体化の範囲 (パラメータの限界値) の確定」を図る．

4.2 Prune 処理を導入したコア計算の手順

山下，林ら [1], [2] での手順に，Prune 処理を導入すると，次のようになる．

1 クローラで収集した web データをリンク構造レコードにする

2 リンク構造レコードから一部主要ページ (ドリフトを起こすページ) を削除

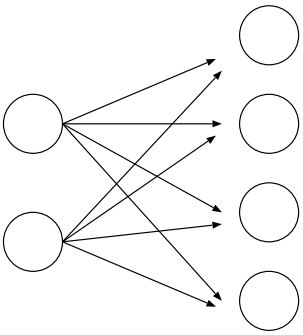

図 5 最小コア ((2, 4) コア)

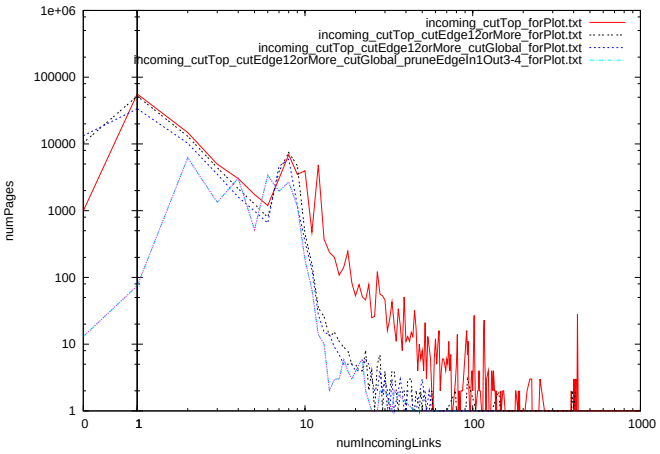

図 6 '05 年の UEC のリンク構造の被リンク数に対するページ数の分布 (上から順に，学科トップ削除，枝刈り $b = 12$，グローバルコア削除，prune 処理後の分布)

3 同様にレコードから，あるページが被リンク数 b 以上の内部リンクのみを持つ場合，その内部リンクを削除．
ただし，このページが外部からリンクされている場合は，内部リンクの削除は行わない．

4 同様にレコードから，最小サポート数 $s = 60$ 以上のコア (グローバルコア) を求め，そのコアの要素となるページを削除

5 prune **処理** $\times n$ **回** (n **はループの回数**)

6 レコードから最小サポート数 s 以上のコアを取り出し，グラフのノードとなるコミュニティを作成する．

4.3 Prune 処理の効果

本稿では，実験に用いるデータ集合を 2005 年の UEC の Web データ (レコード数は 108,221 レコード，リンク数は 534,516 本) としている．

図 6 は，Prune 処理を導入し作成したリンク構造レコードについての被リンク数の分布を表したものである．上から順に,「学科トップ等を削除」の後,「枝刈り数 $b = 12$ で内部リンクを削除」の後,「グローバルコアとなるページを削除」の後,「prune 処理を 4 回かけたもの」である．

表 1 は，各処理を行ったときのリンク数を表している．

この図 6 と表 1 の,「Prune 処理を 4 回後」の分布で，リンク数 0 から 3 辺りについてページ数が大きく変化していることがわかる．これにより，コア計算の負荷を軽減することが出来る．

表 1　'05 年の UEC のリンク構造の被リンク数の分布に対応するレコード数とリンク数

処理方法	レコード数	リンク数
学科 TOP 削除後	108,096	518,558
枝刈り数 b=12 後	90,736	258,245
グローバル要素削除後	63,212	179,761
prune ×4 後	20,746	101,592

表 2　FROM/TO 制約無しでの，Prune 処理によるパラメータの対応表

枝刈り数 b	8	12	20	30	処理なし
最小サポート数 s	4	5	6	25	40
処理時間 (s)	9	75	1831	4	4

4.4　Prune 処理後の計算限界

表 2 は,「枝刈り数 b」,「Prune 処理後のコア計算における最小サポート数 s」と「処理時間」の対応表である．枝刈り数 30 と枝刈り処理なしの 2 つについては，数秒で計算できる最小サポート数 $s(s = 25, 40)$ の設定をしている．

ここで，パラメータ (b, s) の変化による Web マイニング結果の品質について考える．

図 7 は，枝刈り数 b の変化による Web マイニング結果の違いを示す．具体的には，FROM(OTHER) 制約におけるパラメータを変化させた時 (「$b = 8, s = 4$」と「$b = 12, s = 4$」) の比較である．$b = 12$ に変化させたとき，つまり，内部リンクを多く考慮した結果では，いくつかの新規コミュニティ(図 7 右のランク 6, 11, 15 位) が上位に現れることがわかる．この結果からわかるように，UEC の空間などのイントラネット空間の分析には，内部リンクをより多く考慮した分析が有効であると考える．

また，最小サポート数 s については，s が低い程，小さなコアが取り出せるため，より小さい値の分析が有効であると考える．

以上の結果を受けて，FROM/TO 制約無しの実体化の範囲は，Authority 側のノード数を 4 で計算できる,「枝刈り数 $b = 8$，最小サポート数 $s = 4$」が適当であると考える．また，FROM/TO 制約を行った場合については,「枝刈り数 $b = 12$，最小サポート数 $s = 4$」で実体化を行える．これにより，FROM/TO 制約無しの場合と比べ，より詳細な分析が可能となる．

5.　実体化範囲の方針

本稿では，Web 空間構造分析におけるアイテムセットキューブによる効率的な計算方法を課題とし，効率的な計算機構の再設計を行うことである．

そのために，まず，図 8 のように，事前に実体化したアイテムセットキューブを準備する．そして，問い合わせに応じて，準備したアイテムセットキューブを使い，応答することを考える．これにより，効率的な応答をしたい．

例えば，Q = FROM(A or B) を FROM(A) と FROM(B) の各結果から計算したい．この操作は，論理的にはデータキューブのロールアップ演算に対応する．

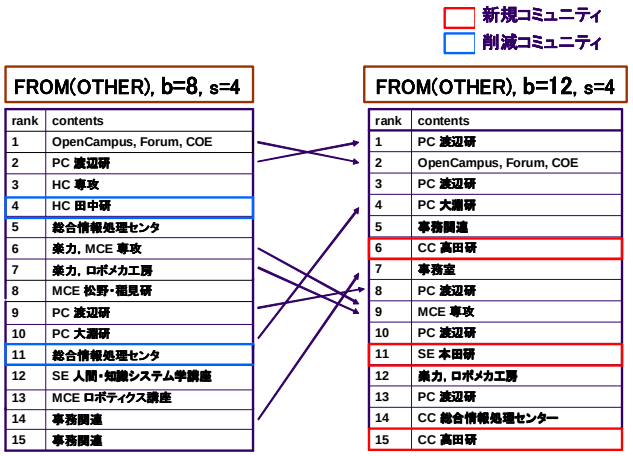

図 7　枝刈り数の変化による Web マイニング結果の違い (FROM(OTHER) 制約での,「$b = 8, s = 4$」と「$b = 12, s = 4$」の比較)

ここで，実体化の範囲 (パラメータの設定) を，Prune 処理による実体化計算の改善を受けて，図 8 のようにキューブを準備した．各キューブの説明は，次のようになる．

D1:　FROM/TO 制約無しでパラメータを「$b = 8$, $s = 4$」として，実体化を行い，極大コアを格納したキューブ

D2:　FROM 制約をドメイン A, B, C それぞれについて，パラメータを「$b = 12$, $s = 4$」として，実体化を行い，極大コアを格納したキューブ

D3:　TO 制約をドメイン A, B, C それぞれについて，パラメータを「$b = 12$, $s = 4$」として，実体化を行い，極大コアを格納したキューブ

これらの事前に実体化したキューブからロールアップ演算をして，FROM/TO 制約に関する問い合わせに対応していく．

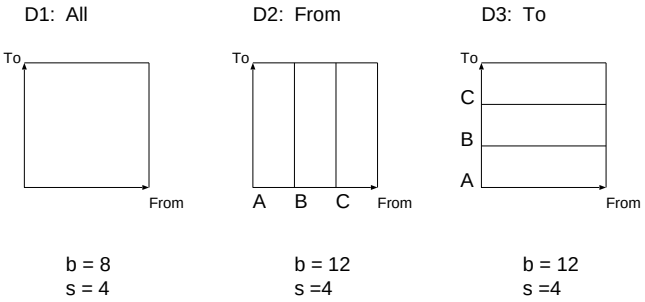

図 8　事前に準備するキューブとそのパラメータ

6.　ロールアップ演算の実現方法

様々な制約の組合せである問い合わせに対するロールアップ演算を実現する方法として，2 つの応答方法「フィルタリング法」と「マージ法」を提案する．

図 8，図 9 から，FROM(A or B) 制約の問い合わせ (図 9 の Q1) に対する,「フィルタリング法」と「マージ法」を使った応答の例を次に示す．

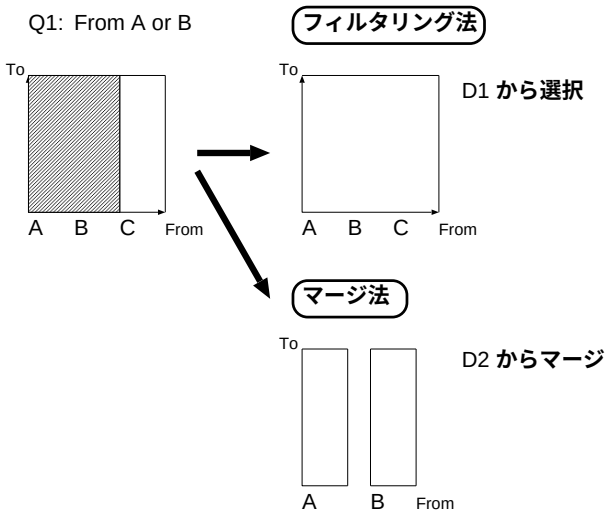

図 9　FROM(A or B) 制約の問い合わせに対する応答

- 「フィルタリング法」:

事前に準備したキューブ (D1) から，制約に対応する極大コアを選択する応答する方法

- 「マージ法」:

事前に準備したキューブ (D2) の「FROM(A) 制約の極大コア集合」と「FROM(B) 制約の極大コア集合」をマージし，一部再実体化を行い FROM(A or B) 制約と同等の極大コアを取り出し応答する方法

また，D1 と D2(または，D3) は，違うパラメータで実体化されているため，この応答方法を用いることで，次の 2 つが実現できる．

- 大雑把に Web の分析を行いたい場合は，D1 からフィルタリング法をすることによって問い合わせに対応する
- 詳細度を高くし，より詳細に分析を行いたい場合は，各制約で事前に実体化した D2(または，D3) を使って，マージ法で問い合わせに対応する

計算時間を考えると，フィルタリング法の方がマージ法に比べ処理が速い．しかし，詳細度で考えると，マージ法の方が個々の制約に対してキューブを実体化しているため，より詳しく分析できる．計算時間等については，次の各処理の説明で評価する．

6.1　FROM 次元上のフィルタリング方法

まず，基本概念を定義する．

1.　リンクレコード:

ノード x について，x を終点とする辺が全部で k 個あるとき，その各辺の始点をノード $i_1, i_2, \ldots, i_k$ とする．このとき，組 $(x, i_1, i_2, \ldots, i_k)$ をリンクレコード (または，レコード) と呼ぶ．

2.　FROM(A) 制約を満たすレコード:

ドメイン A から見たときの制約条件を意味する．形式的には，ドメイン A に属すノードがレコード r の始点側ノードとして存在するとき，r は FROM(A) 制約を満たすと定義する．

3.　極大コア:

完全 2 部グラフ（コア）c_1, c_2 について，c_1 の始点集合が c_2 の始点集合に含まれ，かつ，c_1 の終点集合が c_2 の終点集合に含まれるとき，c_1 は c_2 に含まれると言う．コア c について，自分の他に c を含むコアが存在しないとき，c を極大コアと呼ぶ．

4.　FROM(A) 制約を満たす極大コア:

FROM(A) 制約を満たすレコードのみから計算された極大コア．例えば，図 10 は，コア自体の始点側ノードにドメイン A のページが含まれないが，FROM(A) 制約を満たすような極大コアである．

当然，次の補題が成立する．

[**補題** 1]

FROM(A) 制約を満たす極大コア c について，その終点側ノードは必ず,「FROM(A) 制約を満たすレコード」の終点ノードである．

今，次の 2 つの極大コア集合を考える．

- C_0:

FROM/TO 制約無し，つまり，全てのレコードから計算された極大コアの集合．これは，制約無しの条件で実体化されたアイテムセットキューブに他ならない．このときの詳細度を，枝刈り数 b，最小サポート数 s とする．

- C_1:

C_0 から，FROM(A) 制約フィルタリング処理によって取り出された極大コアの集合．

ここで，FROM(A) 制約フィルタリング処理とは，C_0 への次の処理である．まず，各レコード r について，その終点ノード n に，r が FROM(A) 制約を満たすかどうかを示すフラグ isFromA[n](= 1 or 0) を与えておく．C_1 は，C_0 のコアのうち，終点ノードとして isFromA[n]=1 となるノード n を少なくとも 1 つ持つようなコアの集合とする．

ここで，

- C_2:

C_0 と同じ詳細度パラメータ (b, s) の下で，FROM(A) 制約を満たすレコードのみから計算された極大コアの集合.

を考える．このとき，次の性質が成立する．

[**定理** 1]

C_2 の任意の元 c_2 について，次の性質を満たす C_1 の元 c_1 が唯一つ必ず存在する．すなわち，c_2 の始点ノード集合と c_1 の始点ノード集合が等しく, かつ c_2 の終点ノード集合が c_1 の終点ノード集合に含まれる．(図 10 参照)

定理 1 により，C_0 と同じ詳細度パラメタで良ければ，C_2 を直接計算する必要はなく，C_0 から FROM(A) 制約フィルタ処理によって C_2 を作れば良い．ただし，定理 1 は， c_2 がある c_1 に含まれていることを保証するだけであるから，正確には次の処理を行うことになる．

[FROM(A) 制約フィルタリング処理]
入力： C_0
出力： C_2

1. C_0 のうち，終点ノードとして isFromA[n]=1 となるノードを少なくとも 1 つ持つような極大コア c_x を選ぶ．
2. c_x の終点ノードのうち，isFromA[n]=0 となる終点ノードを削除し，c'_x とする．
3. c'_x の終点ノードの総数が，求める最小サポート数 s より大きければ，c'_x の集合を C_2 とする．

図 10 は，FROM/TO 制約無しの極大コア集合からフィルタリング処理によって，FROM(A) 制約を満たす極大コアを取り出す例である．

次に，FROM(A) 制約の関する問い合わせに対する応答について,「FROM/TO 制約無し，枝刈り数 $b = 8$，最小サポート数 $s = 4$ のパラメータで実体化したキューブからフィルタリング法を適用した応答」と「FROM(A) 制約について同パラメータの再実体化による応答」を比較した．実験に用いたデータ集合は，2005 年の UEC の Web データ (4.3 節における Prune 処理後のリンクレコード) であり，使用計算機環境は，CPU は Intel Pentium 3.20GHz，メモリは 3.5GB である．また，A ドメインは OTHER ドメインを表す(注1)．表 3，表 4 が，比較の結果である．表 3 の処理方法の「フィルタリング」とは，実体化したキューブからフィルタリング法を適用し，極大コアを取り出すまでの処理である.「後処理」は，極大コアからコミュニティノードを作成し，Web 空間構造についてのエッジ付けとランキングまでの処理である．表 4 については,「実体化」がアイテムセットキューブシステムによって Web のリンク構造からコアに対応するアイテムセットを計算する処理であり,「アイテムセットからコアへの変換」と「コアの極大化処理」はアイテムセットから極大コアを取り出す処理までを行っている.「後処理」は，表 3 と同様にコミュニティノードからランキングまでの処理である．

この結果から，フィルタリング法では，再実体化をする必要がないので,「FROM(A) 制約の再実体化」よりも早く応答できることがわかり，有効な手法であると考えられる．

6.2 FROM 次元上のマージ法

FROM 制約のマージ方法とは，図 9 の事前に極大コアの計算を行ったキューブ (D2) を使い，その各制約を組み合わせることで，FROM 制約に関する問い合わせに対応するものである．

例えば，問い合わせ (Q1) を,「FROM(A or B) 制約」としたとき，次の処理が FROM 制約のマージ方法である．

- FROM(A) 制約を満たす極大コアの集合を S
- FROM(B) 制約を満たす極大コアの集合を T

(注1)：IS=情報システム学研究科，EE=電子工学科．C=情報通信工学科，J=情報工学科．OTHER には MCE（知能機械工学科），FEDU（留学生センター）や各種研究所が入る．

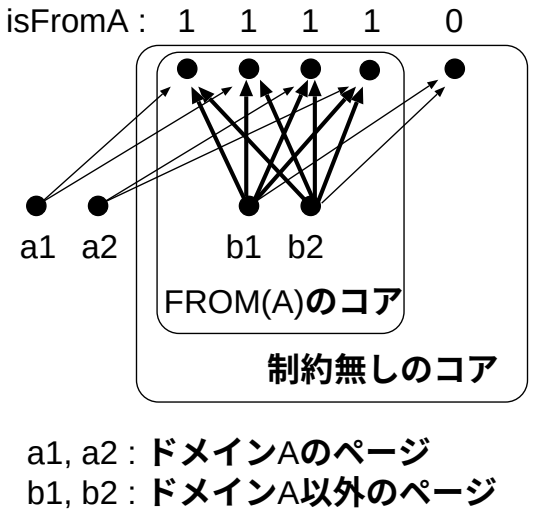

図 10　FROM(A) 制約を満たすコア

表 3　FROM(A) 制約の問い合わせに対するフィルタリング法による応答の実行時間 ($b = 8, s = 4$)

処理方法	実行時間 (s)
フィルタリング	2
後処理	7
合計	9

表 4　FROM(A) 制約の問い合わせに対する再実体化による応答の実行時間 ($b = 8, s = 4$)

処理方法	実行時間 (s)
実体化	4
アイテムセットからコアへの変換	18
コアの極大化処理	9
後処理	5
合計	36

とおく．FROM(A or B) 制約を満たす極大コアの集合を V とおくと V を求める手順は次のようになる．

1. FROM(A or B) 制約を満たすリンクレコードから A または B ドメインに属す始点側ノードをすべて除去し，その結果となるリンクレコードを使って極大コアを計算する．つまり，図 11 における c のようなコアを計算する．このような極大コアの集合を C とおく．
2. C と，すでに計算されている S と T の 3 つの極大コア集合について，重複除去を行う．

以上で，V を求めることが出来る．

ここで，枝刈り数 $b = 12$，最小サポート数 $s = 4$ のパラメータで，A, B ドメインをそれぞれ OTHER, EE ドメイン(注1) としてマージを行った．実験に用いたデータ集合は，2005 年の UEC の Web データ (4.3 節における Prune 処理後のリンクレコード) である．

表 5 は，FROM(A or B) 制約を満たすリンクレコードから A または B ドメインに属す始点側ノードをすべて除去し，残ったリンクレコードを使って極大コアを計算したコストである．

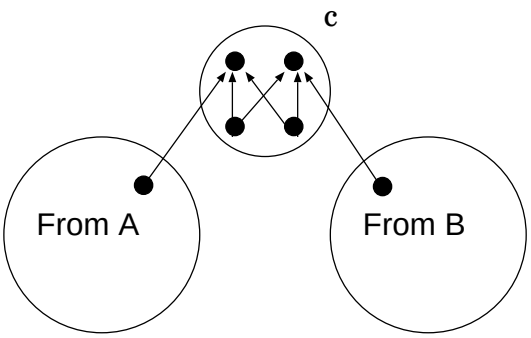

図 11　マージ処理におけるコア

実行時間は 1 秒.

これに対し，表 6 は，FROM(A or B) 制約の問い合わせに対して，マージ方法を使わず，FROM(A or B) 制約について再実体化を行ったものである．実行時間は 79 秒.

この場合，明らかにマージ法を使う方が有効である．

次に，表 7 から,「FROM(A or B) 制約を満たすレコード」と「マージ法において，実体化で用いるレコード」を比較してみると，マージ法で使うレコードの方が構造が小さいことがわかる．

ここで,「FROM(A or B) 制約についての再実体化処理」より「マージ処理」の方がコストがかかる場合があるのかを考える．マージ法では，マージ法の手順 1 にあるように FROM(A or B) 制約を満たすレコードから A または B ドメインに属す始点側ノードをすべて除去し，その後のリンクレコードを使って実体化の計算をするため,「FROM(A or B) 制約を満たすレコードによる実体化処理」より，コストがかからないことがわかる．つまり，いかなる制約においてもマージ法が有効であると考えられる．

表 5　FROM(A or B) 制約の問い合わせに対するマージ法による応答の実行時間 ($b = 12, s = 4$)

処理方法	実行時間 (s)
実体化	1
アイテムセットからコアへの変換	1
コアの極大化処理	1
後処理	244
合計	247

表 6　FROM(A or B) 制約の問い合わせに対する実体化による応答の実行時間 ($b = 12, s = 4$)

処理方法	実行時間 (s)
実体化	79
アイテムセットからコアへの変換	135
コアの極大化処理	62
後処理	244
合計	520

7. おわりに

本稿では，Web 空間構造分析に対して，アイテムセットキューブによる効率的な計算方法を課題として，この Web 空間構造分析システムの評価を行った．まず,

- 「Web データの複雑さから決まる計算コストにどう対応するか」
- 「対象とする Web 空間を多次元制約下でどの程度の細かさで捉え，実体化しておくべきか」

の 2 点については，従来 [1], [2] のコア計算の手順に Prune 処理を導入することで,「計算能力の改善」を行い,「その改善による実体化範囲の設定」を行った.

そして，その結果を受けて，次の

- 「多様な多次元制約の組合せに，ロールアップなどのデータキューブ演算でどう対応するか」

を課題として，ロールアップ演算の実現手法に「フィルタリング法」と「マージ法」を提案し，評価を行った．具体的には，FROM 制約に関する問い合わせに対し,「ロールアップ演算の実現手法」と「問い合わせに対応する FROM 制約を満たす実体化」を比較し，提案した手法の有効性を示した.

表 7　'05 年の UEC のリンク構造，枝刈り数 $b = 12$ における「レコード数」と「リンク数」

処理方法	レコード数	リンク数
FROM(EE or OTHER) 制約	10862	51193
マージ法における C	122	865

文　献

[1] 山下 由展, 大森 匡, 星 守, “多次元データマイニングを用いた Web 空間の構造解析,” 電子情報通信学会 DEWS2006, 3B-o3, 2006.

[2] 林 和宏, 大森 匡, 山下 由展, 星 守, “多次元データマイニングによる Web 空間の構造解析の評価,” DBSJ Letters Vol.6, No.1, 2007.

[3] Ravi Kumar, Prabhakar Raghavan, Sridhar Rajagopalan, Andrew Tomkins, “Trawling the Web for emerging cyber-communities,” WWW8/Computer Networks, Vol.31(11-16), pp.1481–1493, 1999.

[4] Ravi Kumar, Prabhakar Raghavan, Sridhar Rajagopalan, Andrew Tomkins, “Extracting large-scale knowledge bases from the web,” In Proc. of the 25th VLDB Conference, pp.639–650, 1999.

[5] 豊田 正史, 吉田 聡, 喜連川 優, “ウェブコミュニティチャート: 膨大なウェブページを関連する話題を通して閲覧可能にするツール,” 電子情報通信学会論文誌, D-1 Vol. J87-D-1 No.2, pp.256–265, 2004.

[6] 大塚 浩司, 大町 真一郎, 阿曽 弘具, “ウェブコミュニティ内のページ・リンクから成る階層構造の抽出,” 電子情報通信学会, WI2-2006-33, pp.123–128, 2006.

[7] 成瀬 正英, 大森 匡, 星 守, “多次元的なログデータマイニングを実現するデータキューブ機構の提案と評価,” 電子情報通信学会, DEWS2005, 3C-i10, 2005.

[8] Srairam Raghavan, Hector Garcia-Molina, “Complex Queries over Web Repositories,” In Proc. of the 29th VLDB Conference, pp.33–44, 2003.