並列 Modified PrefixSpan 法の設計と実装

Design and Implementation of Parallel Modified PrefixSpan Method

周藤 俊秀♥ 田村 慶一♦
森 康真♦ 北上 始♦

Toshihide SUTOU Keiichi TAMURA
Yasuma MORI Hajime KITAKAMI

本論文では，配列データベースから頻出パターンを抽出するために，ネットワークで接続された複数台の計算機を用いて，既に著者らによって開発されている Modified PrefixSpan 法を並列化する方式について提案している．本方式では，ソケット通信と MPI ライブラリを用いて，複数台の計算機間の通信を行っている．また，マスタプロセスが複数のスレーブプロセスと送受信するために１つのマスタプロセスの中にマルチスレッドを用意している．マスタプロセスでは，初期段階で生成された部分木の集合を管理するグローバルジョブプールとスレーブプロセスの制御を行っている．実験結果として，8 台で 1 台に比べて 6 倍程度の性能向上が得られた．

The parallelization of a Modified PrefixSpan method is proposed in this paper. The Modified PrefixSpan method is used to extract the frequent pattern from a sequence database. This system developed by authors requires the use of multiple computers connected in local area network. This system is achieved through communication among multiple computers using a socket and an MPI library. It also includes multi-threads to achieve communication between a master process and multiple slave processes. The master process controls both the global job pool, to manage the set of subtrees generated in the initial processing and multiple slave processes. The results obtained here indicated that 8 computers were approximately 6 times faster than 1 computer in trial implementation experiments.

1. はじめに

モチーフは，アミノ酸配列上における特徴的なパターンであり，生物の進化の過程で保存されてきた蛋白質の機能に関係していると考えられている．アミノ酸配列は，「B，J，O，U，X，Z」を除く 20 種類のアルファベットから構成され，ワイルドカードを含む特徴的なパターンを抽出することは文字配列上の頻出パターンを取り出すことになる．しかしながら，マルチプルアライメントや PrefixSpan 法[1]といった既存のパターン抽出アルゴリズムにはいくつかの問題があり，大規模なアミノ酸配列からのモチーフ発見に対して機能的ではなく高速ではない．

我々はこの問題を解決するために， PrefixSpan法にワイルドカード数を制限する仕組みを導入した Modified PrefixSpan法[2]を提案している．Modified PrefixSpan法は，800本程度のアミノ酸配列データ(25～4200文字)に対し，計算時間の短縮や余分なパターンを削除することに成功している．以下では，抽出された頻出パターンの集合の中で k 個のアルファベット文字を含むものを k-頻出パターンと呼ぶ．

本論文では，大規模な配列データベースから頻出パターンを高速に抽出するために，ネットワークで接続された複数台の計算機を用いて，Modified PrefixSpan法を並列化する方式について提案する．並列化の特徴は Modified PrefixSpan 法による頻出パターン抽出処理を複数の k-頻出パターン抽出処理の仕事に分けることを行ったことである．並列処理を管理する１つの計算機が複数の k-頻出パターン抽出処理に全体の抽出処理を分割し，１つ１つ並列処理を行う計算機に配る．Modified PrefixSpan 法の特徴として複数に分けた k-頻出パターン抽出処理の負荷はそれぞれ異なる．早く k-頻出パターンの抽出を終えた計算機は，次の抽出処理を行う．ある計算機が負荷の重い処理を処理している間に他の計算機が次々に処理を続行することで動的な負荷分散を行う．

k-頻出パターン抽出を基とした Modified PrefixSpan 法を実際の PC クラスタ上に実装を行った．動的な負荷分散機構をもつ本方式では，ソケット通信と MPIライブラリを用いて複数台の計算機間の通信を行っている．また，各スレーブプロセスと送受信するために１つのマスタプロセスの中にマルチスレッドを用意している．実験結果として，8 台で 1 台に比べて 6 倍程度の性能向上が得られた．検証実験により効果的な並列効果が確認された．

本論文の構成は以下の通りである．2 章ではModified PrefixSpan法の説明をする．3 章ではModified PrefixSpan法の並列処理の詳細設計について述べる．4 章では検証実験を行い，その結果を述べる．5 章で関連研究について述べ，6 章でまとめる．

2. Modified PrefixSpan 法

Modified PrefixSpan 法は支持率とワイルドカード数を決める．支持率とは，ある頻出パターンがデータベース中の配列のうち，何本の配列に存在するかを示す割合である．既存の PrefixSpan 法はワイルドカードを含む頻出パターンを抽出することができない．例えば，AxxxB とAxxxxB (x は任意のアミノ酸の1文字を表す)は，PrefixSpan 法では AB というパターンとみなす．しかし，Modified PrefixSpan 法では文字間のワイルドカード数を指定することで，このふたつのパターンを区別する．このふたつのパターンはそれぞれ，A3B，A4B (各数字はワイルドカード数を表す)と表現する．

PrefixSpan 法は配列の中の k-頻出パターンから (k+1)-頻出パターンを生成する．(k+1)-頻出パターンの最後の文字は，k-頻出パターンの最後の文字の次の位置から，配列の最後の位置までの長さの文字列を頻出パターン抽出の対象とする．一方，Modified PrefixSpan 法は k-頻出パターンの最後の文字の次の位置からワイルドカード数＋1 の先までの長さの文字列を頻出パターン抽出の対象とするため，配列データが長いほど PrefixSpan 法より高速化が期待できる．

Modified PrefixSpan 法は，与えられた文字列データの集

♥ 学生会員 広島市立大学大学院情報科学研究科博士前期課程 tosihide@db.its.hiroshima-cu.ac.jp
♦ 正会員 広島市立大学情報科学部 {ktamura, mori, kitakami}@its.hiroshima-cu.ac.jp

合に対してまず，支持率を満たす 1-頻出パターンを求める．次に求めた 1-頻出パターンをそれぞれ 1 文字目とし，2-頻出パターンを求める．つまり，k-頻出パターンから $(k+1)$-頻出パターンを抽出する$(1 \leqq k)$．例として，表 1 に示すふたつの配列に対して，支持率を 100，ワイルドカード数を 3 とした場合に現れる頻出パターンを図 1 に示す．

表 1　配列データ
Table 1　Set of Sequences

配列 ID	配列
1	MFKALRTIPVILNMNKDSKLCPN
2	MSPNPTNHTGKTLR

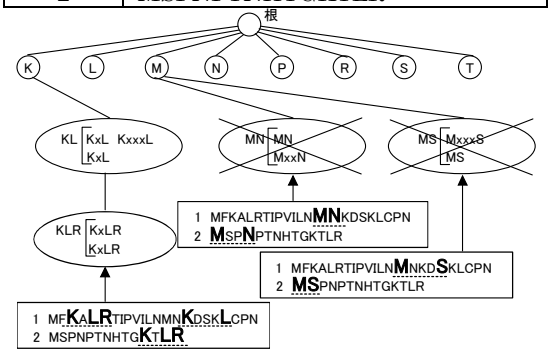

図 1　Modified PrefixSpan 法の頻出パターン抽出

Fig.1　Extraction of Frequent Pattern of Modified PrefixSpan Method

Modified PrefixSpan 法は木構造で表すことができる．図 1 では，まず支持率を満たした「K，L，M，N，P，R，S，T」 が頻出パターンとみなされる．次に，抽出された 1-頻出パターンから 2-頻出パターンの抽出を行う．図 1 に示されるように，例えば 1-頻出パターンが K の場合，2-頻出パターンは KxL となる．これは，ふたつの配列に同じワイルドカード数の文字があるために，K1L という 2-頻出パターンとして抽出される．次に，2-頻出パターン KxL から，3-頻出パターン KxLR を抽出する．この場合，ワイルドカード数を 0 とみなすことで，同じワイルドカード数となり 3-頻出パターン K1LR (ワイルドカード数 0 は頻出パターンには表記しない)として抽出される．しかし，1-頻出パターンが M の場合, 図 1 では 2-頻出パターンとして MN と MxxN が書かれているが，これはふたつの配列のワイルドカード数が等しくないために 2-頻出パターン MN とはみなされない．このように，Modified PrefixSpan 法は，次々に枝分かれをして頻出パターンの抽出が行われている．

3. 並列 Modified PrefixSpan 法

3.1 並列頻出パターン抽出

Modified PrefixSpan 法の並列化では，あらかじめ利用者が指定した閾値としての深さ k までの頻出パターンの抽出を行い，それによって得られる複数の k-頻出パターンの処理を各プロセスが受け取り，探索を続けるという手法を用いる．Modified PrefixSpan 法は 1 文字目のパターンをルートノードの子ノードとする木構造の探索の処理とみなすことができ，容易に複数の部分頻出パターン抽出処理に分け，部分頻出パターン抽出処理は他の抽出処理とは独立して処理を行うことができる．

図 2 に閾値を 2 とし，プロセス数を 4 とした場合を図示する．なお，以降では，複数の部分頻出パターン抽出処理(各プロセッサが探索する部分木)をジョブと呼ぶ．

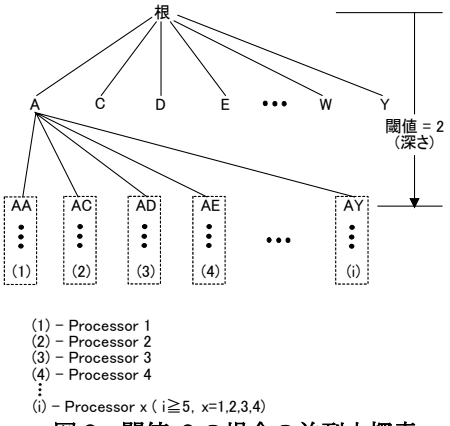

図 2　閾値 2 の場合の並列木探索

Fig.2　Parallel Tree Search for Threshold 2

図 2 の場合，閾値が 2 なので，アミノ酸のアルファベットの組み合わせが最大で 20 文字×20 文字の 400 個のジョブが現れる．このジョブを例えば 4 台の計算機で抽出すると，各計算機は平均で 100 個のジョブを行うことになる．

3.2 設計と実装

ジョブを生成し管理するプロセスをマスタプロセスとした．このプロセスは，ジョブを動的に管理するためにマルチスレッドを用意している．また，マスタプロセス以外のプロセスはスレーブプロセスと呼び，Modified PrefixSpan 法による頻出パターンの抽出処理をおこなう．各スレーブプロセスはマスタプロセスの各スレッドと通信を行う．複数の計算機の管理には MPI ライブラリを利用し，マスタプロセスとスレーブプロセスの通信にはソケットを利用した．マスタプロセスとスレーブプロセスとの通信とマスタプロセスからのジョブの取り出しは並列に行われる．並列 Modified PrefixSpan 法の処理フローを図 3 に示す．

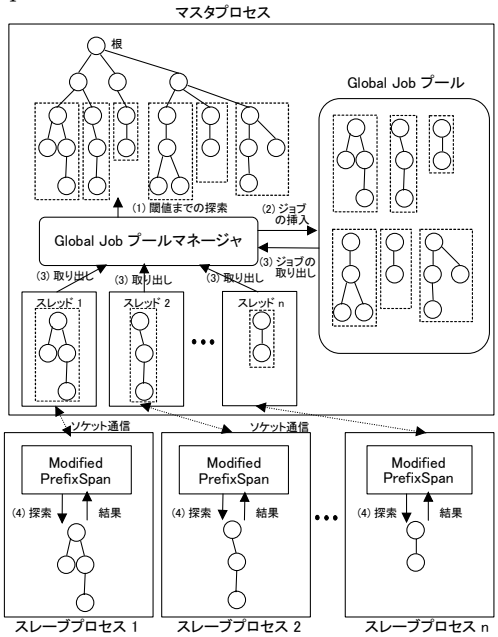

図 3　並列 Modified PrefixSpan 法の処理フロー

Fig.3　Processing Flow of the Parallel Modified PrefixSpan Method

図 3 の処理のステップは以下の順でおこなわれる．

(1)：マスタプロセスで閾値までの頻出パターン抽出を行う．
(2)：(1)で得られる複数のジョブを Global Job プールに挿入する．
(3)：Global Job プールにあるジョブをスレーブプロセスの数だけ作り出したスレッドが取り出し，対応するスレーブプロセスにソケット通信によるデータ送信をする．
(4)：各スレーブプロセスは，送られたデータから Modified PrefixSpan 法による頻出パターンの抽出を行う．
(5)：スレッドは，スレーブプロセスの計算の終了を待ち，終了状態を確認すると(2)以降を繰り返す．

4. 評価

3 章で示した詳細設計をもとに，実際の PC クラスタ上に実装をおこない検証実験をおこなった．

まず，PC クラスタの構成について記述する．本研究で構成したクラスタマシンは，CPU が PentiumIII 450MH，メモリが 128 MB 搭載する 8 台の計算機を 100Mbps イーサネットスイッチングハブで接続した．各計算機の OS は Redhat Linux 8.0，コンパイラは Intel C/C++ complier 6.0 for linux，MPI は mpich-1.2.4 を使用した．

実験に使用した配列データについて記述する．本実験に使用したアルファベットの文字列は，Kringle というモチーフを含む配列データ（データ件数：70，総長：23385 byte，平均長：334 byte)と，Zinc Finger というモチーフを含む配列データ（データ件数：467，総長：245595 byte，平均長：525 byte)である．これらの配列データは PROSITE[3]が提供している．本実験は検証のためデータ件数は少なくしている．

まず，(1) Kringle において閾値を 2 ，支持率を 40 ，ワイルドカード数を 7 とした場合に，(2) Zinc Finger において閾値を 2 ，支持率を 40 ，ワイルドカード数を 5 とした場合に，それぞれの場合において計算機の台数を 2 から 8 に増やしていくことで，どの程度の性能向上比が得られているかを図 4 に示す．

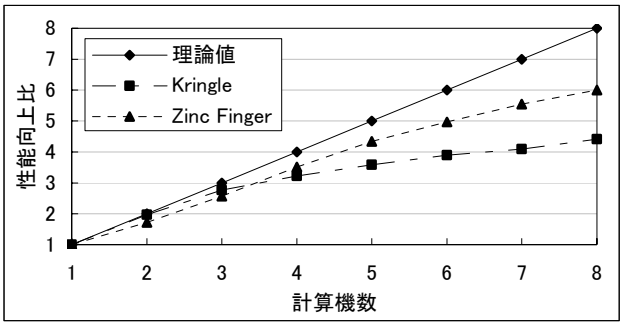

図 4　性能向上比

Fig.4　Performance Ratio

効率の良い並列処理を行うために，No.1 となるスレーブプロセスの計算機には同時にマスタプロセスも実行する．つまり，計算機を N 台使用する場合，N+1 のプロセスを実行することになる．結果として，すべての計算機でジョブから頻出パターンを抽出することができる．

これらの結果より最大で，計算機数 8 のときは約 6 倍の性能向上比が得られた．効率のよい並列処理とは，N 台の計算機で並列にした場合，実行時間を $1/N$ に短縮することである．本実験での性能向上比は配列データによって異なっている．この差の原因として通信のオーバーヘッドが考えられる．この理論値と離れていることを説明するために，Kringle の結果において，計算機の台数が 8 のときの各スレーブプロセスの処理時間(通信時間は含まない)と抽出パターン数を図 5 に，また通信時間と総合受信データ量を図 6 に示す．

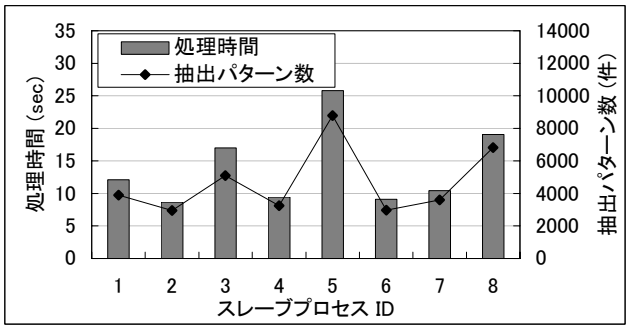

図 5　処理時間と抽出パターン数

Fig.5　Computaion Time and Number of Patterns

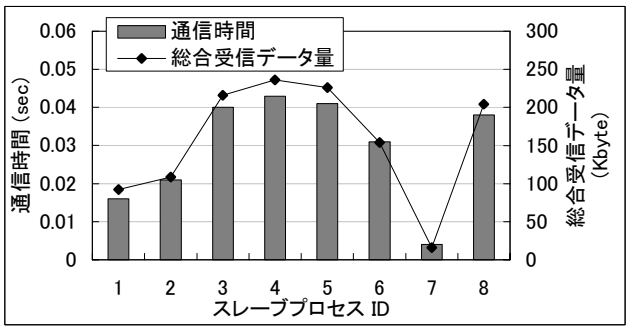

図 6　通信時間と総合受信データ量

Fig.6　Communication Time and Total Received Data

各プロセスの実行時間とは，処理時間に通信時間を足したものである．図 5，6 の結果をみると，実行時間のほとんどが処理時間に相当し，通信時間はほとんどかかっていないことがわかる．これは Zinc Finger においても同様の結果が得られ，通信のオーバーヘッドによる理論値との違いは考えられない．

図 5 の結果により処理時間と抽出パターン数は比例していることがわかる．また，図 6 の結果により通信時間と総合受信データ量は比例していることがわかる．しかし，これらの結果をみると処理時間と通信時間は比例していないことがわかる．つまり，受信データ量にかかわらず処理時間が非常に長い場合がある(例えば，図 5，6 の ID が 4 と 5 のスレーブプロセス)．これは，Zinc Finger においても同様の結果が得られた．各スレーブプロセスの実行時間に差が生じていることより負荷が分散されていないことがわかる．図 4 での配列によって性能向上比が違うのは，Zinc Finger の結果は Kringle の結果よりもスレーブプロセスの処理時間に差がなかったことである．

それぞれのスレーブプロセスの処理時間の違いは，ジョブに原因があると考えられる．受信するデータ量と抽出される頻出パターン数に関連が見られないために，各ジョブに続く頻出パターン数，つまり Modified PrefixSpan 法による木の探索する深さが非常に異なっている．極端な例として考えられるのは，400 個のジョブを生成したとして，そのうち 399

個のジョブはほとんど抽出が続かず，残り 1 個のジョブは次々に抽出が続く場合がある．この場合，結果として，このジョブを受け取ったプロセスは，他のプロセスに比べ極端に計算時間が遅くなってしまう．これを解決する方法として，このジョブをさらに細かく分けることがあげられるが，Modified PrefixSpan 法はジョブを生成した時点では抽出にどのくらいの時間がかかるかわからない．これは，ジョブに続く頻出パターンの数が一定でないので，ジョブの負荷について予測することは困難であるためである．

自動的に最適な負荷分散を実現するために，さまざまな方式について研究する必要がある．

5. 関連研究

トランザクションデータベースから頻出アイテム集合を抽出するアルゴリズムの研究は Agrawal らの研究[4]に始まり，これに関して多くの逐次アルゴリズムが提案されている．これを並列化する研究では，複数の計算機をネットワークで接続した分散メモリ構成の並列マシンが用いられている[5],[6]．一方，配列データベースから頻出パターンを抽出するための逐次アルゴリズムの研究では，GSP[7]，SPADE[8]，PrefixSpan 法などが提案されている．分散メモリ構成[9],[10]や共有メモリ構成[8]の並列マシン上で並列化する研究も一部進められている．これらのアルゴリズムは，顧客の時系列的な購入パターンやネットワークの警報パターンを抽出することを目的とするビジネス分野を想定しているのに対し，著者らが開発した Modified PrefixSpan 法は生命情報科学の分野[11]を想定している．本論文では，この Modified PrefixSpan 法を分散メモリ構成の並列マシン上で並列化する方法[12]を提案し，複数の配列から，配列中のさまざまな位置にある頻出パターンを高速に抽出することができる．

6. おわりに

本論文では，並列 Modified PrefixSpan 法を実現するために，頻出パターン抽出を複数の小さなジョブにわけ，部分的に頻出パターンを抽出する方法について説明した．実際の PC クラスタ上に実装をおこない検証実験をおこない台数効果があることを確認した．また実験で明らかになったのは，極端に重いジョブがある場合には負荷を均一にできないことであった．今後，さらに大きなデータを対象に，さらに効率のよい並列 Modified PrefixSpan 法について検討をおこなっていく予定である．

[謝辞]

本研究について有益なコメントや議論をしていただいた，広島市立大学情報科学部 黒木進 助教授に感謝いたします．なお，本研究の一部は広島市立大学・特定研究費(一般研究費(コード番号：3106))の支援により行われた．

[文献]

[1] Jian Pei, Jiawei Han, Behzad Mortazavi-Asl, and Helen Pinto: PrefixSpan: Mining Sequential Patterns Efficiently by Prefix-Projected Pattern Growth, Proceedings of the 7th International Conference on Data Engineering, IEEE Computer Society Press, pp.215-224, 2001.

[2] Hajime Kitakami, Tomoki Kanbara, Yasuma Mori, Susumu Kuroki, and Yukiko Yamazaki: Modified PrefixSpan Method for Motif Discovery in Sequence Databases, Proc. of PRICAI2002, pp.482-491, Springer-Verlag, August 2002.

[3] PROSITE home page, http://aulexpasy.org/prosite/

[4] Rakesh Agrawal et al.: Fast Discovery of Association Rules, Advances in Knowledge Discovery and Data Mining , AAAI Press / MIT Press, pp.307-328, 1996.

[5] Rakesh Agrawal and John Christopher Shafer: Parallel Mining of Association Rules, IEEE Trans. On Knowledge and Data Engineering, Vol. 8, No. 6, pp.962-969, 1996.

[6] Takahiko Shintani and Masaru Kitsuregawa: Parallel Mining Algorithm for Generalized Association Rules with Classification Hierarchy, SIGMOD'98, pp.25-36, ACM, 1998.

[7] Ramakrishnan Srikant and Rekesh Agrawal: Mining Sequential Patterns: Generalization and Performance Improvements, Proc. of the 5th International Conference on Extending Database Technology, March 1996.

[8] Mohammed Javeed Zaki: Parallel Sequence Mining on Shared-Memory Machines, Journal of Parallel and Distributed Computing, Vol. 61, Academic Press, pp.401-426, 2001.

[9] Takahiko Shintani and Masaru Kitsuregawa: Mining Algorithms for Sequential Patterns in Parallel: Hash based Approach, Proc. of the 2nd Pacific-Asia Conference on Knowledge Discovery and Data Mining, Springer-Verlag, Vol. 1394, pp.283-294, 1998.

[10] David Wai-lok Cheung and Yongqiao Xiao: Effect of Data Distribution in Parallel Mining of Associations, Data Mining and Knowledge Discovery, Vol.3, Kluwer Academic Publishers, pp.291-314, 1999.

[11] 五條堀孝 編: 生命情報科学，シュプリンガー・フェアラーク・東京，2003.

[12] Toshihide Sutou, Keiichi Tamura, Yasuma Mori, and Hajime Kitakami: Design and Implementation of Parallel Modified PrefixSpan Method, Proc. of ISHPC-V, LNCS, Vol.2858, Springer-Verlag, 2003.

周藤 俊秀 Toshihide SUTOU
広島市立大学院情報科学研究科博士前期課程在学中．2002 広島市立大学情報科学部卒業．日本データベース学会学生会員．

田村 慶一 Keiichi TAMURA
広島市立大学情報科学部助手．2000 九州大学大学院システム情報科学研究科修士課程修了．修士（工学）．並列データベースシステムの研究に従事．情報処理学会，日本データベース学会，IEEE CS 各会員．

森 康真 Yasuma MORI
広島市立大学情報科学部助手．1994 北陸先端科学技術大学院大学情報科学研究科博士前期課程修了．データベースシステムの研究・開発に従事．情報処理学会会員．日本データベース学会会員．

北上 始 Hajime KITAKAMI
広島市立大学情報科学部教授．1976 東北大学大学院工学研究科博士前期課程修了．博士（工学）．データベースシステムの研究・開発に従事．情報処理学会一般情報処理教育小委員会委員．情報処理学会 25 周年記念論文．日本データベース学会ビジネスインテリジェンス研究会運営委員．