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

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

栗原 大輔[1]　大森 匡[2]　星 守[3]

Daisuke KURIHARA　Tadashi OHMORI
Mamoru HOSHI

著者らは, '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. 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. 併せて, ロールアップに相当する処理の効率的実行方法を提案する.

The authors previously proposed a multi-dimensional data mining model termed an *itemset cube*, and applied it to the web-strucuture mining problem. This paper reports an efficient implementation of this model. This paper firstly describes how much degree of details itemset cubes should be materialized with. Then two new algorithms are proposed for incrementally evaluating ad-hoc multi-dimensional queries in the problem.

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, 4, 5]. 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.

一方, 本研究室では, データキューブの考えに沿って多次元制約下のデータマイニングを行うデータベースモデルとして,「アイテムセットキューブ」と呼ぶ概念を提案し, 試作している [6][1]. アイテムセットキューブとは, データキューブモデルの最小立方格子 (セル) に, 制約を満たすレコード集合から求まる高頻度アイテムセットを格納したデータベースモデルである. 従来の数値集計用データキューブと同じく, 実体化やロールアップ, スライスなどの演算を持ち, 多様な多次元制約下での高頻度アイテムセットの計算を効率良く行うことを目的としている.

2005 年以降, 著者らは, アイテムセットキューブを用いて, イントラネット型の Web 空間の構造分析 (コミュニティ分析) を行ってきた [1, 2]. 具体的には, Web 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こととし, コミュニティ分析に用いる制約をいくつか与え, その多次元制約空間の中でユーザが与えた個々の制約条件に応じたコアの集合を求めることにした (図 1 参照). 文献 [1, 2] では, 以上の考えに基づいて, 求めたコア集合からコミュニティ間の関係を表すグラフモデルを作り, ランクづけをすることで, 与えられた多次元制約に応じて重要なコミュニティが求まることを示している.

一方で, 上記の研究では, アイテムセットキューブ自体は概念的なモデルでしかなく, 個々の問い合わせの持つ多次元制約に応じてコア集合を再計算していた. これは, 元々, アイテムセットキューブの試作システムが, 多次元制約下のログ分析のために作られたためである. そこで, 本稿では, Web 構造分析において, あらかじめ実体化しておくべきアイテムセットキューブの種類や範囲, を調べた後, ロールアップ演算の効率的な実現アルゴリズムを提案する.

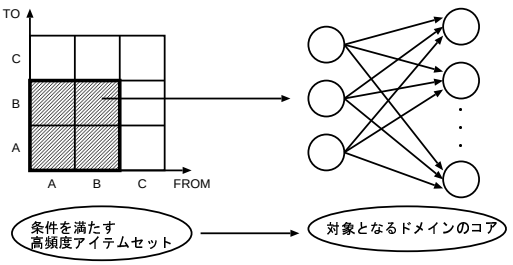

図 1: 高頻度アイテムセット計算によるコア抽出

Fig.1: Core detection by frequent itemsets

2. 多次元制約下の Web 構造マイニング

2.1 アイテムセットキューブ

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ものをアイテムセットキューブと呼ぶ. 著者らは, このデータキューブモデルを文献 [6] で提案し, 実体化やロールアップ処理の効率的実行法を示して, 多次元制約下でのデータマイニングが本モデルにより可能なことを示している.

2.2 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 [3][4]. 一般に, 有向グラフの各ノードとその全入辺の始点集合の組を 1 つのレコードと見なすと, 一定以上の大きさの完全 2 部グラフは当該レコード集合での高頻度アイテムセットとして計算できる [2]. そこで我々は, 電気通信大学 (UEC) 内の Web ページ集合を表す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].

一般に, 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 UEC ドメインでは, トップである `uec.ac.jp` の下に, 情報系ドメイン (情報システム学研究科や情報通信工学科など, IS と表記) や電気系ドメイン (EE と表記), その他の分野系ドメイン (OTHER と表記) 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. そこで, 我々は, 多次元制約として, 図 2 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, 及び, 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, の 2 つを考えた. 例えば, 図 2 の上部は FROM(A) 制約の例であり, これは, 入辺関係を表すリンクレコードのうち, A ドメインに属すページを始点に持つようなレコード集合からコアを

[1]学生会員, 電気通信大学, kurihara@hol.is.uec.ac.jp
[2]正会員, 電気通信大学, omori@hol.is.uec.ac.jp
[3]非会員, 電気通信大学

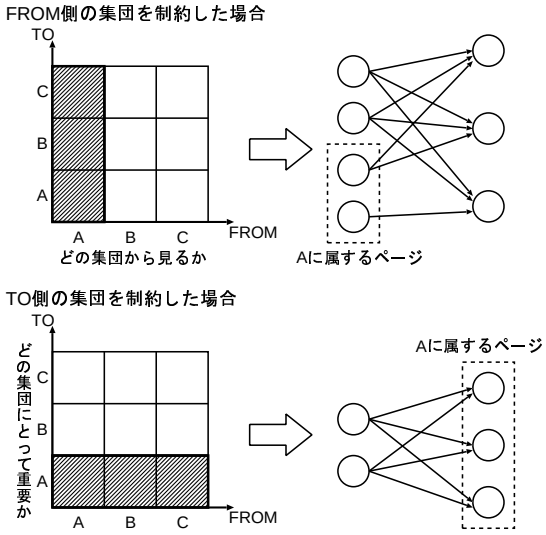

図 2: 多次元制約のコアへの影響
Fig.2: Cores satisfying FROM-/TO- constraints

求めることを意味する. 図 2 の下部は TO(A) 制約であり, これは, A ドメインのページを終点に持つリンクレコードからコアを求めることである.

上記の FROM/TO 制約によって, どのドメインから見て重要か, どのドメインにとって重要かといった多様な視点を反映したコア計算ができる. 我々は, 文献 [2] で, こうしたコア集合から Web コミュニティを表すグラフを作ってランクづけし解析する方法を提案している. この枠組では, 例えば, 図 3 の問い合わせ Q3 は, 対象ドメインが A,B,C の 3 つのサブドメインにわかれている時, A または B ドメインについて FROM 制約と TO 制約両方を満たすコアだけを対象にしたコミュニテイ分析を指示する問い合わせである.

2.3 Web 構造マイニングにおける課題

アイテムセットキューブは元々, 多次元制約下のログマイニング向けに実体化やロールアップの実行算法を与えている [6]. 多次元制約下のコア計算に際しては, アイテムセットキューブの実体化処理は, 制約に応じたレコード集合ごとに Apriori 法でコア計算している. これ自体は問題ないが, 今までの研究 [1][2] では, ロールアップ処理を必要とする問い合わせ (図 3 の各問い合わせ) の実行の場合でも, 該当するレコード集合から必要なコアを Apriori 法で直接再計算していた.

そこで, 本稿では, アイテムセットキューブの考えに沿って, 多次元制約下の効率的なコアの (再) 計算方法を検討する. まず,

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べきか」
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, どのようなアイテムセットキューブを実体化しておくべきか」

の 2 点を評価する. ここでは, 文献 [3] の Prune 処理を導入し, 事前に実体化しておくべきアイテムセットキューブを確定する. 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効率的に実現する方法」

を提案する. 項目 3 については, 図 3 のような問い合わせ (Q1〜Q3) に対応して (極大な) コア集合を効率良く計算するアルゴリズムを提案する.

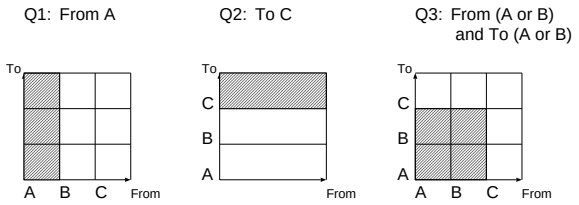

図 3: 問い合わせ Q1, Q2, Q3
Fig.3: Queries having various constraints

3. 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.((h,a)-コア). Kumar らは Pruning 法と呼ぶ枝切り処理の後にコアを列挙する方法を使っている [3]. 我々のコア計算は Apriori 法をそのまま使うが, 今回は Pruning 法の枝切りを入れた上でその計算コストを調べる.

3.1 Prune 処理の目的

本稿で求めている最小のコアは, 始点数 $h = 2$, 終点数 $a = 4$ の $(2, 4)$-コアである. 今回は, この最小コアになりえないリンク関係 (辺) を Prune 法で削除した上で, 対象データ上でコア計算が可能になる範囲 (計算時の各パラメータの限界値) を確定する.

3.2 Prune 処理を導入したコア計算の手順

山下, 林ら [1, 2] の手順に, Prune 処理を導入すると, 次のようになる. ($prune()$ は文献 [3] のループ処理 1 回分で, 入辺数と出辺数を見て不要な辺を削除する).

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
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 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 $prune() \times n$ 回 (n はループの回数)
6 レコードから最小サポート数 s 以上のコアを計算し, コミュニティを表すノードを作成.

なお, 手順 6 では直接 Apriori 法を用いる.

3.3 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) とした. 図 4 は, このリンクレコード集合について, 被リンク数 (x) となるページ数 y を求めて x vs. y 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. 各処

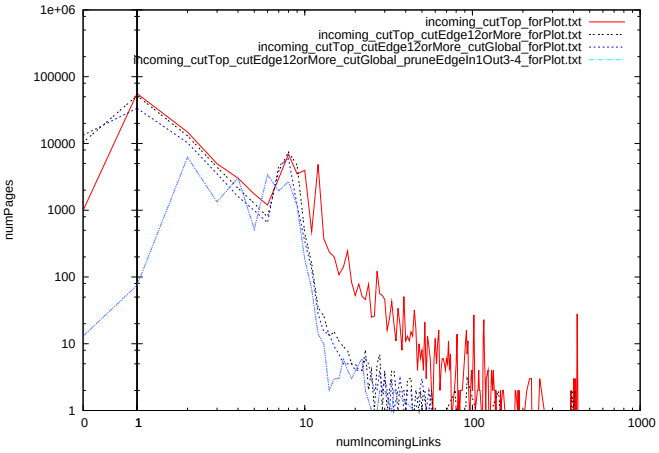

図 4: 被リンク数 x を持つページ数 y の分布 ('05 年の UEC)
Fig.4: Number of pages having x pieces of incoming links

表 1: コア計算の各処理ステップでのリンクレコード集合
Table.1: Size of link records at each filtering step

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

理を行ったときのリンク数の変化を表 1 に示す. Pruning 処理によって披リンク数 0 から 3 付近のページが減っている.

3.4 Prune 処理後の計算限界

3.2 節の手順は, ステップ 3 の枝刈り (下限) 数 b, 及び, Prune 処理後のコア計算における最小サポート数 s, の 2 つがパラメータである. 無制限なコア計算は指数計算であるから, b, s の 2 つを制御する必要がある. 表 2 に, FROM/TO の制約条件のない場合 (つまり，UEC 全体) について, 実質的に計算が可能となるときの b, s の値の組と計算時間を示す. $b = 30$ と $b = \infty$(枝刈り処理なし) の 2 つについては，数秒で終了する範囲しか計算できなかったため, その時の最小サポート数 $s(s = 25, 40)$ を示した. このように, 計算可能な b, s の組には限界がある.

次に, 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調べた. 図 5 は，b の変化による Web マイニング結果の違いを示す．この図は，制約を FROM(OTHER) として, パラメータを変化させた時 (「$b = 8, s = 4$」と「$b = 12, s = 4$」) に, 文献 [2] の方法で求まる上位 15 位のコミュニティの比較である．$b = 12$ 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 5 右のランク 6, 11, 15 位) が上位に現れることがわかる．(図中の矢印は, 主要なコミュニティの対応関係を示す). 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も必要である. つまり, 必要に応じて大きい値の b を使った分析も必要である. また，最小サポート数 s についても, より小さい値の方が小さなコアを取り出せて望ましい.

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

表 2: FROM/TO 制約無しで計算可能な限界となる b, s
Table 2: limits of b, s enabling the materialization

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

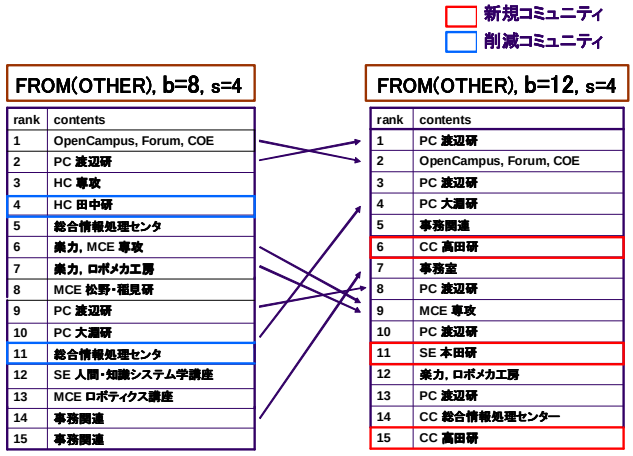

図 5: b の変化による Web マイニング結果の違い
Fig.5: Comparison of top-15 results ($b = 8$ vs. $b = 12$)

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 $s = 4$」で実体化を行えた．これにより，FROM/TO 制約無しの場合と比べ，制約下ではより詳細な分析が可能となる.

4. 実体化範囲の方針

3 節の結果を受けて, アイテムセットキューブモデルによる多次元制約下のコア計算を効率的に実行する体系を考える. 以下, 対象とする Web 空間を A,B,C の 3 つのサブドメインとし, これらからなる FROM/TO の 2 次元制約を考える.

まず，図 6 に示す 3 つのアイテムセットキューブを事前に実体化する．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) なら, これを, FROM(A) と FROM(B) の各結果であるコア集合から効率的に計算したい．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 6 の各キューブは, 実体化時のパラメータ b, s の設定を，3 節の結果に基づいて行っている. すなわち :

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アイテムセットキューブ

D2: 制約を FROM(A), FROM(B), FROM(C) 各々とした場合についてパラメータ $b = 12, s = 4$ の下で実体化を行い，各場合の極大コアを格納したアイテムセットキューブ

D3: 制約を TO(A), TO(B), TO(C) 各々とした場合についてパラメータ $b = 12, s = 4$ の下で実体化を行い，各場合の極大コアを格納したアイテムセットキューブ

である.

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．

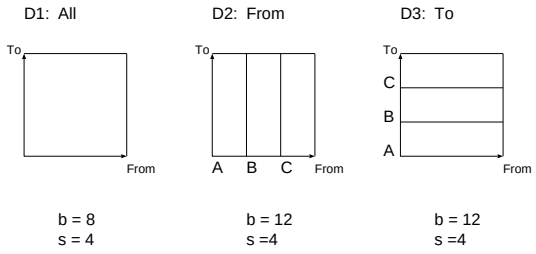

図 6: 事前に準備するキューブとそのパラメータ

Fig.6: Itemset cubes to be prepared

5. 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．

今, 制約 FROM(A or B) を持つ問い合わせが与えられたとき, 図 6 を使って処理する方法は 2 つ考えられる．1 つは, 事前に準備したキューブ (図 6 の D1) から，制約を満たす極大コアを選択して返す方法である．これを「フィルタリング法」と呼ぶ．もう 1 つは, 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，一部足りないものを差分計算して, FROM(A or B) 制約を満たす極大コア集合を返す方法である．これをマージ法を呼ぶ．(図 6 を使って Q1 (=FROM(A or B) 制約) を処理するこれら 2 つの方法を, 図 7 に示す).

ここで, D1 と D2(または，D3) は，違うパラメータ (b, s) で実体化されていることに注意する．このため, 上記の 2 つの方法を次のように用いることにする:

- 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によって問い合わせを処理する．
- 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．以下, 各々の計算方法と処理時間を述べる．

5.1 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．

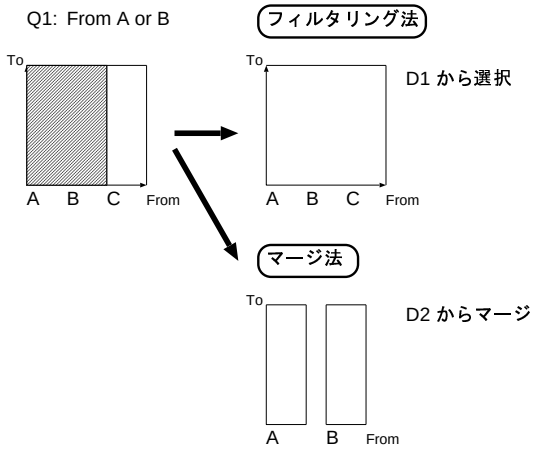

図 7: FROM(A or B) 制約の問い合わせに対する応答

Fig.7: two methods for answering FROM(A or B)

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 8 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， 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」によって選ば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 への次の処理のことである:「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) を予め与えておく．次に, 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 つ持つようなコアを全て選び, 集合 C_1 とする.」

次に,

- 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 8 参照) [4].

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

[4]定理 1 は, 各リンクレコードが自分の終点ノードの持つ全ての始点ノードを含むこと，及び，c_0, c_1, c_2 が全て極大コアであることから自明.

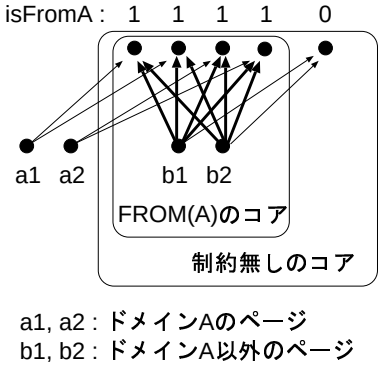

図 8: FROM(A) 制約を満たすコア

Fig.8: Core satisfying FROM(A) constraint

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．全ての c_x について次の 2. と 3. を実行.
2. c_x の終点ノードのうち，isFromA[n]=0 となる終点ノードを全て削除し，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 を C_2 の元とする.

本手法を以下, フィルタリング法と呼ぶ. 図 8 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，制約条件を FROM(A) とした問い合わせの処理時間を,「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て応答する場合」と「同じ b, s パラメータ値の下で，FROM(A) 制約を満たすレコード集合からキューブを直接再実体化して応答する場合」の 2 通りについて比較した．実験に用いたデータ集合は，2005 年の UEC の Web データ (3.4 節における Prune 処理後のリンクレコード) である．(使用マシンは Pentium 3.20GHz，メモリ 3.5GB). また，A ドメインは OTHER(UEC のその他系) ドメインとした.

表 3，表 4 に各場合の処理時間を示す. 表 3 の処理内容の欄の「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 ([2] 参照) である．表 4 については,「実体化」が Apriori 法により 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いえる.

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 ($b = 8, s = 4$)

Table 3: time of the filtering method (FROM(A)))

処理内容	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 ($b = 8, s = 4$)

Table 4: time of the re-materialization (FROM(A))

処理内容	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

5.2 FROM 次元上のマージ法

FROM 制約のマージ方法とは，図 6 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

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 9 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[5].

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 UEC の OTHER(その他系) ドメイン, EE(電気系) ドメインとしてマージ法を行った．実験に用いたデータ集合は，2005 年の UEC の Web データ (3.4 節における Prune 処理後のリンクレコード) である.

表 5 に,「制約条件を FROM(A), FROM(B), FROM(A or B) の各々とした場合にアイテムセットキューブを直接再実体化して応答するまでの計算時間」, 及び,「マージ法により FROM(A or B) 制約下の極大コア集合を求めて応答するまでの計算時間」を示す. (表中の各処理ステップの意味は前述と同じ). 同表において，マージ法での「実体化」ステップの計算コストは, FROM(A

[5]なぜなら, V の元のうち FROM(A) 制約の答えにも FROM(B) 制約の答えにもならないコアを c とすると, c の始点に A も B も含まれない場合しかありえないからである (図 9 の c がその例). もし c が始点として A(または B) のページを持つなら, c を構成するリンクレコードは全て FROM(A)(または FROM(B)) 制約を満たすので, c は既に計算されているはずである.

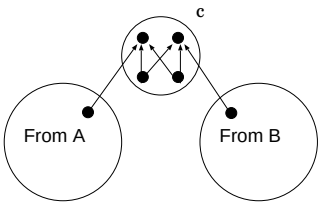

図 9: マージ法で新しく求めるコア

Fig.9: Core to be newly computed in the merge method

表 5: 「制約 FROM(A), FROM(B), FROM(A or B) の下でキューブを直接再実体化して問い合わせ処理した時の実行 時間」及び「マージ法による FROM(A or B) の問い合わせ処理時間」(いずれも $b = 12, s = 4$)

Table 5: Comparison of the merge method

処理内容	実行時間 (s)			
	実体化 A	実体化 B	実体化 A or B	マージ法 A or B
実体化	13	22	79	1
アイテムセットからコアへの変換	70	60	135	1
コアの極大化処理	23	19	62	1
後処理	79	83	244	244
合計	185	189	520	247

表 6: FROM(A or B) のコア計算に使われるレコード集合

Table 6: link-records of the merge method

処理方法	レコード数	リンク数
FROM(A or B) 再実体化	10862	51193
マージ法が C 計算に使うもの	122	865

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コアを計算する時間である. 実行時間は 1 秒であった. これに対し，FROM(A or B) の実体化の計算時間は 79 秒であった.

この原因として, 表 6 に,「FROM(A or B) 制約を満たすレコード集合」と「マージ法において実体化計算に用いるレコード集合」を示した. マージ法のコア計算対象となるレコード数がリンク数もレコード数も直接再実体化のそれらよりはるかに小さいことがわかる．すなわち, 前処理として FROM(A) 制約と FROM(B) 制約の実体化を既に行っているため，マージ法による差分計算の方がオンデマンド時の問い合わせ処理の手法としては適切である.

6. おわりに

本稿では，著者らが提案しているアイテムセットキューブモデルによる多次元制約下の Web 空間構造分析を対象に，その効率的な計算方法を提案した. はじめに,

- 「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 点を調べた. 具体的には，著者らの文献 [1, 2] のコア計算の手順に Kumar らの Prune 処理を導入して計算能力を改善し, その下でコア計算の細かさを制御するパラメータとして, 内部リンクの枝切りを行う下限値 b とコアの大きさを扱う s(最小サポート数) を調べ, 適切な細かさの下で計算可能となる b, s を調べて, あらかじめ実体化しておくべきアイテムセットキューブの範囲を設定した. 次に, 上の結果を受けて,

- 「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」

を課題とした．そして, ロールアップ演算の実行手法として, フィルタリング法とマージ法を提案し，電気通信大学ドメインの Web データを対象に評価を行った．その結果, FROM 制約と呼ぶ制約条件上のロールアップに相当する問い合わせ条件に応じて，直接的な再実体化処理よりも，提案手法が大幅に短い処理時間で応答可能であることを示した．TO 制約とよぶもうひとつの制約条件向けのフィルタリング法とマージ法の開発が現在の課題である.

[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, pp.141-144, 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] R.Kumar, P.Raghavan, S.Rajagopalan, A.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] 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.

栗原 大輔　Daisuke KURIHARA
2008 電気通信大学大学院修士課程了, 工修. データセンター技術等に興味を持つ. DBSJ 学生会員. 現 (株) 野村総合研究所.

大森 匡　Tadashi OHMORI
1990 東京大学大学院博士課程了, 工博. 電気通信大学大学院准教授. 並列データベースシステム, トランザクション処理, Web マイニング等を研究. ACM, IEEE, DBSJ 各正会員.

星 守　Mamoru HOSHI
東京大学大学院修士課程了, 工博. 1992 より電気通信大学大学院教授. 情報理論, データ構造等を研究. ACM, IEEE 等 正会員.