Vol.2011-DBS-153 No.11
2011/11/3

データアクセスの改良による時系列パターンマイニングアルゴリズムの高速化

松原裕貴[†1] 宮崎純[†1] 山本豪志朗[†1]
浦西友樹[†1] 池田聖[†1] 加藤博一[†1]

本論文では，我々が先行研究として提案を行った CPU キャッシュ利用効率の向上を意識した時系列パターンマイニングアルゴリズム CC-PAID に対し，更に CPU キャッシュ利用効率を向上させるための提案を行う．CC-PAID は既存の PAID アルゴリズムに対して，処理する時系列パターンに一括したアクセスを行うことにより，主に時間的局所性の向上を狙い CPU キャッシュミスを削減したものである．本論文の提案手法では，CC-PAID に対してデータアクセスの改良により更に CPU キャッシュミスの削減を実現する手法を提案し，性能評価により有効性の確認を行った．

A Fast Cache-consious Sequential Pattern Mining Alogrithm by Improved Data Access

Yuki Matsubara,[†1] Jun Miyazaki,[†1]
Goshiro Yamamoto,[†1] Yuki Uranishi,[†1] Sei Ikeda[†1]
and Hirokazu Kato[†1]

In this paper, we propose a sequential pattern mining algorithm which has more efficient CPU cache utilization than the CC-PAID algorithm. Though we have already proposed CC-PAID which has less CPU cache misses than PAID by mainly improving temporal locality by performing collective access to sequential patterns to be processed, we extend CC-PAID to reduce further CPU cache misses. We also evaluate the performance of the proposed method in this paper.

†1 奈良先端科学技術大学院大学 情報科学研究科
Graduate School of Information Science, Nara Institute of Science and Technology

1. はじめに

時系列パターンマイニングとは，時系列データにより構成されたデータベースから時系列パターンと呼ばれる出現順序を維持した状態の特定の閾値を満たす部分列を発見する手法であり，データ解析，行動予測，情報推薦といった分野において重要な技術とされている．時系列パターンマイニングの分野では小さな閾値と大規模なデータベースを利用した際，処理時間が膨大になるといった傾向があり，解決すべき重要な問題とされている．そのため，過去の研究においてアルゴリズムの提案や並列化による処理時間の改善が取り組まれてきた．

時系列パターンマイニングでは，アルゴリズムの特性上，特定のデータ構造に対して再帰的なアクセスが発生する傾向があり，CPU のキャッシュサイズを超えるようなデータ構造へのアクセスが生じる場合等でキャッシュミスによるデータアクセスのレイテンシが発生する可能性が考えられる．また，CPU のマルチコア化が進む今日においては，CPU の速度向上に対するメインメモリのアクセス速度の差が埋まることは考えにくく，メモリの壁[1] の問題はいまだ解決していない．それ故に，キャッシュミスにより生じるアクセスレイテンシは時系列パターンマイニングの処理においてもボトルネックとなりうるため，我々は先行研究として既存の時系列パターンマイニングアルゴリズム PAID[2] に対して CPU キャッシュミスの削減を行った時系列パターンマイニングアルゴリズム CC-PAID[3] を提案した．

CC-PAID は主にデータ構造に対しての時間的局所性の向上による CPU キャッシュ利用効率の改善を狙ったものである．PAID アルゴリズムにおいて，共通する接頭辞を持つ時系列パターンは特定のデータ構造で処理範囲が共有可能であり，CC-PAID ではそれらの時系列パターンを一括に処理対象とするといったアクセスパターンに変更することで CPU キャッシュに取り込まれたデータの時間的局所性などを改良する．また，CC-PAID アルゴリズムの性能評価を行った結果として，PAID アルゴリズムに対して 2 倍近い高速化と 6 割程度の CPU キャッシュミスの削減が確認できた．この結果から，我々は時系列パターンマイニングでの CPU キャッシュ利用効率の向上による処理時間の短縮が有効であると考えた．

本論文では，CC-PAID アルゴリズムの CPU キャッシュ利用効率を更に向上させる手法を提案し，それを評価する．提案手法では，再構築された特定のデータ構造を利用することにより二つの処理でデータアクセスの効率化を図る．一つ目は，不要な処理によるデータアクセスの発生を回避をする．二つ目は，処理に必要が無くなったデータを削除して，CPU キャッシュへの読み込みサイズの縮小を行う．

Vol.2011-DBS-153 No.11
2011/11/3

2. 時系列パターンマイニング

時系列パターンマイニングは時系列データベースから最小支持度を満たす出現順序を維持した部分列を時系列パターンとして発見する手法である．時系列データベース (SDB) は複数の時系列データにより構成され，時系列データはアイテムセットにより構成される．

時系列データベースは，$SDB =< s_1, s_2, \cdots, s_n >$ と示され，s_k は時系列 ID （以降 SID）が付与された時系列データであり，SID 順に並んだ時系列データで構成される．時系列データ s は，$s =< is_1, is_2, \cdots, is_m >$ で表され，is_k はアイテムセットであり，これを 1 つのトランザクションとしてトランザクション ID （以降 TID）と呼ばれる時間情報または出現順序により識別され，TID 順に並んだアイテムセットで構成される．また，アイテムセット is は，$is = (i_1, i_2, \cdots, i_c)$ であり，アイテムにより構成されている．時系列データに含まれる TID による順序関係を維持したアイテムの組合せを時系列データの長さ k の部分列 α としたとき，全時系列データの数に対して長さ k の部分列 α が出現する割合または出現回数を支持度と呼ぶ．

時系列パターンマイニングでは，閾値となるユーザが決定した支持度が最小支持度となり，時系列データベース内で長さ k の部分列 α の支持度を調べ，最小支持度を満たす全ての部分列 α を時系列パターンとする．また，時系列パターンの処理は itemset-extention step （以降 I-Step），sequence-extention step （以降 S-Step）に分けられる[4]．SDB 内に現れる is に関して相関ルールマイニングを行い，is の中で最小支持度を満たす組合せを導出するのが I-Step であり，時系列パターンを導出するのが S-Step である．

時系列パターンマイニングを行った際の時系列パターンが出力される例を示す．例として，長さ 1 のアイテムセットにより構成される時系列データベース（図 1）から最小支持度（回数）を 4 として，それを満たす時系列パターンの抽出例を図 2 に示す．図 2 の例には，$A \rightarrow B \rightarrow A : 4$ があるが，これは長さ 3 の部分列 $A \rightarrow B \rightarrow A$ が図 1 の時系列データベース上に 4 回出現することを意味する．

3. 関 連 研 究

本節では，時系列パターンマイニングアルゴリズムと CPU キャッシュ利用効率に関する論文の紹介を行う．

Pei らは時系列データベース上で最小支持度を満たすアイテムを直接発見し，長さ k の時系列パターンに連結させることにより長さ $k+1$ の時系列パターンを発見する PrefixSpan アルゴリズム[5] の提案を行っている．処理対象の時系列パターンより後に出現するデータ範囲を投影時系列データベースと呼び，投影時系列データベース内で最小支持度を満たすアイテムを発見し，長さ k の時系列パターンの後ろに追加して，長さ $k+1$ の時系列パターンを発見する．時系列パターン長が長くなるほどアイテムの探索範囲を縮小できるため，効率良く時系列パターンを発見することが可能となる．

Wang らは時系列データベースのスキャンコストを減らすことにより処理性能を改善させた FSPM アルゴリズム[6] の提案を行っている．PrefixSpan が様々なアイテムにより構築される時系列データベース上で時系列パターンの発見処理を行う際に，それぞれの時系列パターンの処理で同じアイテムがスキャンされることに着目し，最小支持度を満たすアイテムごとに長さ $k+1$ の時系列パターンの発見処理を分け，その中から一つの最小支持度を満たすアイテムを処理対象として選択して，そのアイテム含む全ての時系列パターンを抽出する．これにより，次のアイテムを対象に処理すると，前の処理で処理対象とされたアイテムを時系列データベースの探索範囲から外して処理することができ，探索範囲を減らすことができる．

Yang らは，ILP-list と呼ばれる時系列データ上の長さ 1 の時系列パターンの最終出現位置のリストと長さ $k+1$ の時系列パターンを発見する処理で長さ k の時系列パターンの支持度を有効的に利用する PAID アルゴリズム[2] の提案を行っている．PAID は，ILP-list を利用することにより，時系列データ上で長さ $k-1$ の時系列パターンが出現する位置より後の範囲から，長さ k の時系列パターンが出現する位置より後に出現しなくなるアイテムを調べる．出現しなくなるアイテムは長さ $k+1$ の部分列の $k+1$ 番目のアイテムとして成り立たないため，前の処理で得られている長さ k の時系列パターンの支持度から差し引くこ

TID \ SID	sequence data					
	1	2	3	4	5	6
1	A	C	B	C	A	D
2	A	D	B	A	C	
3	B	A	A	D	C	A
4	C	C	D	A	B	A
5	C	A	C	A	B	A

図 1 時系列データベース
Fig. 1 Sequence datdabase.

length	Sequential pattern : Support
1	A:5, B:5, C:5, D: 4
2	A→A:5, A→B:4, A→C:4, B→A:5, C→A:4
3	A→B→A:4

図 2 最小支持度（回数） 4 を満たす時系列パターン
Fig. 2 Sequential patterns where minimum support is four.

とにより支持度を更新する．これにより，支持度の更新に関連する処理範囲を減らし，処理速度を向上させる．

一方，相関ルールマイニングではアルゴリズムをキャッシュ指向化して高速化を行う研究も行われている．Gohting らは，頻出パターンマイニングアルゴリズム FP-growth[7] で利用されるデータ構造に対し，連続してデータにアクセスできるようなデータの再配置やデータサイズを抑えるといった改良により，空間的局所性の改良を行っている．また，CPU のキャッシュに収まるようにタイルといった単位でデータの分割を行い，フェッチしたタイルを無駄なく利用することにより時間的局所性の改良も行っている．ほかにも，マルチスレッドによりデータの再利用率を向上させることにより，処理速度を向上させた[8]．

4. CC-PAID アルゴリズム

本研究の改良対象となる我々が先行研究として提案を行った時系列パターンマイニングアルゴリズム CC-PAID[3] の説明を行う．

CC-PAID は，既存の時系列パターンマイニングアルゴリズム PAID[2] の時間的局所性等の改良を行い，CPU キャッシュ利用効率の向上による処理時間短縮を目的とした手法である．また，CC-PAID の CPU キャッシュ利用効率の向上は，PAID のアルゴリズム自体を変更することなく，処理対象となる時系列パターンへのアクセスパターンを変えることにより達成される．時系列パターンは木構造で表わすことができ[4]，PAID では深さ優先探索により一つの長さ k の時系列パターンを選択し，長さ $k+1$ の時系列パターンを発見する．それに対し，CC-PAID では長さ $k+1$ の時系列パターンの発見処理において，処理対象として共通する接頭辞を含む長さ k の複数の時系列パターンを選択することにより，長さ $k+1$ の時系列パターンを発見する．これにより，CPU キャッシュに取り込まれた支持度の調査に関係するデータ構造などが他の時系列パターンの処理で直ぐに再利用され，CPU キャッシュミスが軽減される．ここで，図 1 の時系列データベースを対象に時系列パターン $A \to B \to A$ が発見されるまでの PAID と CC-PAID のアクセスパターンの例を図 3 と図 4 に示す．枠で囲まれた時系列パターンは処理が終了したことを表す．CC-PAID により時系列パターン $A \to B \to A$ が発見されるこの例では，長さ 2 の時系列パターン $A \to A$, $A \to B$, $A \to C$ が幅優先探索のように処理対象として選択される．また，これらの共通する接頭辞は長さ 1 の時系列パターン A である．

CC-PAID による時系列データごとの支持度の調査の流れを説明する．CC-PAID は支持度の調査のために大きく分けて二つのデータ構造にアクセスする．一つは Position list で

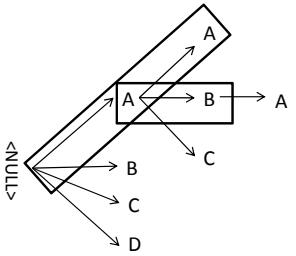

図 3 PAID アクセスパターン
Fig. 3 Access pattern of PAID.

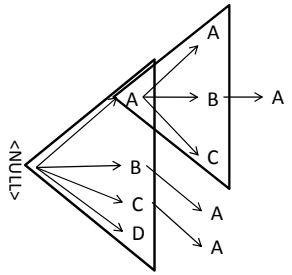

図 4 CC-PAID アクセスパターン
Fig. 4 Access pattern of CC-PAID.

あり，もうひとつは Item last position list （以降 ILP-list）である．Position list は時系列データに含まれるアイテムの TID をアイテムの種類ごとにまとめたものである．また，ILP-list は時系列データを構成する長さ 1 の時系列パターンとそれが時系列データ上で最後に出現する TID を記録したものである．

まず，時系列データ上で長さ k の時系列パターンが出現する TID を調べる．長さ k の時系列パターンの TID を k-prefix border position (以降 k-pbp) とし，その時系列パターンが含む長さ $k-1$ の接頭辞の TID を $k-1$-prefix border position (以降 $k-1$-pbp) とする．そして，処理対象となる時系列パターンに対応する Position list から $k-1$-pbp より大きい k-pbp を探す．この処理は処理対象になっている複数の長さ k の時系列パターンで行われ，得られた k-pbp は長さ k の時系列パターンの k 番目のアイテムと共に，multi-k-pbp として記録される．

次に，得られた k-pbp ごとに ILP-list で k-pbp の対応位置を調べる処理と支持度からの差し引き処理が行われる．k-pbp の対応位置を調べる処理では ILP-list 上で $k-1$-pbp が対応する位置以降で k-pbp より大きい位置を調べる．また，得られた k-pbp の位置を PAID では candidate border index としており，次回の長さ $k+1$ の時系列パターンの処理で，ILP-list の $k-1$-pbp の対応位置として利用可能であり，同様に CC-PAID でも利用可能である．本論文では ILP-list 上の $k-1$-pbp が対応する位置を S_{k-1}-position，k-pbp が対応する位置を E_k-position とする．ILP-list 上で S_{k-1}-position から E_k-position までに出現するアイテムは，その時系列データ上で長さ $k+1$ の部分列の $k+1$ 番目のアイテムとして出現しないため，これらのアイテムは長さ $k-1$ の時系列パターンの投影 ILP-list のデー

タベースのアイテムの支持度から差し引かれる．また，長さ $k-1$ の時系列パターンの投影 ILP-list のデータベースとは S_{k-1}-position 以降の範囲の全時系列データ分の集合である．

Position list と ILP-list に関する処理を長さ $k-1$ の共通する接頭辞を持つ長さ k の時系列パターンで一括して行い，全時系列データで処理が終了したとき，それぞれの長さ k の時系列パターンで長さ $k+1$ の部分列の支持度が決定され，最小支持度を満たす部分列を長さ $k+1$ の時系列パターンとして発見する．

ここで，図 1 の時系列データベースの SID 1 と最小支持度 4 を使った時系列パターン $A \to A$, $A \to B$, $A \to C$ を処理対象とした例を示す．まず，図 1 の時系列データの SID 1 を対象とした図 5 のアイテム A, B, C の Position list から multi-k-pbp を取得する．その時，$k-1$-pbp は 1 なので TID 1 より大きい TID をそれぞれ探す．それぞれの k-pbp は k 番目のアイテムと共に保存され，結果として multi-k-pbp=$\{(A,5),(B,3),(C,2)\}$ となる．

次に，S_{k-1}-position 以降の範囲で支持度から差し引くアイテムを決める．図 6 は図 1 の時系列データベースの SID 1 を対象としたもので，時系列パターン $A \to A$, $A \to B$, $A \to C$ の処理を表わしている．時系列パターン $A \to A$ について着目すると，k-pbp=5 であり，差し引き対象となるアイテムは A，B，C となる．それらのアイテムは時系列パターン A の投影 ILP-list のデータベースのアイテム支持度から差し引かれる．その他の時系列パターン $A \to B$, $A \to C$ についても，差し引かれるアイテムを探し，支持度から差し引く処理を行う．

CC-PAID では，共通する接頭辞を持つ複数の長さ k の時系列パターンを一括して処理するため，CPU キャッシュに取り込まれた ILP-list が他の時系列パターンによって直ぐに再利用されるため，時間的局所性が改良される．そのほかにも，一括した処理により，他の部分 (例えば，position list 等) でも CPU キャッシュ利用効率が向上する可能性がある．CC-PAID は PAID と比較して最大で約 2 倍の速度向上と約 6 割の CPU キャッシュミスの削減が確認されている．

5. 提案手法

本節では，CC-PAID のデータアクセス効率化手法の提案を行う．CC-PAID では，発見された時系列パターンに対しての処理対象へのアクセスパターンの変更により，CPU キャッシュ利用効率を向上させ，処理時間を短縮している．その結果から，時系列パターンマイニングでは CPU キャッシュ利用効率の改善は有効であると判断できる．そこで，我々はアクセスパターンとは別のアプローチによって CPU キャッシュ利用効率を向上させる手法の検討を行った．時系列パターンマイニングの特徴として，特定のデータ構造に対して再帰的なアクセスが発生する傾向があり，CC-PAID では Position list と ILP-list がそれに該当すると考えられる．データ構造へのアクセスは CPU キャッシュ利用効率に関連があると考えられるため，我々はこの二つのデータ構造に対してのアクセスについて考察した．

SID	Item ID	Transaction ID
1	A	1, 5
	B	3
	C	2, 4
	D	6

図 5 図 1 の SID 1 を対象とした Position list
Fig. 5 Position list table of SID 1 in Fig 1.

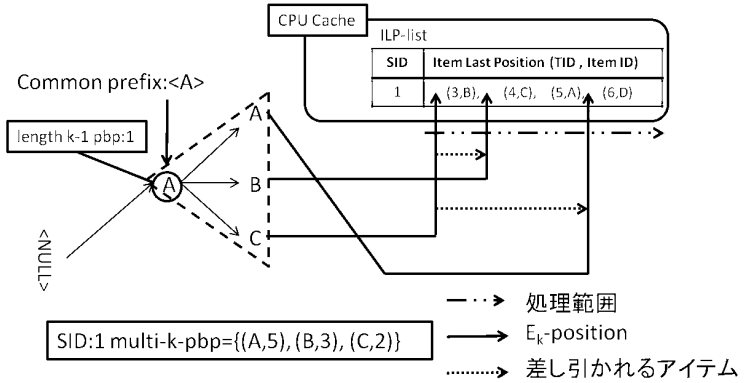

図 6 図 1 の SID 1 を対象とした ILP-list の処理
Fig. 6 Process of an ILP-list of SID 1 in Fig 1.

ここで，CC-PAID の処理の流れと利用されるデータ構造を示す (図 7)．CC-PAID の処理は大きく分けて，multi-k-pbp の取得と支持度更新に関する処理の二つになる．multi-k-pbp では長さ k の時系列パターンのそれぞれで position list にアクセスする．支持度の更新処理は ILP-list から差し引かれるアイテムを探す処理とそれらのアイテムを支持度のリスト (Support list) から差し引く処理を意味し，それらの処理は長さ k の時系列パターンごとに

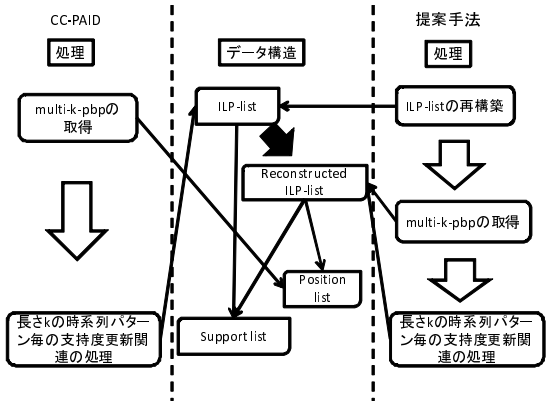

図 7 CC-PAID と提案手法の処理とデータ構造
Fig. 7 Processes and data structures in CC-PAID and proposed method.

行われる．また，支持度のリストから差し引く処理は ILP-list を介して行われる．

5.1 Position list でのデータアクセス

Position list は長さ k の時系列パターンの prefix border position (k-pbp) の取得に用いられる．CC-PAID では長さ $k-1$ の時系列パターンの prefix border position ($k-1$-pbp) が Null である場合を除いて，成立している長さ k の時系列パターンに関する k-pbp の取得処理が行われる．CC-PAID では k-pbp の取得の際に，長さ k の時系列パターンとしては成立しているが，処理対象となる時系列データにはその時系列パターンが存在しないといった場合が考えられる．その場合，k-pbp は Null となる．

図 8 の例はある時系列データベースで長さ 3 の時系列パターン $A \to B \to A$, $A \to B \to B$, $A \to B \to D$ が成立しており，ある時系列データの Position list の処理を行っている．時系列パターン $A \to B \to B$ はアイテム B の Position list で探索処理を行い，k-pbp として NULL を得る．なお，以降の例では図 8 と同じ時系列データと時系列パターンを対象とする．

もし k-pbp を取得する前に，対象となっている長さ k の時系列パターンが時系列データに含まれているか否かが判断できれば，含まれていない時系列パターンに関する処理を省略できると考えられる．このため，提案手法では ILP-list を用いることにより，長さ k の時系列パターンが含まれるか判断を行う．ILP-list は長さ 1 の時系列パターンの最終出現位置を記録したものであるため，ILP-list 上で S_{k-1}-position より前にあるアイテムは時系列

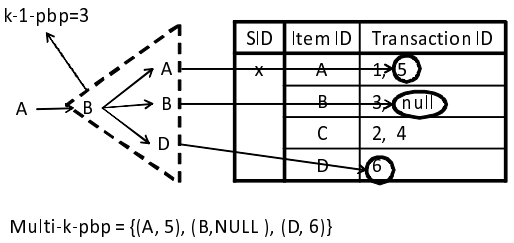

図 8 CC-PAID の Position list へのアクセス
Fig. 8 Access to position lists in the CC-PAID algorithm.

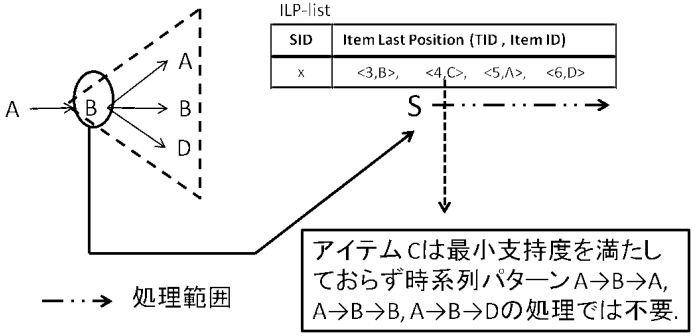

図 9 CC-PAID の ILP-list での処理範囲
Fig. 9 Process range in the CC-PAID algorithm.

データ上で長さ k の部分列の k 番目のアイテムとして出現しないことがわかる．提案手法では S_{k-1}-position 以降で長さ k の時系列パターンの k 番目のアイテムが存在するか調べることにより，存在しない場合の長さ k の時系列パターンの処理を省略する．

5.2 ILP-list でのデータアクセス

ILP-list は支持度から差し引かれるアイテムを調べるのに用いられる．CC-PAID では，ILP-list の S_{k-1}-position 以降の処理範囲に長さ $k+1$ の時系列パターンの $k+1$ 番目のアイテムとなる可能性のないアイテムが含まれている可能性がある．これは長さ k の時系列パターンの k 番目のアイテムとして存在するか否かで判断できる．なぜならば，k 番目のアイテムとして成立していないなら，そのアイテムは既に最小支持度を満たしていないためで

ある．

図 9 の例では，長さ 3 の時系列パターン $A \to B \to A$, $A \to B \to B$, $A \to B \to D$ が成立しているため，長さ 4 の時系列パターンの 4 番目のアイテムとなる可能性があるのはアイテム A，B，D である．しかし，S_{k-1}-position (時系列パターン $A \to B$ の対応位置) 以降にアイテム C の要素が含まれており，これは無駄な要素と判断できる．

FSPM アルゴリズム[6] では，長さ $k+1$ の時系列パターンの $k+1$ 番目のアイテムとして成立する可能性のあるアイテムを candidate-set として，それに含まれないアイテムを支持度の調査に利用される範囲から除外しており，これを candidate-driven として提案を行っている．ILP-list も同様に $k+1$ 番目の成立する可能性のないアイテムを処理範囲から除外できると考えられる．

5.3 ILP-list の再構築

提案手法では，Position list を用いた multi-k-pbp の取得の際，処理しない長さ k の時系列パターンを判断するために，S_{k-1}-position 以降から順にアクセスして長さ k の時系列パターンの k 番目のアイテムが存在するか調べ，存在するアイテムの position list にアクセスを行う．同様に，支持度の更新に関連する処理に用いられる ILP-list でも処理が不要なデータへのアクセスを避けるため，S_{k-1}-position 以降で長さ k の時系列パターンの k 番目のアイテムとして成立するアイテムのみにアクセスを行う．二つの処理において必要な ILP-list のデータは一致しているため，提案手法では上記二点の処理が行われる前にあらかじめ ILP-list から必要なデータを取り出して，ILP-list の再構築を行う．

両方の処理において，別々に必要なデータのみにアクセスして処理や再構築等を行う場合，そのコストが大きくなる可能性がある．事前に再構築された ILP-list を両方の処理で利用することにより，コストに対するメリットが大きくなると考えられる．また，CC-PAID では再構築された ILP-list は複数の時系列パターンにより利用される．なお，本提案手法では ILP-list の再構築は必要なデータの複製により行われる．データの複製はメモリ消費が大きくなる可能性があるが，CPU キャッシュについて考えると，必要なデータのアドレスが近くなり，取り込まれるサイズも小さくなると考えられる．

5.4 提案手法の処理の流れ

提案手法による支持度更新の処理までの流れ (図 7) を例を用いて示す．

まず ILP-list の再構築を行う．S_{k-1}-position 以降の長さ k の時系列パターンの k 番目として成立しているアイテムとその TID を全て抜き出し，それらのデータをコピーすることにより ILP-list の再構築を行う．例 (図 10) として，長さ 3 の時系列パターン $A \to B \to A$,

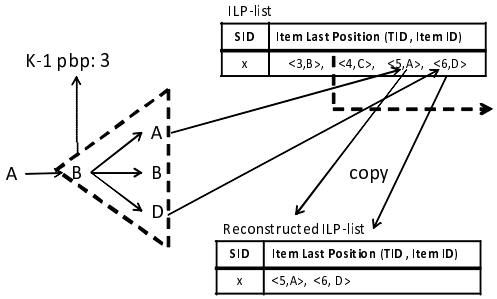

図 10 ILP-list の再構築
Fig. 10 Reconstruction of ILP-list.

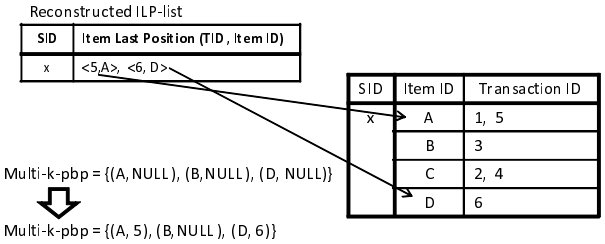

図 11 再構築された ILP-list を用いた multi-k-pbp の取得
Fig. 11 Finding multi-k-pbp by using a reconstructed ILP-list.

$A \to B \to B$, $A \to B \to D$ があり，common prefix である時系列パターン $A \to B$ の $k-1$-pbp が 3 のとき，S_{k-1}-position はアイテム C からであり，それ以降からアイテム A，B，D を探す．結果として，抜き出されるアイテムと TID は <5, A>，<6, D> となり，これが再構築された ILP-list の要素となる．

次に，再構築された ILP-list にアクセスし，それに含まれるアイテムの Position list へアクセスし，multi-k-pbp を取得する．図 11 では，再構築された ILP-list にはアイテム A と D が含まれているのでそれらの Position list にアクセスし，k-pbp を取得する．CC-PAID ではアイテム B の Position list が処理されるが提案手法ではその処理は省かれる．

最後に，再構築した ILP-list にアクセスし，長さ k の時系列パターンのそれぞれで支持度の更新に関係する処理を行う．再構築された ILP-List の処理範囲となるデータは <5, A>,

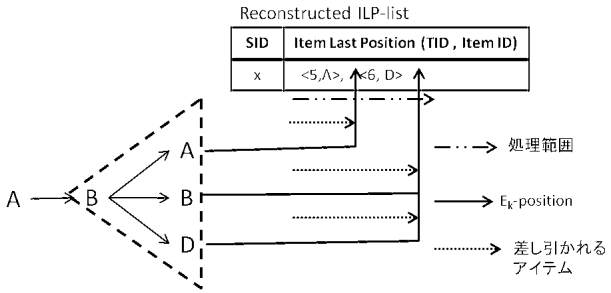

図 12 再構築された ILP-list を用いた支持度更新の関連処理
Fig. 12 Update of support counts using a reconstructed ILP-list.

<6, D> の二つである（図 12）．CC-PAID では S_{k-1}-position 以降を考えると <4,C> を含む三つの要素が処理範囲となるので，再構築された ILP-list は処理範囲を縮めることが可能である．なお，再構築した ILP-list は次の長さ $k+1$ の時系列パターンの発見処理で利用可能である．

6. 性能評価

本節では，時系列パターンマイニングアルゴリズム PAID，CC-PAID，提案手法についての性能評価を行う．IBM data generator により，3 つのデータセットの生成を行った．生成時に利用したパラメータを表 1 に示す．また，評価環境を表 2 に示す．評価内容は，3 つのデータセットを用い，最小支持度を変更して処理時間と L3 キャッシュミスの計測を行うものである．L3 キャッシュミスは，Intel VTune Amplifier XE 2011 を用いて計測した見積もり値である．なお，性能評価は S-step のみの評価となる．データセットごとの性能評価の結果を示す．

- データセット D_1，最小支持度 0.24 ～ 0.3 (図 13)
  CC-PAID に対して提案手法では，処理時間では最小支持度 0.24 のときに，約 16%の短縮，L3 キャッシュミスでは最小支持度 0.26 のとき，約 18%の削減が確認された．
- データセット D_2，最小支持度 0.6 ～ 0.66 (図 14)
  CC-PAID に対して提案手法では，処理時間では最小支持度 0.6 のときに，約 10%の短縮，L3 キャッシュミスでは最小支持度 0.6 のとき，約 16%の削減が確認された．
- データセット D_3，最小支持度 0.08 ～ 0.14 (図 15)

表 1 データセットのパラメータ
Table 1 Data set parameters.

	Data set name		
	D_1	D_2	D_3
時系列データ数	50000	50000	50000
時系列データ内の平均トランザクション数	50	100	50
トランザクション内の平均アイテム数	4	4	4
アイテムの種類数	350	350	700

表 2 評価環境
Table 2 Experimental environment.

OS	Fedora 13 64bit	
Kernel	2.6.34.7-61	
CPU	Intel Core i7 980X	
	周波数	3.33GHz
	コア数	6
	L2 cache	256KB x 6
	L3 cache	12MB
Main memory	12GB	

CC-PAID に対して提案手法では，処理時間では最小支持度 0.08 のときに，約 25%の短縮，L3 キャッシュミスでは最小支持度 0.08 のとき，約 28%の削減が確認された．

性能評価により全てのデータセットと最小支持度で，CC-PAID に対して提案手法では処理時間の短縮と L3 キャッシュミスの削減が確認された．特に，アイテムの種類数が多いほど処理時間の短縮と L3 キャッシュミスの削減がされることが確認できた．これは，時系列パターン長が短いときに多くの部分列が最小支持度を満たさなくなり，その結果，早い段階で Position list の処理対象が少なくなることや ILP-list が短くなるといった要因が考えられる．

7. まとめ

我々は，時系列パターンマイニングアルゴリズム CC-PAID で用いられるデータ構造を再構築し，2 つの処理で有効的に利用することにより，データアクセスの効率化を図り，CPU キャッシュ利用効率等を改善することにより処理時間を短縮する手法の提案を行った．また，性能評価を行い，提案手法は CC-PAID に対して処理時間で最大約 25%の短縮，L3 キャッシュミスで最大約 28%の削減が確認できた．今後の課題として，GPGPU といったメニー

コアプロセッサによる有効的な処理時間の短縮手法の提案があげられる．

謝　　辞

本研究の一部は、科研費補助金基盤研究 (A)(課題番号: 22240005) ならびに若手研究 (B)(課題番号: 21700111) の支援による。ここに記して謝意を表す。

参 考 文 献

1) Wulf, W.A. and McKee, S.A.: Hitting the memory wall: implications of the obvious, *SIGARCH Comput. Archit. News*, Vol.23, pp.20–24 (1995).
2) Yang, Z., Kitsuregawa, M. and Wang, Y.: PAID: Mining Sequential Patterns by Passed Item Deduction in Large Databases, *Proceedings of the 10th International Database Engineering and Applications Symposium*, Washington, DC, USA, IEEE Computer Society, pp.113–120 (2006).
3) 松原裕貴，宮崎　純，藤澤　誠，天野敏之，加藤博一：CC-PAID：CPU キャッシュを有効利用した並列時系列パターンマイニングアルゴリズム，情報処理学会論文誌データベース（TOD），Vol.4, No.2, pp.88–100 (2011).
4) Ayres, J., Flannick, J., Gehrke, J. and Yiu, T.: Sequential PAttern mining using a bitmap representation, *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '02, New York, NY, USA, ACM, pp.429–435 (2002).
5) Pei, J., Han, J., Mortazavi-Asl, B., Wang, J., Pinto, H., Chen, Q., Dayal, U. and Hsu, M.-C.: Mining Sequential Patterns by Pattern-Growth: The PrefixSpan Approach, *IEEE Transactions on Knowledge and Data Engineering*, Vol.16, pp. 1424–1440 (2004).
6) Wang, J., Asanuma, Y., Kodama, E., Takata, T. and Li, J.: Mining Sequential Patterns More Efficiently by Reducing the Cost of Scanning Sequence Databases, *IPSJ Digital Courier*, Vol.2, pp.768–782 (2006).
7) Han, J., Pei, J. and Yin, Y.: Mining frequent patterns without candidate generation, *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, SIGMOD '00, New York, NY, USA, ACM, pp.1–12 (2000).
8) Ghoting, A., Buehrer, G., Parthasarathy, S., Kim, D., Nguyen, A., Chen, Y.-K. and Dubey, P.: Cache-conscious frequent pattern mining on a modern processor, *Proceedings of the 31st international conference on Very large data bases*, VLDB '05, VLDB Endowment, pp.577–588 (2005).

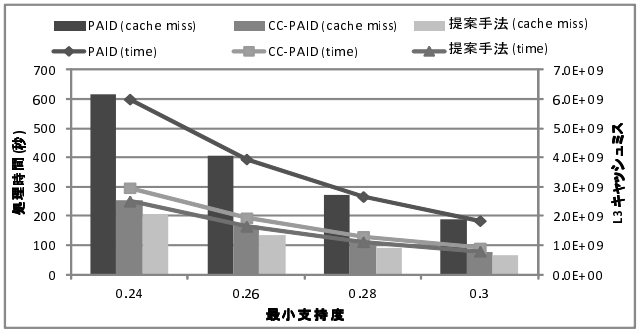

図 13　データセット D_1 利用時の処理時間と L3 キャッシュミス
Fig. 13　Execution time and L3 cache misses using D_1.

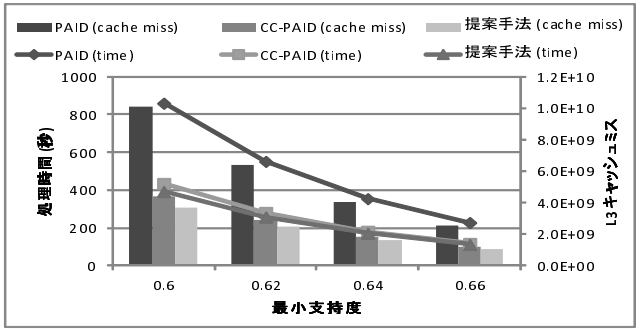

図 14　データセット D_2 利用時の処理時間と L3 キャッシュミス
Fig. 14　Execution time and L3 cache misses using D_2.

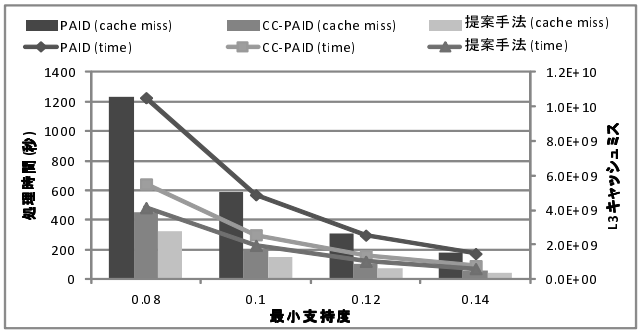

図 15　データセット D_3 利用時の処理時間と L3 キャッシュミス
Fig. 15　Execution time and L3 cache misses using D_3.