SMP 上における再帰 BLAS ライブラリの自動チューニング方式

木 下 靖 夫[†] 片 桐 孝 洋[†,††] 弓 場 敏 嗣[†]

数値計算ライブラリを高速に実行するには，キャッシュの大きさやプロセッサ台数などのパラメタ設定をする必要がある．そこで，使用する計算機環境に応じてこれらの設定を自動的に行う自動チューニングソフトウェアが開発されてきた．現在利用されている自動チューニングソフトウェアとして ATLAS (Automatically Tuned Linear Algebra Software) が知られている．本研究では，SMP 型の並列計算機に向くように ATLAS を用いて BLAS を再帰実装し，かつ Posix thread を用いて実装することにより高速化を狙う．我々は，この方式を AutoTuned-RB と呼んでいる．AutoTuned-RB では，再帰段数に関するパラメタチューニングを実行することで自動チューニングを行っている．性能評価の結果，提案手法は SGI Origin（CPU16 台）において，ピーク性能に対し約 90%の効率，および ATLAS の実測性能に対し最大で 3.3 倍の効率を実現できることを確認した．

An Auto-Tuning Method for Recursive BLAS Libraries on SMP Machines

Yasuo Kinoshita[†] Takahiro Katagiri[†,††] Toshitsugu Yuba[†]

In order to perform a numerical computation library at high speed, it is necessary to optimize parameters according to the size of cache, the number of processors, etc. For this reason, the auto-tuning software, which optimizes these parameters according to computer environments, has been developed. For example, ATLAS (Automatically Tuned Linear Algebra Software) is well-known auto-tuning software. In this research, BLAS is implemented as a recursive manner using ATLAS, and is parallelized to archive high performance in SMP type parallel machines. We call this method AutoTuned-RB -- it can optimize the parameter for the number of recursive levels. As a result of performance evaluation, about 90% of efficiency to the peak performance in 16 CPUs on the SGI Origin, and the maximum speedup factor of 3.3 times to the performance of ATLAS, are obtained.

１． はじめに

大規模な行列演算を必要とする科学技術計算において，高性能な数値計算ライブラリの使用は非常に有効な手段である．もしライブラリをユーザが計算カーネルとして使用しない場合，高度なコーディング技術と実装知識が必要とされ，これは非常に困難である．ところが現在，数値計算ライブラリにおいては計算機を開発しているメーカが提供している基本線型代数計算副プログラム BLAS (Basic Linear Algebra Subprograms)がある．BLAS を用いれば，簡易に，かつ高性能計算をすることができる．

ところが，提供される BLAS などの数値計算ライブラリのなかには，性能に関するパラメタの設定を必要とするものがある．このパラメタの設定を誤ると計算性能が劣化する．これを回避するため，最適なパラメタを設定する必要がある．このパラメタ設定を自動的に行うのが，数値計算ライブラリにおける自動チューニングである．

数値計算ライブラリにおける自動チューニングには，主に以下の２つの型がある．まず，ライブラリをインストールする時に行うインストール時型である．もう一方は，ライブラリを実行する時に行う実行時型である．これらの型を利用した自動チューニングソフトウェアには PHiPAC[5], ATLAS[2], FFTW[3], および I-Lib[4]などが知られている．また片桐らの FIBER[6]は，上記２種のチューニング型に加え，ユーザによるパラメタ設定時にチューニングを行う実行起動前型という型を導入した．

ところが，これら自動チューニングソフトウェアのパラメタチューニングは，インストールされる条件（指定される問題サイズ）上では高速化がなされるが，それ以外の局所的な問題サイズでは性能が劣

† 電気通信大学大学院情報システム学研究科
Graduate School of Information Systems,
The University of Electro-Communications
†† 科学技術振興機構さきがけ
PRESTO, JST

化する可能性がある．さらに，最適化された性能に再現性があり，かつ維持されるという安定性が必要である．

そこで我々は, まず高速化を達成するため, ATLAS に注目した．つぎに，チューニング対象の BLAS を再帰化して実装する方式 [1]を採用した. 再帰の実装において階層メモリ型を考慮した再帰段数指定方式を提案することにより，性能の安定性を保証する．また, SMP 型の並列計算機上で高性能を実現するため, Posix thread を用いて並列化する．この方式を，我々は AutoTuned-RB (Auto-Tuning with Recursive BLAS) と呼ぶ.

本稿の構成を以下に示す．２章で，行列-行列積演算における BLAS の再帰化アルゴリズムをまとめる．３章では, 提案する AutoTuned-RB の自動チューニング手法と，そのインストールの流れを説明する．４章では提案する AutoTuned-RB の性能評価をし, さらに ATLAS BLAS との性能を評価する. 最後に５章で, まとめを行う.

2． 再帰 BLAS

2.1 概要

Gustavson らが提案した再帰 BLAS[1]は, BLAS の演算を再帰により部分小行列に分割し，その分割された部分小行列単位で演算をすることにより，データ参照の局所化（キャッシュブロッキング）を促進する手法である．すなわち，階層メモリを有する計算機上で高速計算を行うことを目的とする手法である. なお，再帰を用いることで，コレスキー分解等の線形代数演算も再帰実装可能である.

2.2 GEMM

GEMM は，行列‐行列演算(レベル 3 BLAS)を行うライブラリ･インターフェースの一つである．具体的には行列積において，C←αop(A)op(B)+βC の計算を実行する．ここで op(A)において，行列 A，転置 A^T, 複素共役 A^H の選択が可能である．また α，β は int 型のスカラであり，行列積の係数を表している.

2.3 再帰 DGEMM

DGEMM の頭文字 D は，行列のデータ型が double 型を使用していることを表しており，行列積を行う関数である．本稿で示す再帰 BLAS の処理は，２分木構造で表現できる．これを，**図** 1 に示す．**図** 1 では, 各節はタスクを表している. 木の根においては，１つのタスクから２つのタスクに分かれる．木の葉に向かうにつれて，各々のタスクが２つのタスクに分割されていくので, 末端葉の数は 2^L(L：再帰段数)となる．ここで，末端葉部で BLAS をコールする．たとえば，**図** 1 中の処理①のあと，②が実行され，その処理のあと，再び末端葉の BLAS コール③がなされる.

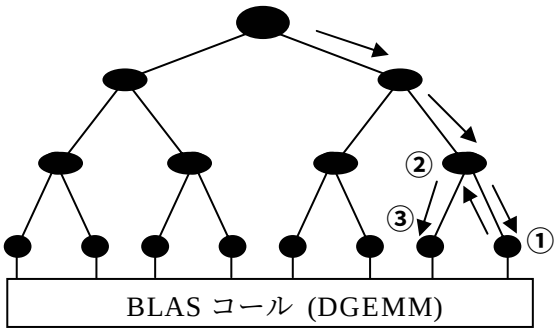

図 1　再帰 DGEMM の処理

```
function [C(1:M,1:N)] = RB-GEMM(C(1:M,1:N),
                          A(1:M,1:K),B(1:K,1:N),m,n,k)
if m = 1 and n = 1 and k = 1 then
    DGEMM(C(1:M,1:N),A(1:M,1:K),B(1:K,1:N));
else if m = max (m,n,k) then
    m1 = m/2;      m2 = m - m1;
    RB-GEMM(C(1:m1･mb,1:N),
            A(1:m1･mb,1:K),B(1:K,1:N),m1,n,k);
    RB-GEMM(C(m1･mb+1:M,1:N),
          A(m1･mb+1:M,1:K),B(1:K,1:N),m2,n,k);
else if n = max (m,n,k) then
    n1 = n/2;      n2 = n - n1;
    RB-GEMM(C(1:M,1:n1･nb),A(1:M,1:K),
            B(1:K,1: n1･nb),m,n1,k);
    RB-GEMM(C(1:M, n1･nb+1:N),A(1:M,1:K),
            B(1:K, n1･nb+1:N),m,n2,k);
else
    k1 = k/2;      k2 = k - k1;
    RB-GEMM(C(1:M,1:N),A(1:M,1:k1･kb),
            B(1: k1･kb,1:N),m,n,k1);
    RB-GEMM(C(1:M,1:N),A(1:M, k1･kb+1:K),
            B(k1･kb+1:K,1:N),m,n,k2);
end
```

図2　Gustavson再帰BLASのアルゴリズム

図 2 に，再帰 DGEMM の概略を示す．**図** 2 より，C：M×N，A：M×K，B：K×N の各行列について，再帰分割を制御するパラメタは n，m および k である．それぞれ，以下の３つの工程で分割される：(１) n は A，B 行で２分割；(２) m は B，C の列で２分割；(３)

k は A 行列の列，B 行列の行を２分割；ここで，末端葉の総数は 2^L= n・m・k である．

具体的には, (１)n=max(m,n,k)の時に n1 = n / 2 とし，この分割行列２つを再帰呼び出しする；(２)m=max(m,n,k)の時に m1 = m / 2 とし，この分割行列２つを再帰呼び出しする；(３)(１)，（２）以外の時の時に k1 = k / 2 とし，分割行列２つを再帰呼び出しする；ここで，max(a,b,c)は，a，b，c の中で最大値を返す関数である．また，mb= M/m, nb=N/n, kb= K/k で，再帰葉における行列サイズに対応している．

再帰の終了条件は，上記の分割処理より n=m=k=1 になった時であり，この時 DGEMM をコールする．この再帰葉部の計算が終了すると，親節に戻るが，この親節では，再び再帰呼び出しがなされる．ここで注意すべきは，mb，nb，kb で指定される行列のデータ量が，L1 キャッシュサイズを超えないように再帰段数を取ることである．

３． 再帰 BLAS による自動チューニング

3.1 ATLAS における問題点

ATLAS での最適化手法（パラメタ決定手法）は，ライブラリインストール時に物理的な L1, L2 キャッシュサイズを参照することによって，推定される最適なブロックサイズを決定する．この推定には，AEOS[2]という経験的手法の枠組みが利用される．具体的には，決められたブロックサイズを微小に増減させ実行時間を測定し，最も早くなるパラメタ値を推定する．そのパラメタ値に基づき，ATLAS BLAS を生成する．しかし局所的な行列の次元では性能が劣化するという問題がある．

また ATLAS は自動並列化機能をサポートしているが，これはライブラリのソースコードをコンパイルする際に，コンパイラによる自動並列化を利用するものである．したがって，ライブラリのソースコード特性や使用するコンパイラのコード最適化手法に依存するため，広範な並列計算機環境における適用性は低い．

3.2 設計方針

このような問題を解決するため，まず提案する AutoTuned-RB でのチューニングのタイミングは ATLAS と同様にライブラリインストール時に行うこととする．

つぎに，コンパイラ最適化による依存性を除去するため，再帰 BLAS ライブラリを Posix thread (以下 Pthread) によって並列化する．このことにより，広範な並列計算機環境でも高性能を達成できることが期待できる．

3.3 実装の詳細

図 2 中の再帰関数 **RB-GEMM** のインターフェースでは，行列 A，B，C を分割して各行列の先頭アドレスを引数としている．Pthread のスレッド生成関数では引数を１つしか取ることが出来ないので，AutoTuned-RB の再帰関数では，グローバル構造体で行列データ A，B，C で定義して，その構造体を引数として引き渡す．また，再帰関数内で n=m=k=1 の時 **pthread_create** 関数を用いて，スレッド内で ATLAS BLAS をコールし並列計算を行う．

3.4 パラメタ決定手法

3.4.1 提案手法の概略

再帰 BLAS でチューニングするパラメタは，再帰の段数である．パラメタの決定方式は２つの方式が知られている．

- **全探索方式**：再帰段数 L を L=1，2，3，4，…と順に測定して行き，最も性能が出るものを採用する．
- **ヒューリスティック方式**：全探索方式では確実にピーク性能が出る再帰段数を求めることが出来る．しかし段数が大きくなると，計算量が増えるのでインストール時間が膨大になる．それに対してこの方式では，経験的にピークに近い値を達成できるようなサンプリング点を選ぶ．そのことで，パラメタ決定時間が高速化される．

本稿では，ヒューリスティック方式により再帰段数を決定する方式を採用する. 具体的には, 行列 A, B，C の合計サイズが L1，L3 キャッシュサイズを超えないような再帰段数 L を決定する. 次に L-2, L-1, L，L+1，L+2 段で速度測定を行い，最も速くなる段数を決定する．

また，行列サイズを固定して実行時間を測定し，それをもとにして最適化をすると，行列サイズが大きい，または小さい場合に，本来ピークとなるべき再帰段数が推定できない可能性がある．そのため、本提案方式では，測定する行列サイズを複数用意する．各々の測定結果から，行列サイズに関するサンプリング点におけるピーク値を取る再帰段数を２次多項式関数化する．このことで，行列サイズに依存する再帰段数を切り分ける．

3.4.2 行列サイズによる再帰段数決定モデル

図 3 は，本稿で仮定する行列サイズ依存の再帰段

数決定モデルのグラフである．横軸が行列サイズで，縦軸がピーク性能達成時の再帰段数である．

行列サイズに関するサンプリング点 s_S, s_M, s_L は， AutoTuned-RB でチューニングする際に計測した行列サイズである．このサイズは概述の通り，複数設定するが，以下の基準で設定する。

- サンプリング点 s_S：行列 A，B，C の合計サイズが, L1 キャッシュサイズより小さくなる行列サイズのうち最も大きいもの
- サンプリング点 s_M：行列 A，B，C の合計サイズが, L3 キャッシュサイズより小さくなる行列サイズのうち最も大きいもの
- サンプリング点 s_L：s_M サイズより大きい問題サイズのうち適当な値

これらのサンプリング点では，最も高速となる再帰段数を、実測を基に決定する．

いま，サンプリング点間のパラメタは，２次多項式 f(N)=a+bN で推定する．ここで，0～サンプリング点 s_S までの推定関数を f_S(N), サンプリング点 s_S よりおおきな値～サンプリング点 s_M までの推定関数を f_M(N), サンプリング点 s_M よりおおきな値～サンプリング点 s_L までの推定関数を f_L(N)とする．

いま, D_1, D_2 は, 実際にチューンした再帰 BLAS を使用する際にユーザが指定した行列サイズとする．それぞれ点 D_1，D_2 で測定する場合，以下のように最適段数を決定する．点 D_1 では，推定する最適な再帰段数は f_M(D_1)で決定されるが，この値に最も近い整数値を推定段数とする．一方 D_2 では，s_L の段数そのものを推定段数とする．

AutoTuned-RB ではこのような２次多項式を用いてモデル化し，行列サイズに応じて適切な再帰段数を切り分ける．

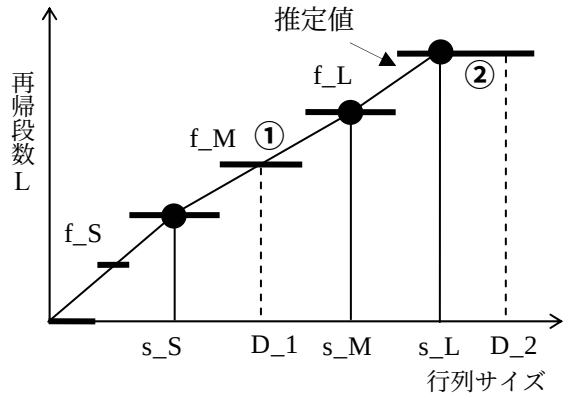

図 3　再帰段数決定モデル（行列依存）

3.5 インストールの流れ

AutoTuned-RB のコンピュータへのインストールの流れ [9]を**図** 4 に示す．

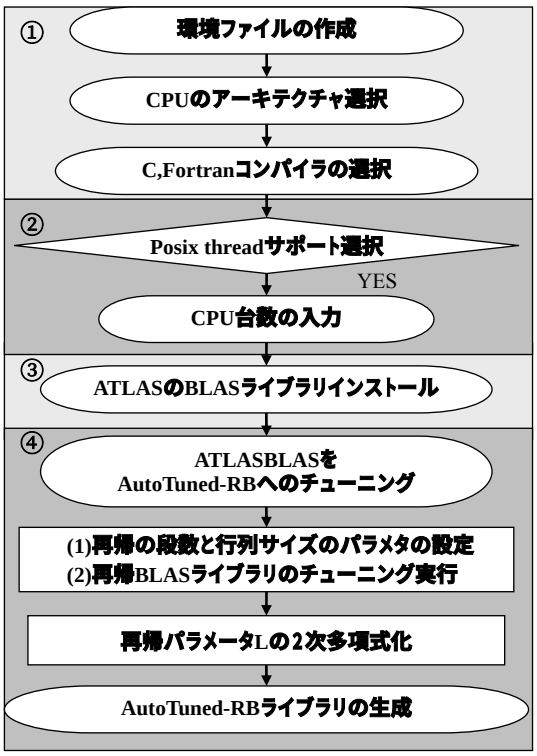

図 4　AutoTuned-RB インストール流れ

図 4 中の①，③は，ATLAS で行っている処理部分である. AutoTuned-RB で新たに組み込んだ機能は②，④の部分である．①～④の説明を以下に示す。

① ATLAS をインストールする過程で最初に行う処理部分である．ここでは，チューニングに必要な L1，L2，L3 キャッシュサイズ，CPU の種類，インストールで用いるコンパイラの種類，およびコンパイラオプションをユーザが指定する．

② AutoTuned-RB で Posix Pthread を使用するか否かをユーザが指定する．使用しない場合は，再帰 BLAS の使用及びチューニングは行わずに③の BLAS チューニングを行いインストールを終了する．Posix Pthread 使用する場合, 計算機環境での CPU 台数をユーザが指定する．

③ ATLAS の BLAS を, ①で作成した環境ファイルを元にキャッシュブロックサイズを決定する．

④ ③でチューニングした BLAS を, SMP 向けにするために，再帰 BLAS をチューニングする．**図** 4 中の(1)では，ヒューリスティック法で定められた再帰段数と計測用の行列のサイズを決定する．その後，サンプルプログラムを用いて再帰 BLAS のチューニングを行う．つぎに，求めた測定データから

再帰段数パラメタ L に関する実行時間を２次多項式モデル化する．

４． 性能評価

4.1 測定環境

この章では，電気通信大学総合情報処理センター所有の SGI Origin3400 による性能評価を行う．SGI Origin3400 のスペックは，OS：IRIX 6.5，CPU：MIPS Technologies R14000 500MHz，理論ピーク性能：1Gflops/1CPU，L1 キャッシュ：32KB(データキャッシュ)，L2 キャッシュ：8MB DDR，フルスピード SDRAM である．測定には CPU16 台，メインメモリ 8GB，コンパイラは IRIX C コンパイラで，オプションは ”-64 -O3” を用いた. ATLAS のバージョンは 3.4 を使用した．測定プログラムは，ATLAS がチューニングを行った DGEMM(以下 DGEMM_P)を利用して再帰化したもの (以下 RDGEMM)を使用した．

4.2 再帰 BLAS の性能

ここでは，各行列の次元 N において，再帰段数 L を変化させた時の Flops 値変化の評価を行う．測定は，**表** 1 の条件で行った．行列は密の正方行列で，各要素は乱数値で生成した．行列の次元は N=100～400 と N=500～5000 で，再帰段数 L の変化の幅を変え，計算カーネルは DGEMM_P をコールしている．

図 5 左から，行列の次元 N＝200～400 では L=5 でピーク値を取っている．しかし，N=100 では L=0 の時が最高速になっている．これは，行列のデータ量が L1 キャッシュ以下になり, 再帰分割がオーバーヘッドとなったためと推定される. そのため L > 1 では遅くなっている．**図** 5 右から，全ての次元 N において段数 L=5 でピーク値を取っている．これは，**図** 5 の N=200～400 と同様の結果である．理由として，DGEMM_P が L1，L2 キャッシュの最適化を行っているため, 再帰分割した行列の大きさが L1 を超えた時点で CPU 台数に適した L を選択できるからと考えられる．

図 6 は，ソースコードをコンパイラによって自動並列化を行った DGEMM_P と AutoTuned-RB との性能比較のグラフである．評価は 16CPU で行った．

表 1　再帰 BLAS の測定条件

測定対象	行列の次元 N	再帰段数 L	計算カーネル
L1 キャッシュ内	100, 200, 300, 400	0,1,2,3,4,5	DGEMM_P
L1 キャッシュ外	500,1000,2000,3000,4000,5000	4,5,6,7,8,9	DGEMM_P

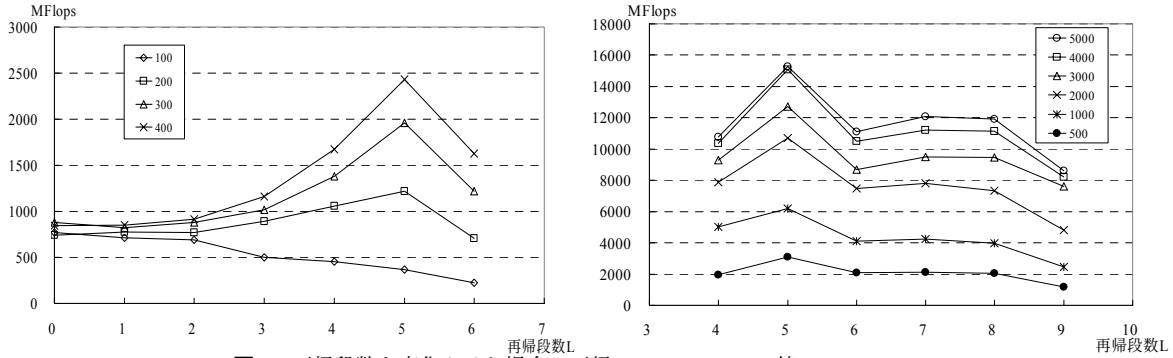

図 5　再帰段数を変化させた場合の再帰 BLAS の MFlops 値(n=100～400，500～5000，16CPU)

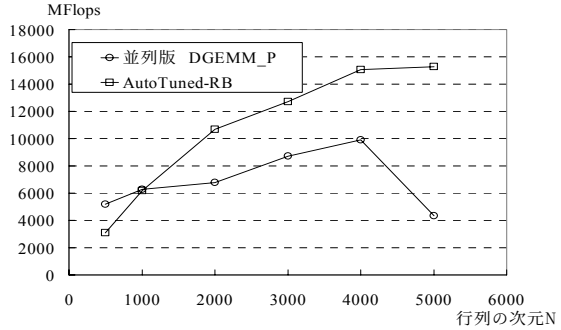

図 6　RDGEMM と並列化した DGEMM_P の Mflops 値 (16CPU)

行列 2000～5000 での AutoTuned-RB は, DGEMM_P に比べて大幅にスピードアップしている．特に次元 N=5000 では，DGEMM_P の性能が劣化している．この結果, DGEMM_P の実行時間に比べ, AutoTuned-RB が約 3.3 倍の性能向上を達成した. 一方, N=100～1000 では，DGEMM_P のほうが性能がよい．これは，L1 キャッシュ以下の時，再帰分割のオーバーヘッドが行列の分割による速度向上に対して顕著になるためと思われる．

1CPU での DGEMM_P では，理論ピーク性能に対

する効率が約 90%であるが，16CPU で N=4000 のときは，約 62%になる．それに対して AutoTuned-RB では，16CPU で N=5000 のとき，ピーク性能に対する効率は約 90%となる．したがって，並列処理効果は十分高い．

4.3 再帰段数パラメタの多項式化

N=100 での多項式は f(N)=1，N=100～200 では f(N)=(1/25)N-3，および N=200～5000 では f(N)=5 という単純な結果になった．これは，Orgin3400 は L3 キャッシュがないので，細かい最適化を行う余地がなかったためだと考えられる．

5．まとめ

本研究では，インストール時型自動チューニングソフトウェア ATLAS が備えている，コンパイラによるソースコード自動並列化による性能を改善するために，再帰 BLAS 方式および Pthead を用いて実装する方式を提案した．また SMP マシンにおいて，CPU 台数と L3 キャッシュサイズに適応するように，行列の次元に依存する再帰段数を決定する自動チューニング手法を提案した．

今回提案したパラメタ決定手法は，(１)キャッシュサイズからヒューリスティックを用いて再帰段数を決定する，(２)決定した再帰段数の値を振らして，サンプルプログラム上での行列次元を変化させ，各次元でのピーク値となる段数を決める，(３)階層メモリを意識し行列次元に対する再帰段数の挙動を多項式化する，という特徴をもつ．

SGI Orgin3400 の 16CPU 上で，ATLAS での DGEMM 性能に対し，AutoTuned-RB は次元 2000～5000 で最大 3.3 倍の速度向上を達成した。また，ピーク性能に対する効率 90%を達成した．これより，十分に AutoTuned-RB の有用性が確認できた．

本稿で述べた，性能安定化の関連研究として，以下が挙げられる．問題サイズが増加したときに，急激に性能が悪化しないという安定性に関する自動チューニング方式を，今村ら [7]が提案している．一方直野ら [8]は，性能劣化が起きた場合，性能保証を考慮した品質に関する自動チューニングの概念を提案している．

今後の課題を以下に列挙する．

- 本性能評価では，L3 キャッシュを搭載していないマシンを使用したために，提案した再帰段数チューニング手法の効果を十分に示すことが出来なかった．したがって，評価環境をさらに増やし，提案方式によるチューニング効果を検証する．
- 再帰 BLAS に比べ，コンパイラによる自動並列化を行った ATLAS の方が性能が良い場合があった．ゆえに，再帰 BLAS と ATLAS BLAS を切り分ける方式が有効となるので，この方式を検討する．さらに，ベンダー提供の BLAS との比較をする．
- 再帰化をすることにより，システムが余分なメモリを確保する．これを回避するために，再帰を for ループ展開し，使用メモリ量を減らす．

参考文献

1) Gustavson, F.， Henriksson, A., Jonsson, I., Kågström, B. and Ling, P. : Recursive Blocked Data Formats and BLAS's for Dense Linear Algebra Algorithms , *PARA98 (Proceedings of the Fourth International Workshop on Applied Parallel Computing)*, Lecture Notes in Computer Science 1541, pp.195-206, Springer (1998).

2) Whaley, R.C., Petitet, A. and Dongarra, J.J.: Automated Empirical Optimizations of Software and the ATLAS project, *Parallel Computing*, Vol.27, pp.3-35(2001).

3) Frigo, M.: A Fast Fourier TransForm Compiler, *Proceedings of the 1999 ACM SIGPLAN Conference on Programming Language Design and Implementation*, Atlanta, Georgia, pp.169-180(1999).

4) 片桐孝洋, 黒田久泰, 大澤清, 工藤誠, 金田康正: 自動チューニング機構が並列数値計算ライブラリに及ぼす効果, 情報処理学会論文誌: ハイパフォーマンスコンピューティングシステム, Vol.42, No.SIG12(HPS4), pp. 60-76 (2001).

5) Bilmes, J., Asanović, K., Chin, C.-W. And Demmel, J.: Optimizing Matrix Multiply using PHiPAC: a Portable, High-Performance, ANSI C Coding Methodology, Proceedings of International Conference on Supercomputing 97, pp. 340-347 (2001).

6) 片桐孝洋, 吉瀬謙二, 本多弘樹, 弓場敏嗣: FIBER : 汎用的な自動チューニング機能の付加を支援するソフトウエア構成方式,情報処理学会研究報告, 2003-HPC-94, pp.1-6(2003).

7) 今村俊幸, 直野健: キャッシュ競合を制御する性能安定化機構内蔵型数値計算ライブラリについて，情報処理学会論文誌: コンピューティングシステム, vol.45, No.SIG 6(ACS 6), pp113-121(2004).

8) 直野健, 今村俊幸, 恵木正史: GRID コンピューティング環境における行列ライブラリ向け性能保証方式の検討, 情報処理学会論文誌, コンピューティングシステム, vol.45, No.SIG 6(ACS 6), pp105-112(2004).

9) 木下靖夫, 片桐孝洋, 本多弘樹, 弓場敏嗣: SMP マシン上での BLAS ライブラリ用自動チューニング機構の設計と実装, 電子情報通信学会総合大会論文集, ソフトウェアサイエンス, p.29 (2004).