ユーザ知識を活用するソフトウエア自動チューニングについて

片桐孝洋[†,††] 吉瀬謙二[†,††]
本多弘樹[†] 弓場敏嗣[†]

本稿では，ソフトウエア自動チューニングの枠組みについて述べる．広範な対象に適用できるソフトウエア自動チューニング機構を実現するには，ユーザ知識の活用が必要不可欠となる．ここでユーザ知識とは，対象部位，アルゴリズム切替え可能情報，速度パラメタ，経験値，コスト定義関数，最適化方式，および実行時情報のことである．ある数値計算ベンチマークを利用した評価の結果，ユーザ知識を活用することで，1.02 倍〜2.23 倍の速度向上が達成できる例を確認した．

A Framework for Software Auto-tuning Using User's Knowledge

Takahiro KATAGIRI,[†,††] Kenji KISE,[†,††] Hiroki HONDA[†]
and Toshitsugu YUBA [†]

In this report, a framework for software auto-tuning is discussed. To implement widely-adapted software auto-tuning facility, user's knowledge is a crucial factor. Target Parts, Algorithm Selection Information, Speed Parameters, Experience Values, Cost Definition Functions, Optimization Methods, and Run-time Information are the user's knowledge in this framework. The result of performance evaluation with a numerical benchmark indicated that the speedup factors from 1.02 to 2.23 were obtained by using the user's knowledge.

1. はじめに

近年，PHiPAC[1)]，ATLAS[2)]，FFTW[3)]，および ILIB[4)] に代表される，いわゆる「自動チューニング機構付き」の数値計算ソフトウエア（以降，自動チューニング機構付きソフトウエア，Software with Auto-Tuning Facilities (SATF) とよぶ）が多数開発されるようになった．現在，SATF の性能評価が多くなされており，その有効性が周知のものとなりつつある[4)]．

これら従来の SATF は，速度の観点のみの性能をチューニングするものであった．この理由は，数値計算ライブラリという限定処理のソフトウエアが対象であったことによる．自動チューニング対象を任意のソフトウエア処理に広げる場合，速度面だけの改善ではない,「品質的な」性能改善も必要となる．たとえば，計算結果の精度保証，計算速度が急激に悪化しないこと，利用可能なメモリ量の制約，などである．このように自動チューニング対象を広げる場合，SATF の構築そのものが困難になることが予想される．なぜなら従来方式が成功した理由は，速度面の改善のみを目的にし，かつ処理を数値計算に限定したので，対象のモデル化が容易に行えたことによる．

本稿では，広範な対象を自動チューニングするためにはユーザ知識の活用が重要となることを述べる．すなわち自動チューニングの対象を広げる場合，ユーザ知識を利用すれば SATF の構築が可能となる．

本稿の構成は以下の通りである．2 章で本稿で対象にするソフトウエア自動チューニングについて述べ，3 章で FIBER 方式の概要について説明し，4 章でユーザ知識に基づく自動チューニングの効果を評価する．最後に知見をまとめる．

2. ソフトウエア自動チューニング

2.1 チューニング処理

図 1 に，本稿で仮定するソフトウエアにおけるチューニング処理を示す． 図 1 では，5 つの処理がある．少なくとも，＜ 4. 最適化＞が自動に行えなければ自動チューニングとはいえない．また，自動化が難しい処理は，＜ 2. パラメタ抽出＞と＜ 3. モデル化＞処理である．

2.2 実現レベル

ここでは，図 1 の処理を SATF として実装する場合の実現レベルについて述べる．

- **原始操作レベル：**図 1 の処理全てを手動で実装する．具体的には，職人による実装がこれにあたる．この段階では，＜ 4. 最適化＞のため特化されたソフトウエアを開発し，半自動化する場合もある．

† 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
†† 科学技術振機構 戦略的創造研究推進事業 さきがけプログラム「情報基盤と利用環境」領域
“Information Infrastructure and Applications”, PRESTO, Japan Science and Technology Agency (JST)

＜1. **対象決定**＞：対象ソフトウエアにおける対象部位 A を決定する
＜2. **パラメタ抽出**＞：対象部位 A の性能に影響を及ぼすパラメタ集合 X を抽出する
＜3. **モデル化**＞：対象部位 A の性能挙動を示す関数 F を，パラメタ集合 X で定義する (この関数を $F(X)$ と書き**コスト定義関数**とよぶ)
＜4. **最適化**＞：関数 $F(X)$ を最小化するようなパラメタ集合 X^* を部位 A を実行しながら見つける (集合 X^* を**解パラメタ集合**とよぶ)
＜5. **対象実行**＞：解パラメタ集合 X^* を用いて対象ソフトウエアを実行する

図 1 ソフトウエアにおけるチューニング処理

しかし，＜4. 最適化＞の完全自動化はできない☆．

- **半自動化レベル：**図 1 の処理のうち，＜4. 最適化＞処理を完全自動化する．かつ，＜2. パラメタ抽出＞および＜3. モデル化＞処理を，一部自動化する．具体的にはユーザが，パラメタ抽出対象，コスト定義関数，および最適化方法を記述できる．このユーザ記述には，システムが自動的に抽出できない，ユーザ知識情報の記述が必要となる．
- **完全自動化レベル：**図 1 の処理すべてを自動化する．すなわち，ユーザによる記述を一切行わずに，対象部位，パラメタ抽出，モデル化，および最適化が実行できる．この段階は，ソフトウエア自動チューニングの究極的な到達点であるが，実現は困難である．

本稿で対象とするのは，＜半自動化レベル＞である．

2.3 最適化コンパイラとの違い

本稿で取り扱う実現レベルには，コンパイラのコード最適化の概念と類似点がある☆☆．しかし，以下の観点から異なる．

- **最適化対象：**コンパイラのコード最適化は，生成コードの実行が高速となるパラメタを抽出し，実行時間の観点でモデル化をし，コードを最適化する．一方で自動チューニングは，生成コードの実行時間のみに限定しない観点で，パラメタ抽出，モデル化，および最適化を行う．
- **アルゴリズム変更：**コンパイラのコード最適化では，基本的にアルゴリズムを変更してはならない．一方で自動チューニングは，ユーザ要求の範囲でアルゴリズムを変更する．

2.4 本稿で対象とするユーザ知識

本稿では，数値計算に特化しない自動チューニング枠組みについて言及した．ここではユーザ知識を特定するため，数値計算における自動チューニングをターゲットとして説明する．そのためユーザとは，ソフトウエア開発者，エンドユーザの 2 種を仮定する．

本稿で仮定するユーザ知識は，これら 2 種のユーザごとに以下にまとめられる．

- ソフトウエア開発者知識
  - ＜対象部位＞：対象ソフトウエア上で自動チューニングが有効な箇所に関する知識．たとえば演算カーネルなど．
  - ＜アルゴリズム切替え可能情報＞：ユーザが要求する機能を満たすアルゴリズムの知識．たとえば，ベクトル計算機用実装方式や階層メモリ計算機用実装方式．
  - ＜速度パラメタ＞：コードの意味上の仕様を満たし，計算機構成方式に依存し実行速度が変化する実装方式に関する知識．たとえば，ループアンローリング方法/段数，キャッシュブロッキングのタイリング幅．
  - ＜経験値＞：ソフトウエア開発者が経験的に有効と知る値．たとえば，効果がある最大のアンローリング段数，ブロック化アルゴリズムの最大ブロック幅．
  - ＜コスト定義関数＞：対象部位において，最適化対象となる事項の挙動を示す関数の知識．
  - ＜最適化方式＞：対象部位のパラメタ最適化を，どの方式を用いると誤差が少なく実現できるかに関する知識． たとえば，適切なサンプリング点，最小二乗法を用いた最適化方式指定．
- エンドユーザ知識
  - ＜実行時情報＞：対象ソフトウエアにおいて，実行時にエンドユーザが指定する情報に関する知識．たとえば，実行する問題サイズ，適するアルゴリズムを選択するための基準値 (要求される演算精度)．

これらの知識が指定可能な枠組みが FIBER 方式である．ユーザによる知識指定の支援として，FIBER 方式ではディレクティブ ABCLibScript を提供する．

3. FIBER による自動チューニング

FIBER は，SATF に幅広く適用できるソフトウエア枠組みの確立を目指し提案された自動チューニングの枠組みである[5),6)]．具体的には SATF の構成方式を，適用対象のアプリケーションの幅を増やし，かつパラメタ推定精度を高める目的で，(1) ライブラリインストール時に行う**インストール時最適化** (Install-time Optimization)，ソフトウエア開発者が指定した特定パラメタ (たとえば問題サイズなど) を，ソフトウエア利用者が固定した時点で行う**実行起動前最適化** (Before Execute-time Optimization)，そして実

☆ たとえば BLAS の自動チューニングソフトウエア ATLAS[2)] は，パラメタ最適化を自動的に行っている．しかし，任意の対象部位やコスト定義関数の記述ができる枠組みでない．ゆえに，本稿で示す半自動化レベルの範疇にない．

☆☆ すべてのコンパイラのコード最適化は，完全自動化レベルでなくてはならない．ただし，ユーザ知識をコード最適化に与えるディレクティブなどは，半自動化レベルの範疇だといえる．

際にライブラリが実行された時点で行う**実行時最適化** (Run-timeOptimization)，の 3 種の最適化に分けて構成する枠組みである．この SATF のためのソフトウエア構成を，FIBER (Framework of Install-time, Before Execute-time, and Run-time optimization layers，ファイバー) と呼ぶ．

3.1 2 種のユーザと FIBER のソフトウエア構成

FIBER 方式では，ソフトウエア開発者が FIBER が提供するディレクティブの ABCLibScript を利用して，自分が作成したソフトウエアに自動チューニング機能を付加する．すなわち，この付加がユーザ知識の記述となる．

図 2 に，2 種のユーザからの観点での FIBER での処理手順をのせる． 図 2 では，2 種のユーザごとに処理手順が異なる．

図 2(1) **ソフトウエア開発者の処理手順：** ソフトウエア開発者は，FIBER ツールキット開発者が提供するディレクティブ ABCLibScript を用いて自動チューニングを施したい＜対象部位＞ (自動チューニング対象領域，AT 領域とよぶ) の指定と，どのような自動チューニング機能を適用するかを指定する．この自動チューニング機能指定には，概述の知識である，＜アルゴリズム切替え可能情報＞，＜速度パラメタ＞，＜経験値＞，＜コスト定義関数＞，および＜最適化方式＞の知識指定が含まれる．

その後，FIBER ツールキット開発者が提供するプリプロセッサ (`ABCLibCodeGen`) を起動する．プリプロセッサ起動後，以下の 3 種の構成要素がもとのプログラムに自動付加される．なおこのプリプロセッサにおいて，ライブラリ開発者による知識に基づき，＜2. パラメタ抽出＞および＜ 3. モデル化＞処理が自動的になされる．

- **パラメタ最適化コンポーネント：**ソフトウエア開発者が指定したパラメタを最適化する処理を含む部分．最適化方式は多種考えられるが，現在の実装では全探索 (brute-force search)，ソフトウエア開発者が指定する関数 ＜コスト定義関数＞ による探索空間の絞り込み，および最適パラメタ推定法指定 (最小二乗法) の機能を有する．すなわち，＜ 4. 最適化＞処理の実装コード部分である☆．
- AT **領域選択コンポーネント：**現在の最適化の前に行われた最適化済データが蓄積しているファイル (パラメタ情報ファイル) 上の情報と実行時情報から，AT 領域内にある最適な自動チューニング対象を選択する部分．すなわち，＜ 5. 対象実行＞処理の実装コード部分である．
- AT **領域ライブラリコンポーネント：**ソフトウエア開発者が指定した AT 領域を，FIBER 内で識別可能なサブルーチンとして蓄積する部分．

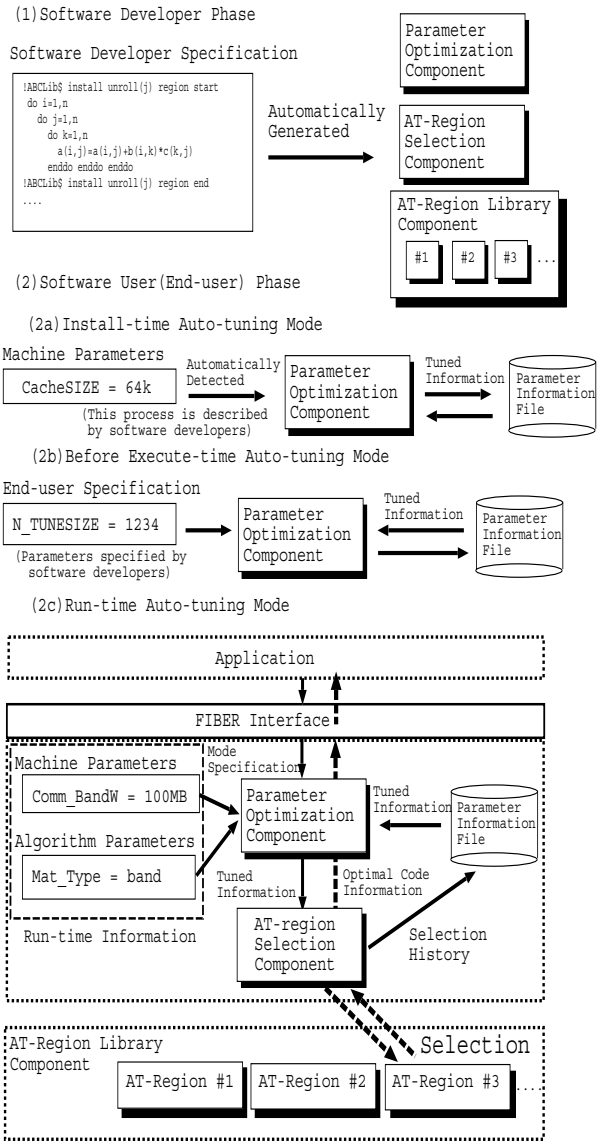

図 2 FIBER での処理手順

この処理のあと，ソフトウエア開発者は自動生成されたプログラムのソースコード，もしくはコンパイル後のオブジェクトコードをエンドユーザに公開する．

図 2(2) **エンドユーザの処理手順：** エンドユーザは，ソフトウエア開発者が公開したソフトウエアを利用する．この時，以下に示すタイミングに起因するモードで自動チューニングがなされる．

☆ FIBER 方式では，性能に影響する 2 種のパラメタ (PP と BP) の概念をシステム変数として導入することにより自動チューニングにおける最適化の定義をしている．すなわち最適化とは，AT 領域におけるコストの挙動を定義した関数（コスト定義関数 F）について，パラメタ BP を固定した上で，PP を 3 種のタイミング（インストール時，実行起動前，および実行時）においてコスト定義関数 F を最小化する処理と定義する.（文献 5)，6) を参照）後述の ABCLibScript での最適化処理は，この PP と BP の概念を導入している．

図 2(2a) **インストール時自動チューニングモード：** 公開されたソフトウエアのインストール時に，**インストール時最適化**を用いた自動チューニングを行うモードである．このモードでは，ソフトウエア開発者が記述した処理に従い，キャッシュサイズなどのエンドユーザが利用している計算機環境情報を自動取得し，それを最適化に利用する．この自動チューニングは，エンドユーザが特に意識することなく行われる．

図 2(2b) **実行起動前自動チューニングモード：**ソフトウエア開発者が指定したパラメタについて，エンドユーザがその値を固定した時を実行起動前時とよび，その時に行われる最適化が**実行起動前最適化**である．この実行起動前最適化を用いて自動チューニングするモードである．このモードでは，インストール時自動チューニングと異なり，エンドユーザが積極的にFIBERツールキット開発者が提供するAPI(Application Programing Interface) を利用し，自動チューニングのための情報をシステムに通知する．すなわちエンドユーザは，概述知識の＜実行時情報＞をシステムに通知する．

図 2(2c) **実行時自動チューニングモード：** ソフトウエア開発者が指定したAT領域が実行された時に，**実行時最適化**を用いて自動チューニングするモードである．このモードでは，実行時におけるシステム情報や，この実行より前に行った当該AT領域での自動チューニング情報 (インストール時最適化情報，実行起動前最適化情報，および現実行より前に実行された当該AT領域の最適化情報) から，ソフトウエア開発者が指定した自動チューニングを行う．

3.2 自動チューニング記述用ディレクティブ ABCLibScript

3.2.1 指示形式

ソースプログラムにおいて，!ABCLib$で始まる行はABCLibScriptによる自動チューニングのための指示行とみなす．この表記法の概略を**図 3** に示す．

```
!ABCLib$ ⟨ 自動チューニング種類 ⟩ ⟨ 機能名 ⟩
        [(対象変数)]  region  start
 [!ABCLib$ ⟨ 機能詳細 ⟩ [sub region start]]
              AT 領域
 [!ABCLib$ ⟨ 機能詳細 ⟩ [sub region end]]
!ABCLib$ ⟨ 自動チューニング種類 ⟩ ⟨ 機能名 ⟩
        [(対象変数)]  region  end
```

図 3 ABCLibScriptにおける自動チューニング指示形式

図3における⟨ 自動チューニング種類 ⟩や⟨ 機能名 ⟩のことを指定子とよぶ．また⟨ 機能詳細 ⟩のことを補助指定子とよぶ．

3.3 ABCLibScriptによるユーザ知識記述例

以下は，ソフトウエア開発者がABCLibScriptを用いて自動チューニングのための知識記述を行う場合の，プログラミング例について述べる．

3.3.1 インストール時最適化

以下のプログラム例1は，インストール時最適化において行列積コードのアンローリング段数調整処理を記述した例である．

【プログラム例1】行列積コードのアンローリング

```
!ABCLib$ install unroll (i) region start
!ABCLib$ name MyMatMul
!ABCLib$ varied (i) from 1 to 16
!ABCLib$ fitting least-square 5
!ABCLib$ & sampled (1-5, 8, 16)
do i=1, n
  do j=1, n
    da1=A(i,j)
    do k=1, n
      dc=C(k,j)
      da1=da1+B(i,k)*dc
    enddo
    A(i,j)=da1
  enddo
enddo
!ABCLib$ install unroll (i) region end
```

プログラム例1では，region start〜region endの間のコードをチューニング対象としている．また，install指定子を用い，対象のインストール時最適化を指定している．これが，＜対象部位＞知識である．

プログラム例1では，概述の2種のシステムパラメタのうち基本パラメタ BP について，特にソフトウエア開発者が指定していないので，デフォルト基本パラメタ BP を問題サイズに関する変数 n としている．これが，＜最適化方式＞知識である．

unroll指定子を用い，かつループ変数 i を指定しているので，性能パラメタ PP は対象となる行列積ループの最外側ループ変数 i に対するアンローリング段数となる．これが，＜速度パラメタ＞知識である．

varied補助指定子から，アンローリングは1段から16段まで行うことを定義している．これが，＜経験値＞知識である．

また，fitting補助指定子から最適化の対象となるコスト定義関数は5次多項式となる．unroll指定子が使われているので，対象となるAT領域の実行時間の挙動に関するモデル化を指定したことになる．これが，＜コスト定義関数＞知識である．

システムとしては，パラメタ BP をAPIで指定した情報を用いて固定し，パラメタ PP を推定する．このパラメタ PP については，sampled補助指定子によりサンプリング点 (＝アンローリング段数) が指定されている．それらは，1段から5段，8段，および16段で，この段数において実行時間を実測する．この実測結果を用いて，指定された5次多項式モデルによる PP の挙動を決定する (＝5次多項式の係数を決定する) が，その方法として最小二乗法を指定 (least-squre補助

指定子から) している．これらが，＜最適化方式＞知識である．

つぎに以下のプログラム例 2 は，ブロック化アルゴリズムを用いて実装されているカーネルサブルーチンをコールする場合における，ブロック幅調整処理を記述した例である．

【プログラム例 2】ブロック幅調整

```
!ABCLib$ install variable (MB) region start
!ABCLib$ name BlkMatMal
!ABCLib$ varied (MB) from 1 to 64
do i=1, n, MB
  call MyBlkMatVec(A,B,C,n,i)
enddo
!ABCLib$ install variable (MB) region end
```

プログラム例 2 では，variable 指定子により性能パラメタ PP がブロック幅変数 MB であることを指定する．これが，＜速度パラメタ＞知識である．

このブロック幅変数 MB の範囲はvaried 補助指定子により，1 から 64 であることを指定している．これが，＜経験値＞知識である．

3.3.2 実行起動前最適化

以下は，アルゴリズム選択処理に関するプログラミング例である．アルゴリズム選択基準として，ソフトウエア開発者が定義するコスト定義関数を利用する．

【プログラム例 3】アルゴリズム選択処理

```
!ABCLib$ static select region start
!ABCLib$ name TestSelect
!ABCLib$ parameter (in CacheS,in NB,in NPrc)
!ABCLib$   select sub region start
!ABCLib$   according estimated
!ABCLib$ & (2.0d0*CacheS*NB)/(3.0d0*NPrc)
               AT 領域 1
!ABCLib$   select sub region end
!ABCLib$   select sub region start
!ABCLib$   according estimated
!ABCLib$ & (4.0d0*CacheS*dlog(NB))
!ABCLib$ &  /(2.0d0*NPrc)
               AT 領域 2
!ABCLib$   select sub region end
!ABCLib$ static select region end
```

プログラム例 3 では，static 指定子を用い，対象部位の実行起動前最適化を指定している．これが，＜対象部位＞知識である．

AT 領域において，アルゴリズムの選択処理をすることを指定 (select 指定子) している．性能パラメタ PP として，AT 領域 1 もしくは AT 領域 2 の実行指定がパラメタ化される．これが，＜アルゴリズム切替え可能情報＞知識である．

ソフトウエア開発者が定義するコスト定義関数は，according estimated 補助指定子で定義されている．この例でのコスト定義関数では，この最適化が行われる前に定義された浮動小数点変数 (CacheS, NB, NPrc) が参照されている．AT 領域 1 のコストは $(2.0d0 * CacheS * NB)/(3.0d0 * NPrc)$ で見積もられ，AT 領域 2 のコストは $(4.0d0 * CacheS * dlog(NB))/(2.0d0 * NPrc)$ で見積もられる．これが，＜コスト定義関数＞知識である．

コスト定義関数のコスト評価は，実行起動前自動チューニングモードで行われる．そして，実行 AT 領域が 1 つ決定される．その後，対象となる AT 領域の実行時において，決定された AT 領域のみが選択的に実行される．これが，＜ 5. 対象実行＞処理である．

4. 性能評価

ここでは，東京大学情報基盤センタの SR8000/MPP☆，京都大学学術情報メディアセンタの VPP800/63☆☆，および PC クラスタ☆☆☆を利用して，ユーザ知識を活用する自動チューニング方式の予備評価を行う．

4.1 対　　象

評価対象ベンチマークとして，我々が開発した並列数値計算ライブラリABCLib_DRSSED に入っている，以下のルーチンについて行った．

- 固有値ソルバ (EIG)：Householder 三重対角化，Hoseholder 逆変換などによる密対称行列の全固有値・全固有ベクトル求解ソルバ．Householder 三重対角化のカーネル部分は BLAS2 で構成されている．ブロック化アルゴリズムでない．カーネルに関してアンローリング段数の調整を行う．ベクトルリダクションに関する通信実装方式選択が 2 種ある．また，Hoseholder 逆変換のカーネル部分は BLAS1 で構成されている．ブロック化アルゴリズムでない．カーネルに関してアンローリング段数の調整を行う．ベクトル放送処理に関する通信実装方式選択が 3 種ある．
- QR 分解による直交化 (QR)：カーネル部分は BLAS3 で構成されている．ブロック化アルゴリズムである．カーネルに関してアンローリング段数の調整を行う．ブロック幅調整と，ブロック幅に応じて通信粒度が調整できる．

これらは，＜対象部位＞，＜速度パラメタ＞，および＜アルゴリズム切替え可能情報＞知識を記述したベンチマークである．

対象の自動チューニング処理は以下の通りである．

- インストール時最適化：ソフトウエア開発者によ

☆ 日立最適化 Fortran90 V01-04, コンパイラオプション “-opt=4, -parallel=0”, および日立最適化 MPI 利用.

☆☆ 富士通最適化 UXP/V Fortran/VPP V20L20, コンパイラオプション “-O5 -X9”, および富士通最適化 MPI 利用.

☆☆☆ Pentium4 2.0GHz, 1GB メモリ (Direct RDRAM/ECC256MB*4) 4 台, MB ASUTekP4T-E+A (Socket 478), ネットワークカード Intel EtherExpressPro100+, OS Linux 2.4.9-34, PGI 社製コンパイラ Fortran90 4.0-2, コンパイラオプション “-fast”, および MPICH 1.2.1 利用.

る＜コスト定義関数＞知識として，対象部位の実行時間挙動を指定する．この関数として 5 次多項式を指定する．問題サイズに関するサンプリング点は，実行問題サイズ以外を指定する．＜最適化方式＞知識として最小二乗法を指定する．

- 実行起動前最適化:エンドユーザによる＜実行時情報＞知識として，実行する問題サイズを指定する．＜最適化方式＞知識として，全探索を指定する．

4.2 結　　果

表 1 は，固定パラメタを用いた実行を「ユーザによる知識なし」とみなし，その速度向上比を 1 としたときの，各最適化方式の平均速度向上比である☆．表 1 から，ユーザ知識を自動チューニングに活用すると，1.02 倍〜2.23 倍の速度向上が達成できることがわかる．いずれも，実行起動前最適化がインストール時最適化より高い速度向上比を得た．これは，エンドユーザによる＜実行時情報＞知識である実行問題サイズ指定が，ライブラリ開発者知識を用いたインストール時最適化によるパラメタ推定よりも効果的であることを意味している．

表 1 ユーザ知識を活用した自動チューニング機構による平均速度向上比．固定パラメタによる実行を 1 とする．PC クラスタでは 512, 5123, 10123，それ以外では 512, 5123, 6123 の各問題サイズによる自動チューニング済みパラメタを用いた実行時間との各速度向上比から平均値を算出．

(a) HITACHI SR8000/MPP

最適化	知識なし	インストール時	実行起動前
EIG	1.00	1.14	1.28
QR	1.00	2.06	2.23

(b) Fujitsu VPP800/63

最適化	知識なし	インストール時	実行起動前
EIG	1.00	1.14	1.15
QR	1.00	1.88	1.91

(c) PC クラスタ

最適化	知識なし	インストール時	実行起動前
EIG	1.00	1.02	1.09
QR	1.00	1.16	1.44

5. おわりに

本稿では，ユーザ知識を活用することによるソフトウエア自動チューニングの枠組みについて述べた．またその実現例として，FIBER 方式を説明した．さらに，FIBER 方式によるユーザ知識記述ディレクティブ ABCLibScript の説明と，それによるユーザ知識記述例を示した．

自動チューニング機構の構築においては，どこまで「自動化」を行うかが本質的な問題となる．本稿では，狭義の自動チューニングである，パラメタ最適化処理のみ完全自動化することを自動チューニングと定義した．この定義では，それ以外の処理は半自動化のみ行う．また，その半自動化のためにユーザ知識の記述が必要となる．

ユーザ知識を自動チューニングに活用する場合，その知識を記述するユーザの能力が自動チューニング機構の性能を決定する．また知識記述が誤っていた場合，最適化結果も誤差を含む．現在，誤差に関する定量的評価手段や自動回避手段がない．これらの手法の確立や有効となるツール開発が今後の課題となろう．

なお ABCLibScirpt 試用版は，科学技術振興機構の助成による WWW サーバ (`http://www.abc-lib.org/`) 上でマニュアル等を含め公開される．

謝辞 日頃討論頂く自動チューニング研究会のみなさま，東京大学 須田礼仁 助教授，電気通信大学 今村俊幸 講師，名古屋大学 山本有作 講師，および日立中央研究所 直野健 氏に感謝いたします．なお本研究は，科学技術振興機構さきがけプログラムの助成による．

参 考 文 献

1) 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 (1997).
2) ATLAS project; http://www.netlib.org/atlas/index.html.
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.SIG 12 (HPS 4), pp. 60–76 (2001).
5) Takahiro, K., Kise, K., Honda, H. and Yuba, T.: FIBER: A General Framework for Auto-Tuning Software, *Proceedings of The Fifth International Symposium on High Performance Computing*, Vol. Springer Lecture Notes in Computer Science, No. 2858, pp. 146–159 (2003).
6) Takahiro, K., Kise, K., Honda, H. and Yuba, T.: Effect of Auto-tuning with User's Knowledge for Numerical Software, *Proceedings of ACM Computing Frontiers 04*, pp. 12–25 (2004).

☆ 経験的に良い固定パラメタは，ある意味，ソフトウエア開発者による知識といえる．本評価では，自動チューニングにユーザ知識を入れた機構の評価を対象とするので，自動チューニングを全く行わない固定パラメタによる実行を「知識なし」とした．