ゼロサプレス型二分決定木を用いた集合間類似結合

白井　康之[†,††]　　高嶋　宏之[†]　　鶴間　浩二[†††]　　小山　　聡[††]

† JST-ERATO 湊離散構造処理系プロジェクト　〒 060-0814 札幌市北区北 14 条西 9 丁目
††† 日本電気 (株)　〒 211-8666 川崎市中原区下沼部 1753
†† 北海道大学大学院情報科学研究科　〒 060-0814 札幌市北区北 14 条西 9 丁目
E-mail: †{shirai,takashima}@erato.ist.hokudai.ac.jp, ††k-tsuruma@ak.jp.nec.com,
†††oyama@ist.hokudai.ac.jp

あらまし　近年，データベース分野やデータマイニング分野において，集合間類似結合 (Set Similarity Joins) に関する研究が注目を集めており，購買履歴からの類似データ抽出や文字列照合等への応用が期待されている．集合間類似結合の方法論としては，データエントリに着目したフィルタリング手法に基づくものが多いが，本稿では，疎な大規模データ集合を効率的に表現する手法であるゼロサプレス型二分決定木 (ZDD：Zero-suppressed Binary Decision Diagrams) を用いた手法を提案する．ZDD を用いて表現されたアイテム集合間の類似結合アルゴリズムを示すとともに，他の手法との比較結果や実データを用いた実験結果を示す．

キーワード　集合間類似結合, 情報推薦, ゼロサプレス型二分決定木

Similarity Joins on Item Set Collections Using Zero-Suppressed Binary Decision Diagrams

Yasuyuki SHIRAI[†,††], Hiroyuki TAKASHIMA[†], Koji TSURUMA[†††], and Satoshi OYAMA[††]

† JST-ERATO Minato Discrete Structure Manipulation System Project
††† NEC Corporation
†† Graduate School of Information Science and Technology, Hokkaido University
E-mail: †{shirai,takashima}@erato.ist.hokudai.ac.jp, ††k-tsuruma@ak.jp.nec.com,
†††oyama@ist.hokudai.ac.jp

Abstract　In this paper, we propose a new approach to similarity joins based on Zero-suppressed Binary Decision Diagrams (ZDDs) for general item set collections, such as purchase history data, research keyword data, and so on. ZDDs are special types of Binary Decision Diagrams (BDDs), and suitable for implicitly handling large-scale combinatorial item set data. We show, in this paper, the algorithms for similarity joins between two data collections represented as ZDDs and the experimental results for performance comparison with other systems and the results using real huge data collections.

Key words　similarity joins, recommendation, zero-suppressed binary decision diagram

1. はじめに

近年，データベース分野や自然言語処理分野，あるいはデータマイニング分野において，アイテム集合を要素とする異なる集合（トランザクションデータベース）間での類似結合 (Set Similarity Joins) 手法が注目を集めている [1], [2], [4], [6], [13]〜[15], [17]〜[19]．集合間類似結合は，余弦や Jaccard 係数，編集距離といった尺度に基づき，2 つの大規模なトランザクションデータベースから類似したトランザクションのペアを正確にすべて抽出・列挙する手法であり，一般に要素間の組み合わせ数は膨大になることから，いかに効率的に与えられた条件を満たすペアを発見できるかが課題となっている．こうした集合間類似結合は，実データベースにおいて幅広い応用分野を持っている．たとえば，特許検索，データクリーニング，類似研究の発見，不正検知（Fraud detection），誤り検出，優良（不良）顧客の発見などへの応用が期待されている．

集合間で類似したトランザクションを正確にすべて列挙する問題は，集合間の組み合わせ処理を必然的に内包するため，より効率的なマッチングアルゴリズムが求められる．本稿では，購買履歴データや研究キーワードといったトランザクション

データベースに対する類似結合に対して，ゼロサプレス型二分決定木 (Zero-suppressed Binary Decision Diagrams，以下 ZDD) [8], [10], [12] を用いた効率的な実装方式を提案する．ZDD は Bryant による二分決定木（Binary Decision Diagrams，以下 BDD）[3] の変種であるが，大規模な組み合わせデータ集合に対してより圧縮効果の高いデータ構造である．

本稿の構成は以下のとおりである．2. 章では，関連する既存研究を概観し，3. 章で，基本的定義ならびに ZDD を用いた類似結合アルゴリズムを示す．4. 章では，実行効率評価として，トライ構造を用いた実装 [6], [17] との比較結果を示す．また，5. 章では，DBLP や NSF の実データを用いた解析結果を紹介する．6. 章では，本稿のまとめと今後の課題を示す．

2. 関 連 研 究

類似ペアをすべて正確に列挙する集合間類似結合に関して，近年，さまざまな取り組みが報告されている [1], [2], [4], [6], [13]〜[15], [17]〜[19].

Chaudhuri と Arasu ら [1], [4] は，SSJoin と呼ばれる一般的な演算を導入し，編集距離や Jaccard 係数，ハミング距離等さまざまな尺度へ拡張可能な集合間類似結合技術を提案している．SSJoin を用いた PartEnum や WtEnum といったアルゴリズムは，データの分割とシグネチャの生成・比較といった処理に基づき，効率的なフィルタリングを行う手法である．

Bayardo ら [2] は，与えられた閾値以下の関連度を持つトランザクションの組み合わせを効率的に除外するフィルタリング手法を示した．彼らの提案する All-Pairs アルゴリズムは，大規模なレコードサイズに対してもスケーラブルであることが示されている．

Xiao [18] らは，ミスマッチの個数に着目した Ed-Join アルゴリズムを示し，効率的な候補の絞り込み手法を提案している．

これらのアプローチは，いずれも不要なマッチングを可能な限り回避するためのフィルタリング手法である．すなわち，条件を満たさないペアを早期に枝狩りし，可能性のあるペアについては，テストフェーズとして検証が行われる．

こうした枝狩り手法は，アルファベットの個数やレコード長に対して，一般にセンシティブであり，うまくパラメータを設定できる問題に対しては有効であるものの，例えば，比較的長さの短いストリングに対しては，オーバーヘッドとなることも指摘されている [6], [17].

以上のようなフィルタリング手法に対して，Feng，Wang ら [6], [17] はトライ構造を用いて，Trie-Join と呼ばれる探索手法と枝狩り手法を提案している．また，Trie-Join を改良したより効率的なアルゴリズムも提案されている [7].

本稿では，Trie-Join [6], [17] 同様に，フィルタリング手法ではなく，データの表現方法とこれに基づく類似データの探索手法を検討するものとする．ただし，Trie-Join では，文字列等の系列シーケンスを対象としているのに対し，本稿では，アイテム集合（以下，トランザクション）を扱うものとする．また，Trie-Join では，編集距離を制約としているが，本稿では，順序のないトランザクションに対してより詳細な条件を指定可能な追加アイテム数と削除アイテム数に関する制約を扱うものとする．編集距離はシーケンスデータを対象とした場合には，編集コストという観点から有意性を持っているが，一方，購買履歴データ等の順序のないデータセットにおいては，アイテム集合間の相違をより細かいレベルで指定可能な追加アイテム数ならびに削除アイテム数に関する制約を与える方が適していると考えられる．

3. ZDD を用いた集合間類似結合

本研究では，トライ構造ではなく，ゼロサプレス型 BDD (ZDD) [8], [10], [12] を用いたアルゴリズムを提案する．ZDD は部分構造を共有することにより，疎なデータ構造をコンパクトに表現することが可能な無閉路有向グラフ（DAG）である．

本章では，基本的定義ならびに本論文で扱う類似結合について説明し，ZDD を用いた類似結合手法について記す．

3.1 準　　備

以下，いくつかの用語と記法を定義する．

定義 1 [アイテム]： アイテムは，小文字 $a, b, c, \ldots$ などを使って表わす．また，アイテムとして使用する文字集合を Σ で表わす．

定義 2 [トランザクション]： アイテムの集合をトランザクションと呼ぶ．すなわち，トランザクションはアルファベットの集合で表わされる（以下，D で表わす）．また，トランザクションの集合をトランザクションデータベースと呼び，$\mathcal{S}$ あるいは $\mathcal{T}$ などの記号を用いて表わす．

トランザクションデータベースを単に “データベース” と呼ぶこともある．

定義 3 [追加・削除制約]： トランザクションに追加または削除するアイテム数の上限をそれぞれ N^+, N^- で表わす．

たとえば，以下のようなトランザクションデータベース $\mathcal{S}$, $\mathcal{T}$ があったとする：

$$\mathcal{S} = \{\{a, b\}, \{a, c\}, \{c\}\} \tag{1}$$

$$\mathcal{T} = \{\{a, d\}, \{a, b, c\}, \{b\}\} \tag{2}$$

このとき，条件 $N^+ = N^- = 1$ における $\mathcal{T}$ の $\mathcal{S}$ に対する類似結合は以下の 6 つのペアから構成される：$\{\{a, b\}, \{a, d\}\}$, $\{\{a, c\}, \{a, d\}\}$, $\{\{a, b\}, \{a, b, c\}\}$, $\{\{a, c\}, \{a, b, c\}\}$, $\{\{a, b\}, \{b\}\}$, $\{\{c\}, \{b\}\}$.

もしここで条件が $N^+ = 1, N^- = 0$ であったとすると，$\{\{a, b\}, \{b\}\}$ のみが条件を満たすペアである．

3.2 ゼロサプレス型 BDD (ZDD)

本節では，二分決定グラフ（BDD）ならびにゼロサプレス型二分決定グラフ（ZDD）について概説する．

BDD [3], [8] は，大規模なブール関数を効率的に表現する無閉路有向グラフである．縮約ルールは，ノード削除ルール（2 つのエッジがともに同じノードを指しているようなノードを冗長なものとして削除する）ならびにノード共有ルール（等価な

サブグラフを共有する）からなる．一方，ZDD [8], [10], [12] は，大規模な組み合わせアイテム集合を効率的に取り扱うことができるように改良された BDD の変種である．ZDD の縮約ルールは，BDD のそれとは若干異なっている．

- 等価なノードを共有する（BDD と同様）．(図 1 の (1)).
- 1-edge が直接 0-ターミナルノードを指しているようなノードを削除し，0-edge の到達先に直接接続させる．(図 1 の (2)).

図 1 は，3.1 で示した $\mathcal{S}$ に対する ZDD の縮約例である．

ZDD は，BDD と比較して，購買履歴データなどのような疎なデータ構造に対してより効率的であることが知られている．たとえば，1000 アイテムから 10 アイテムを選択するような組み合わせ集合の表現においては，ZDD は BDD と比較して，100 倍程度簡潔に表現できる例もある．

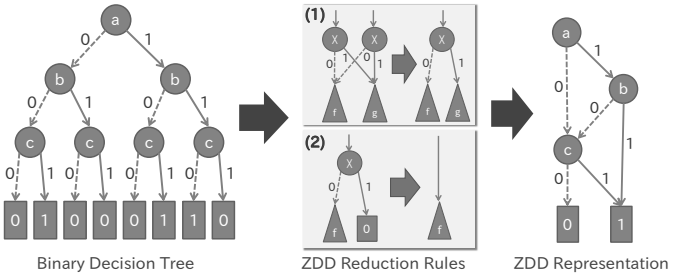

図 1　ZDD による二分決定木の縮約

3.3　ZDD に基づく類似結合

我々の手法においては，まず与えられた 2 つのトランザクションデータベースは，ZDD パッケージ [11] を介して，ZDD 構造に変換される．我々のシステムにおける入力データは，変換された ZDD 構造ならびに制約記述である．以下では，2 つの ZDD 構造に対する集合間類似結合方法について述べる．

ZDD 構造上での探索アルゴリズムの概要をアルゴリズム 1 に示す．図 2 は ZDD を用いた類似結合の例である．この図では，$N^+ = N^- = 1$ の制約のもとで $\mathcal{S}$ (左) と $\mathcal{T}$ (右) において類似したトランザクションを見つけることを目的としている．

探索プロセスは，トップノードである $\mathcal{S}$ 上の a より開始される（アルゴリズム 1 における $search\_zdd(n)$）．各エッジ上のボックスは，探索結果，すなわち $\mathcal{T}$ 上のパス（枝の系列）と，それに対応するアイテム追加数と削除数からなるリストである．たとえば，$\mathcal{S}$ 上のボックス (1) は，a の否定を表わすエッジにつけられている．ボックス (1) の $+0-1:2$ は，現状の制約が $\mathcal{T}$ 上のエッジ 2 に対して，“追加アイテムなし (+0) かつ 削除アイテム数 1 (−1)” であることを表わしている．同様に，ボックス (1) の $+0-0:1$ は $\mathcal{T}$ 上のエッジ 1 に対して，“追加アイテムなし (+0) かつ 削除アイテムなし (−0) ” であることを表わしている．

アルゴリズム中の $update\_candidate$ は現状の候補集合 $n_1.cand$ と $n_0.cand$ に対して，新しい候補を追加するものである．また，$reduce$ は候補集合を縮約し，制約検査を行うものである．

同様に，ボックス (2) は (1) の結果に基づき生成される．ボックス (4) を生成するためには，(2) の結果と同様に，(3) の結果も必要である．我々のアルゴリズムでは，各ノードに対する探索プロセスは，すべての親エッジが処理されたのちに起動する．ボックス (3) の計算後，ボックス (2)(3) の結果に基づき，ボックス (4) が生成される．

$\mathcal{S}$ において，d を追加するボックス (6) については，親である (4) と (5) におけるすべての追加カウントに 1 を追加しなければならない．なぜなら，$\mathcal{T}$ のトランザクションはいずれも d をその要素として持たないからである．

結果として，(6) のカウントは制約 N^+ を満たさない．もしボックスのすべての要素が条件 N^+ もしくは N^- を満たさない場合には，その枝上の探索プロセスは停止する．このため，ボックス (6) は ϕ となり，以降の探索は枝狩りされる．

Algorithm 1 ZDD 構造上の探索アルゴリズム

```
n_0 is a top node of the ZDD S;
n_0.cand = {{+0 − 0 : 0}};
search_zdd(n_0);

function SEARCH_ZDD(n)
  if all of other ancestors of node n have not been processed
then return
  end if
  if n is a terminal node then return cand;
    // output candidates
  else
    n_1 = n.edge_1.dest; // n_1 : destination of 1 edge of n
    n_0 = n.edge_0.dest; // n_0 : destination of 0 edge of n
    n_1.cand = update_candidate(n.edge_1, n.cand, n_1.cand);
    n_0.cand = update_candidate(n.edge_0, n.cand, n_0.cand);
    // update candidates for edge 1 and 0
    n_1.cand = reduce(n_1.cand);
    n_0.cand = reduce(n_0.cand);
    // reduction of the candidate set
    // and check the constraints
    if n_1.cand is not NULL then return search_zdd(n_1);
    end if
    if n_0.cand is not NULL then return search_zdd(n_0);
    end if
  end if
end function
```

図 2 で示した例では，制約を満たす解は存在しないが，図 3 で示した例では，2 つの結果，すなわち，$\{a, d\}$ ($\{a, c, d\}$ から c を削除) ならびに $\{b, c, d, e\}$ ($\{b, c, d\}$ への e の追加) を得る．

以下，本節で示した ZDD に基づく集合間類似結合を ZDD-Join と呼ぶ．ZDD-Join は，既存の ZDD パッケージを利用して ZDD データ構造を構築し，それらをもとにした類似結合アルゴリズムを C++ により実装したものである．

4.　Trie-Join との比較評価

既に述べたように，Trie-Join [6], [17] は，順序づけられたス

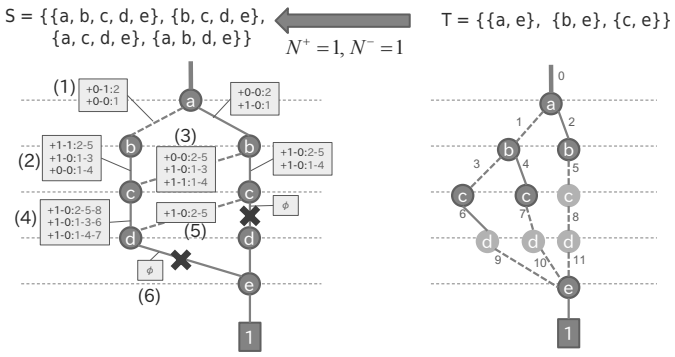

図 2　ZDD を用いた集合類似検索の例 (1)

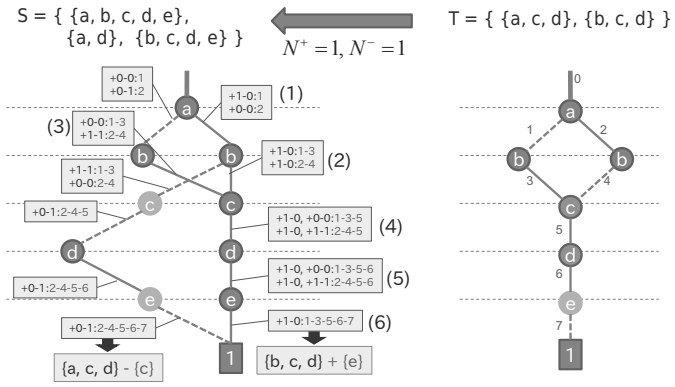

図 3　ZDD を用いた集合類似検索の例 (2)

トリングデータを対象としているが，本論文のアプローチと目的ならびに手法が類似している．対象とするデータ構造や制約の与え方が異なっているため，精緻な比較は困難であるが，本節では，人工的なデータを利用した比較実験結果について記すこととする．本実験で使用したデータは以下のとおりである：

- トランザクションは，Trie-Join においてはアルファベット順（“a”–“z”）に生成され，ZDD-Join においても同様に “x01”–“x26” を対応させて生成する．
- 各トランザクションに含まれるアイテム個数（トランザクションの長さ）は 10 とする．ただし，重複が発生した場合には，これを削除するものとする．

表 1 に，Trie-Join との比較実験で使用したデータの例を表わす．

表 1　比較実験に使用したデータ例

Trie-Join input	ZDD-Join input
aegklorstw	x01 x05 x07 x11 x12 x15 x18 x19 x20 x23
bcegjmtvxy	x02 x03 x05 x07 x10 x13 x20 x22 x24 x25
filmrsux	x06 x09 x12 x13 x18 x19 x21 x24
dijkmqrt	x04 x09 x10 x11 x13 x17 x18 x20
aeinprst	x01 x05 x09 x14 x16 x18 x19 x20
kqrvwy	x11 x17 x18 x22 x23 x25
aefghlqvx	x01 x05 x06 x07 x08 x12 x17 x22 x24
acgknoruxy	x01 x03 x07 x11 x14 x15 x18 x21 x24 x25

データは 100000，500000，1000000 トランザクションからなる 3 セットを用意した．Trie-Join (注1) ならびに ZDD-Join による実行結果を表 2 に示す．なお，以降の実験は，いずれも SUSE Linux Enterprise Server 11 上で，32 Intel Xeon CPUs (2.66 GHz) ならびに 1.024 TB RAM を用いて実行したものである．

また，図 4 は，表 2 に示した結果の中からいくつかをプロットしたものであり．ここで，横軸は出力された結果数を表わし，縦軸は実行時間（単位：秒）を表わす．

表からみられるように，より大規模な問題に対しては，少なくとも Trie-Join と同等のパフォーマンスを持っていることがわかる．たとえば，1,000,000 トランザクションで編集距離が 2 の制約の場合には，Trie-Join の実行時間は，約 149 百万個の結果に対して，436.6 (秒) を要している．一方，ZDD-Join においては，同じく 1,000,000 トランザクションを対象とし，$N^+ = N^- = 2$ の条件のもとでは，約 809 百万個の結果に対して，1039.0 (58.5 + 980.5) (秒) を要している．$N^+ = N^- = 2$ の条件は，編集距離 2 の条件と比較して明らかに緩く（実際，ZDD-Join の結果は Trie-Join の結果の真部分集合となっている），出力される結果数は Trie-Join の場合と比較しておよそ 5 倍となっている．

以上のように，2 つのシステムは，データセットに関する異なる前提を置いているうえ，制約の与え方も異なっているために，精緻な比較は困難であるが，生成される解の個数（すなわち問題のサイズ）と計算時間の関係をみる限り，少なくとも同等レベルのパフォーマンスを達成できていると考えられる．

5. 応 用 実 験

本章では，DBLP の論文タイトルや NSF の研究概要データ

(注1)：Trie-Join プログラムは以下にあるものを使用した．
http://dbgroup.cs.tsinghua.edu.cn/wangjn/codes/triejoin.tar.gz

表 2　ZDD-Join と Trie-Join との実行比較

ZDD-Join

Size of DB1	Size of DB2	Search Condition	Num. of Results	Exec.Time(sec)		
				ZDD Setting	Search	
100,000	100,000	add ≦ 1,delete ≦ 0	16,624	5.4	1.8	(Z1-1)
		add ≦ 0,delete ≦ 1	16,664		1.6	(Z1-2)
		add ≦ 1,delete ≦ 1	244,675		4.8	(Z1-3)
		add ≦ 2,delete ≦ 0	68,530		3.1	(Z1-4)
		add ≦ 0,delete ≦ 2	68,775		2.5	(Z1-5)
		add ≦ 2,delete ≦ 1	1,293,882		11.1	(Z1-6)
		add ≦ 1,delete ≦ 2	1,295,137		10.7	(Z1-7)
		add ≦ 2,delete ≦ 2	8,713,274		37.2	(Z1-8)
500,000	500,000	add ≦ 1,delete ≦ 0	401,823	27.9	12.6	(Z2-1)
		add ≦ 0,delete ≦ 1	405,813		10.2	(Z2-2)
		add ≦ 1,delete ≦ 1	5,922,847		31.5	(Z2-3)
		add ≦ 2,delete ≦ 0	1,649,671		19.2	(Z2-4)
		add ≦ 0,delete ≦ 2	1,681,117		16.4	(Z2-5)
		add ≦ 2,delete ≦ 1	31,163,852		84.4	(Z2-6)
		add ≦ 1,delete ≦ 2	31,554,584		86.1	(Z2-7)
		add ≦ 2,delete ≦ 2	210,967,890		368.7	(Z2-8)
1,000,000	1,000,000	add ≦ 1,delete ≦ 0	1,532,292	58.5	27.6	(Z3-1)
		add ≦ 0,delete ≦ 1	1,563,436		23.1	(Z3-2)
		add ≦ 1,delete ≦ 1	22,695,485		67.7	(Z3-3)
		add ≦ 2,delete ≦ 0	6,312,234		40.0	(Z3-4)
		add ≦ 0,delete ≦ 2	6,524,292		36.2	(Z3-5)
		add ≦ 2,delete ≦ 1	119,123,804		210.6	(Z3-6)
		add ≦ 1,delete ≦ 2	121,636,513		208.0	(Z3-7)
		add ≦ 2,delete ≦ 2	809,214,292		980.5	(Z3-8)

Trie-Join

Size of DB1	Size of DB2	Search Condition	Num. of Results	Exec.Time(sec)	
100,000	100,000	edit distance=1	78,315	1.1	(T1-1)
		edit distance=2	1,746,849	19.8	(T1-2)
500,000	500,000	edit distance=1	1,827,303	8.2	(T2-1)
		edit distance=2	40,789,678	159.1	(T2-2)
1,000,000	1,000,000	edit distance=1	6,701,562	23.1	(T3-1)
		edit distance=2	149,904,112	436.6	(T3-2)

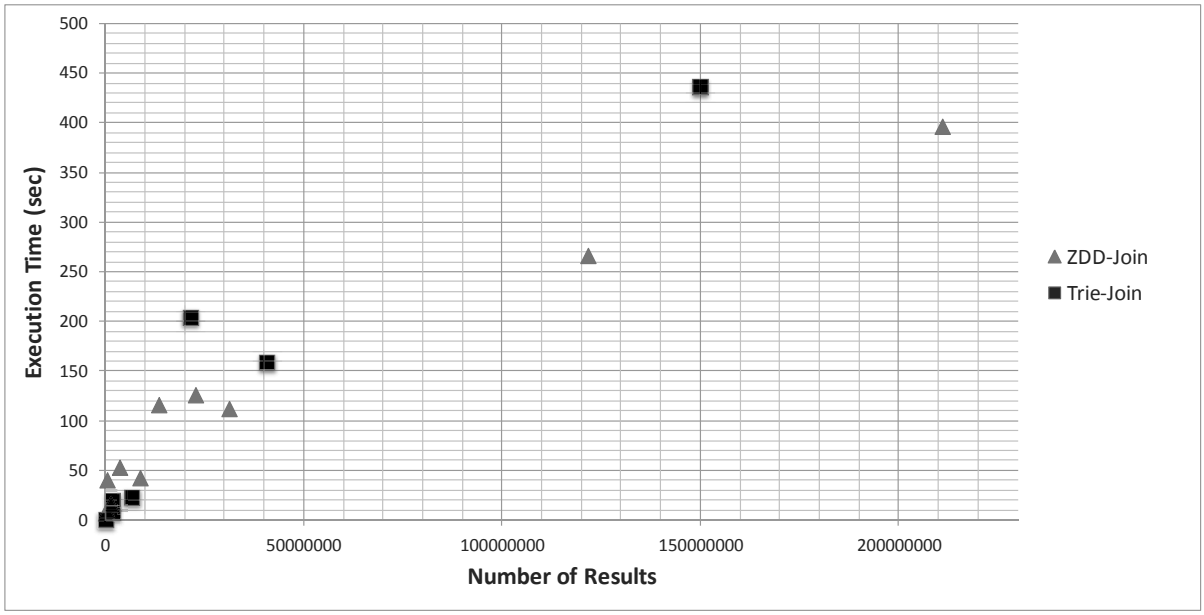

図 4　ZDD-Join と Trie-Join のパフォーマンス比較

を用いたいくつかの実験結果を示す.

5.1　DBLP 論文タイトル

ここでは，DBLP に含まれる論文タイトルを対象とする（DBLP のデータは，`http://dblp.uni-trier.de/xml` で xml 形式で公開されており，このうち，“article” または “inproceedings” のタグがついた 863,580 件のデータを対象とする）.

データセットから，発行年（publish year）を抽出し，発行年に応じてデータセットを以下のように三分割する.

- DS 1 (1997 年以前, 158706 トランザクション),
- DS 2 (1998 年〜2007 年, 348882 トランザクション),
- DS 3 (2008 年以降, 355992 トランザクション)

ここで各トランザクションのアイテムは，出現するキーワードを表わすものとする．ただし，本実験では，頻度が 10 未満のキーワードを削除した.

表 3 は，実験結果として，DS1 と DS2，また，DS2 と DS3 の類似結合の結果を記したものである．たとえば，1-a と 1-b は互いに類似であるとして検出されたものであり，1-a の発行年は 1987 年，また，1-b の発行年は 2001 年である（また互

いに異なる著者による）．しかしながら，前者の論文は実数列に対する計算複雑性を扱っているのに対して，後者はバイナリシーケンスに対する記述計算量を扱ったものであるため，この場合の直接的な関連は高くはない．一方，6-a，6-b は，ともに極大平面グラフの検出方法に関する論文であり，後者から前者への論文中での直接的な参照はないものの，相互に関連の深い研究であると考えられる．

以上のように，実応用を考えた場合には，単にタイトルだけでなく，アブストラクトを含めた特徴付けが必要であるが，集合間の類似結合により，関連の高い研究論文を検出することが可能であると考えられる．

5.2 NSF 研究概要

次に 1990〜2003 年までの NSF 研究概要データを用いた実験結果を記す．データベースは，合計で 129,000 エントリ (タイトル，期間，予算，概要，分野コード，...) からなり，タイトル，概要に含まれる単語の頻度データもともに提供されている．(注2)

実験では，“1996 年未満” と “1996 年以降” にデータを分割した．また，本実験では，“computer” を概要に含むエントリーを対象とした．各トランザクションは，タイトルまたは概要に出現するキーワードから構成されている．

実験結果の一部を表 4 に示す．ここで，$N^+ = 3, N^- = 3$ である．実験結果より，先に示した DBLP での結果同様，関連する過去の研究アクティビティが抽出されていることがわかる．たとえば，1-a (1992) は並列アルゴリズムに関する基本的技術に関する研究ミーティングであるが，一方，1-b (1999) は，並列計算の応用，すなわち，流体力学での並列計算に関するワークショップである．

一般に，プロジェクトの遂行やマネジメント，あるいはプロジェクト評価といった局面において，過去の類似したプロジェクトの成果を参考にすることは有意義であり，本実験のように膨大な過去データの中から類似したペアを検出することは応用面においても有用性が高いと考えられる．

6. まとめと今後の課題

本稿では，集合間類似結合に対して，疎なデータ集合に対する効率的な表現方法ならびに演算手法を提供するゼロサプレス型 BDD（ZDD）を用いた新しい方法論を提案した．

また，集合間類似結合において効率的な実装として知られている Trie-Join との比較結果を示し，ZDD を用いた方法が Trie-Join による方法と少なくとも同等程度であることを示した．特に，アイテム数が限定されたデータにおいては，重複する部分構造を如何に効率的に管理できるかが鍵である．また，Trie-Join においても，共有すべき部分構造へのリンクを貼ることにより，実質的に ZDD と同様なデータ管理を行っている．その意味においても，集合間類似検索において ZDD 構造を利用することは自然であり，かつ効率上の利点も大きい．

今後の課題は以下のとおりである．

- 他の手法で取り込まれている枝狩り手法やフィルタリング方法 [1], [2] あるいは Trie-Join におけるいくつかの枝狩り手法 [6], [17] （長さによる枝狩りや，単一枝による枝狩りなど）は，本アプローチでも取り入れることが可能である．
- 現実的な応用を考えた場合には，編集可能なアイテムや本質的なアイテムなど，アイテムに関する特徴付けが必須であり，こうした特徴付けに基づいたアイテムの置き替え（追加または削除）が必要である．これらの拡張は，実応用においては非常に有益であると考えられる．
- ZDD はアイテムの集合を扱うことができるが，順序付けられたストリングを扱うことができない．一方，Sequence BDD [5], [9] は系列集合を扱うように拡張された無閉路有向グラフである．文字列照合への応用として，Sequence BDD をベースにしたアルゴリズムを検討中である．

(注2)：本データセットは，UCI Machine Learning Repository において公開されているものである．(NSF Research Award Abstracts in UCI Machine Learning Repository： http://archive.ics.uci.edu/ml/datasets.html)

文　献

[1] A. Arasu, V. Ganti, R. Kaushik： Efficient Exact Set-Similarity Joins, In Proc. of 32nd International Conference on Very Large Data Bases (VLDB 2006), 2006

[2] R. J. Bayardo, Y. Ma, R. Srikant： Scaling Up All Pairs Similarity Search, In Proc. of 16th international conference on World Wide Web, 2007

[3] R. E. Bryant： Graph-based algorithms for Boolean function manipulation, IEEE Transactions on Computers, Vol. 35 Issue 8, 1986

[4] S. Chaudhuri, V. Ganti, R. Kaushik： A Primitive Operator for Similarity Joins in Data Cleaning, In Proc. of 22nd International Conference on Data Engineering (ICDE '06), 2006

[5] S. Denzumi, R. Yoshinaka, S. Minato, H. Arimura： Efficient Algorithms on Sequence Binary Decision Diagrams for Manipulating Sets of Strings, Hokkaido University, TCS Technical Reports, TCS-TR-A-11-53, 2011

[6] J. Feng, J. Wang, G. Li： Trie-join: a trie-based method for efficient string similarity joins, The VLDB Journal 21:437.461, 2012

[7] K. Gouda, M. Rashad： PreJoin: An Efficient Trie-based String Similarity Join Algorithm, The 8th International Conf. on Informatics and Systems (INFOS), 2012

[8] D. E. Knuth： The Art of Computer Programming, Vol. 4, No.1, Bitwise Tricks & Techniques, pp.117-126, Addison-Wesley, 2009

[9] E. Loekito, J. Bailey, J. Pei： A Binary decision diagram based approach for mining frequent subsequences, Knowledge and Information Systems, 24(2), 2010

[10] S. Minato： Zero-Suppressed BDDs for Set Manipulation in Combinatorial Problems, In Proc. of 30th ACM/IEEE Design Automation Conference (DAC'93), 1993.

[11] S. Minato： VSOP (Valued-Sum-of-Products) Calculator for Knowledge Processing Based on Zero-Suppressed BDDs, Federation over the Web, LNAI 3847, 2006.

[12] S. Minato： Implicit Manipulation of Polynomials Using Zero-Suppressed BDDs, In Proc. of IEEE The European Design and Test Conference, 1995.

[13] M. Neuhaus, H. Bunke： An Error-tolerant Approximate Matching Algorithm for Attributed Planar Graphs and its Application to Fingerprint Classification, In Proc. of Joint IAPR International Workshops, SSPR 2004 and SPR 2004, 2004

[14] K. Oflazer： Error-tolerant Finite-state Recognition with

表 3　DBLP データベースに対する実験結果

Experiments for articles on DBLP (1) (-1997 : 1998-2007)

Num.	Year	Title
1-a	1987	On The Complexity of Computable Real Sequences.
1-b	2001	Descriptive complexity of computable sequences
2-a	1982	Approximations for the waiting time distribution of the M/G/c queue.
2-b	2004	Mean Waiting Time Approximations in the G/G/1 Queue.
3-a	1979	On the connectivity of cayley graphs.
3-b	2005	Parameters of connectivity in (a, b)-linear graphs.
4-a	1993	An Affinity-Based Dynamic Load Balancing Protocol for Distributed Transaction Processing Systems.
4-b	2006	Dynamic Load Balancing Protocol for Locally Distributed Systems.
5-a	1989	Efficient monotone circuits for threshold functions.
5-b	2006	Monotone circuits for monotone weighted threshold functions.
6-a	1981	The reconstruction of maximal planar graphs. I. Recognition.
6-b	2004	A simple recognition of maximal planar graphs.

Experiments for articles on DBLP (2) (1998-2007 : 2008-)

Num.	Year	Title
1-a	2005	k-Center problems with minimum coverage.
1-b	2008	Asymmetric k-center with minimum coverage.
2-a	2006	On complexity of multistage stochastic programs.
2-b	2008	On Stability of Multistage Stochastic Programs.
3-a	2006	A remote laboratory for electrical engineering education.
3-b	2011	Developing a remote laboratory for engineering education.
4-a	2006	Arboricity and tree-packing in locally finite graphs.
4-b	2008	Locally finite graphs and embeddings.
5-a	2005	Transforming semantics by abstract interpretation.
5-b	2009	Abstract interpretation of resolution-based semantics.
6-a	2006	PolicyUpdater: a system for dynamic access control.
6-b	2008	A privacy-aware access control system.

表 4　NSF データベースに対する実験結果

Num.	Year	Title and Abstract
1-a	1992	Design of Parallel Algorithms • support for postdoctoral associate, • experimental computer science • developed for idealized parallel computers on real parallel computers, • load balancing techniques
1-b	1999	WORKSHOP: Parallel CFD'99 International Conference • computational fluid dynamics research on parallel computers, • parallel software system, • case studies from fluid dynamics, • early experiences on teraflops-class computers
2-a	1993	Constructing, Maintaining, and Searching Geometric Structures • construction and maintenance of geometric structures, • efficiently searching in such structures, • computational geometry unsolved problems, • dynamic maintenance of geometric structures, • computer graphics and computer vision, • sequential and parallel computation models
2-b	1999	Towards Simpler Algorithms in Computational Geometry • design and analysis of algorithms for large amounts of geometric data, • efficient algorithms for fundamental problems in computational geometry, • computer graphics and computer vision, • geometric optimization, • construction of basic geometric structures, • randomization, approximation, and techniques for correcting pessimistic worst-case analyses
3-a	1991	Undergraduate Computer Integrated Design Laboratory • establishment of an undergraduate computer integrated design laboratory, • development of a unified education program, • computer graphic simulation that gives insight into complex phenomena
3-b	1998	The Development of a Communication Networks Laboratory at Queens College • computer communication networks laboratory, • design and implementation of the Token Ring and Ethernet Local Area Networks, • undergraduate laboratory and an associated laboratory manual

Applications to Morphological Analysis and Spelling Correction, Computational Linguistics, 22(1), 1996

[15] K. Oflazer : Error-tolerant Tree Matching, In Proc. of 16th conference on Computational Linguistics (COLING '96), 1996

[16] Y. Shirai, K. Tsuruma, Y. Sakurai, S. Oyama, S. Minato : Incremental Set Recommendation Based on Class Differences, In Proc. of 16th Pacific-Asia Conference on Knowledge Discovery and Data Mining, LNAI 7301, 2012

[17] J. Wang, J. Feng, G. Li : Trie-Join: Efficient Trie-based String Similarity Joins with EditDistance Constraints, In Proc. of the VLDB Endowment, 3(1-2), 2010

[18] C. Xiao, W. Wang, X. Lin : Ed-join: an efficient algorithm for similarity joins with edit distance constraints, In Proc. of VLDB Endowment, 1(1), 2008

[19] C. Xiao, W. Wang, X. Lin, J. X. Yu : Efficient Similarity Joins for Near Duplicate Detection, In Proc. of 17th International Conference on World Wide Web, 2008