自動チューニング技術に関する課題調査

2008 年 11 月

自動チューニング研究会

標題：自動チューニング技術に関する課題調査
自動チューニング研究会

序

計算機ソフトウェアの自動チューニング（Automatic Tuning）技術は、高性能コンピューティング分野における計算科学からの要請をうけて生まれている。自動チューニングの枠組みは、古くは自動制御理論が対象としてきた領域である。本報告書では、計算科学（Computational Science）にとどまらないで計算機科学（Computer Science）という、より大きな立場から自動チューニング技術をとらえる。今回の調査対象の重心は計算科学におくが、今後の発展を考えると、計算機科学全般に敷衍して適用すべきと考える。

2003 年 11 月、ソフトウェアによる自動チューニング技術に関する研究会を発足させた。以来 5 年を経て、自動チューニング技術における過去の成果と現状の問題点を踏まえ、今後取り組むべき課題について調査することとした。同分野の研究展開において、新たな課題提案の指針を示すものとなることを企図した。執筆にあたって、目次の章（分野）担当者は、既発表論文、自動チューニング研究会資料などをベースに自由に発想をめぐらし、記述内容については研究会で議論を重ねた。

第 1 章では、本課題調査の目的と意義について述べる。第 2 章では、本報告書で扱う自動チューニング技術の枠組みを解説する。第 3 章では、数値計算応用を中心とした自動チューニング技術の研究動向を総括する。第 4 章以降では、章の標題に関するこれまでの研究成果を概観し、問題点を明らかにする。それを踏まえて、今後取り組むべき研究課題について論じる。本報告書の目次構成と執筆担当者を次頁に示す。

本報告書が、自動チューニング研究の今後の発展に寄与することを願っている。なお、本調査は、2007 年 7 月以降、約 1 年半の歳月をかけて行われた。上記執筆者に加えて、その間、熱心に議論に参加された自動チューニング研究会のメンバ各位に感謝する。

2008 年 11 月
弓場敏嗣/自動チューニング研究会主査

■報告書の構成と執筆者

1. 調査の目的　（弓場敏嗣　電気通信大学）
2. 自動チューニングのモデル　（片桐孝洋　東京大学）
3. 自動チューニング技術の研究動向　（直野　健　日立製作所）
4. 自動チューニングのための数理的共通基盤　（須田礼仁　東京大学）
5. 自動チューニング機構とシステム　（片桐孝洋　東京大学）
6. 自動チューニング機能をもつライブラリ　（黒田久泰　東京大学）
7. 自動チューニングが対象とするシステム　（今村俊幸　電気通信大学）
8. 自動チューニングを支援するソフトウェアツール　（伊藤祥司　理化学研究所）
9. 自動チューニングが適用可能な応用分野　（山本有作　名古屋大学）
   - 数値計算応用　（山本有作　名古屋大学）
   - 信号処理応用　（山本有作　名古屋大学）
   - データベース応用　（直野　健　日立製作所）
   - Webシステム応用　（直野　健　日立製作所）
   - 認識システム応用　（黒田久泰　東京大学）
   - ゲームコンピューティング応用　（黒田久泰　東京大学）
   - 実世界コンピューティング応用　（片桐孝洋　東京大学）
   - バイオインフォマティクス応用　（清水謙多郎　東京大学）

第1章　調査の目的

1.1 調査の目的

1.1.1 目的と意義

2003 年 11 月、ソフトウェアによる自動チューニング技術に関する研究会を発足させ、以来、5 年を経ている。研究会活動としては、研究ファンドへの応募と採択された課題の実施という形で、自動チューニング（Automatic Tuning; 以下 AT と略記）技術の発展に寄与してきたと総括できよう。6 年目を迎え、AT 技術における現状の問題点を踏まえ、今後の課題について調査し、報告書としてまとめることは、研究会の今後に向けて意義深いと考える。一般に、「調査報告書」は、過去の成果と現状の総説、あるいは、今後に向けた研究課題の探索という 2 つの性格をもつ。今回の課題調査においては、後者に力点をおくこととする。すなわち、今後の研究展開において、新たな課題提案の指針を示すものとなることを企図した。

研究課題の探索を目的とした調査の意義は、以下の 3 つに細分化できる。

1. 新規課題の所在調査
2. 新規ファンド獲得に向けた提案への準備
3. 自分（分野担当者）にとっての再考機会

1 と 2 の意義は建て前である。本当の意義は、この機会に自動チューニングの枠組みを見直し、自分（達）の興味のある分野に＜自動チューニングパラダイム＞を持ち込むとどうなるかを再考（再検討）することと考えている。そうすることが、魅力的な新しい研究課題の発見につながると信じている。その意味では、本課題調査においては、新しい研究課題の枠組みをスケッチするだけでよい。とりあえずは、提案課題の内容に深入りしないこととした。発想を広げて、自動チューニング技術の新しい可能性を見つけることが重要である。本報告書の記述では、参考文献リストを必要としないとか、図表の参照はできるだけ少なくする、という指針の意味するところである。研究会での議論を通じて、自動チューニング技術に関する各自の現状認識を確認し、研究企画や研究実施の方法論を深化させる狙いもある。

情報処理学会誌特集号向けに準備を進めている自動チューニング技術の総説（動向解説）は、以下の特徴をもつ。両者の部分的な内容重複は可とし、バランスの問題と考えた。

1. 学会会員（一般読者）向けの技術動向の総説
2. 自動チューニング技術の重要性を広く啓蒙

3. 同研究分野への新規参入者に対する勧誘と招待
4. 特集号の構成論文ごとに高い独立性
5. 参考文献、解説のための図表が重要

各章では、章の標題に関するこれまでに得られた知見を概観し、問題点を明らかにする。それを踏まえて、今後取り組むべき研究課題について論じる。

1.1.2 調査の視点と方法

ソフトウェアによる自動チューニング技術は、後述するように、高性能コンピューティング分野における計算科学（Computational Science）からの要請をうけて生まれている。しかし、自動チューニングの枠組みは古くは自動制御理論が対象としてきた領域であり、計算機科学（Computer Science）という、より大きな立場からとらえることもできる。歴史的経緯から、今回の調査対象分野の重心は計算科学におくが、今後の発展を考えると計算機科学全般に、より広範囲に適用すべきと考える。とはいえ現実的には、自動チューニング研究会メンバが、これまで現実にかかわってきた分野により強く焦点があてられる。調査の過程で、自動チューニング技術分野の取り組むべき課題を展望するが、それは本報告書を現実的研究課題の提案へと繋げる意図から発する。執筆にあたって、各目次の章（分野）担当者は、既発表論文、AT 研究会資料などをベースに自由に発想をめぐらし、記述された内容については研究会において議論を重ねた。約1年半を要した調査の手順は、以下の通りである。

1. 自動チューニング技術の発展をにらみ、課題分野を目次化する。
2. 分担執筆部分について、研究会の場で内容を議論する。
3. 議論を踏まえ、修正したものを集積する。
4. 統合版として1つの書きものとなるように推敲する。
5. 推敲にあたっては、用語の統一、内容の単純化、重複の排除を目指す。
6. 統合版を再検討後、「AT 研究会資料」として製本する。
7. 時期を見て、調査結果をホームページに公開する。

1.2 自動チューニング技術の枠組み

1.2.1 自動チューニングの定義

自動チューニング（Automatic Tuning; AT）は、広義には自動的に性能をチューニングすること、あるいは、性能チューニングの自動化と定義される。このとき、＜性能＞は計算機の計算速度に限らない。ネットワークを対象とすれば通信速度が性能となる。計算と通信の両者を包含した性能として、応答速度が用いられる場合もある。速度以外にも、所要記憶量、消費電力量、計算精度、性能安定性、信頼性など、性能基準を

表す測度が設定可能な対象はすべて AT の枠組みに含まれる。

今まで中心的に取り組まれてきた計算科学（数値計算応用）では、問題（応用）を記述したあるいはこれから記述するソースプログラム（ソースコード）の性能向上をもたらす自動的な性能チューニングと狭義に定義している。対象計算機への問題の自動的な最適化写像という表現も可能である。＜自動＞の狭義な意味合いは、与えられた応用（数値計算）プログラムを、異なる機種の計算機に対しても高性能が得られるようにチューニングする過程を自動化することから出発している。つまり、人手による性能チューニング作業の自動化が原点（原義）である。この視点にとどまることなく、自動チューニングの枠組みが適用可能な応用分野を拡大する必要がある。各応用においては、性能基準の多様性に留意する。定義にこだわる理由は、以下の通りである。「課題調査」を進める中で、AT の範囲が人によって異なるとの認識をもった。この研究会が考えるところの AT の独自性を主張することが重要と考えた。つまり、他の分野で一般に使われていた＜自動＞という用語との差異を明らかにする必要があった。パラダイムの概念は、拡げること、絞ることを繰り返すことで、均衡点としてその特徴付けが明確になる。今回の調査においては、応用を数値計算分野に限定しない広義の AT の枠組みを対象とする。

1.2.2 自動チューニングの技術的枠組み

意義、有用性の観点ではなく、情報（科学）技術として見た場合の自動チューニング（AT）の特徴を考える。AT の技術的枠組みに関して、研究会メンバの一応のコンセンサスを以下に掲げる。

1. AT 技術は、広義には最適化技術である。
2. AT は必ずしも＜自動＞でなくてもよい。
3. AT はプログラマに可視、不可視の両方の機能を許容する。
4. AT は実時間処理、一括処理を使い分けることができる。
5. AT は最適化に関するパラメタ空間が大きい。
6. AT 処理の過程でフィードバックを許す。
7. AT（最適化）対象に、知識とモニタリング結果を反映できる。
8. AT 機構を集中あるいは分散して配置できる。
9. AT 基準を与える目的関数は、プログラマによって定義が可能である。
10. AT の継続的適用過程を学習して、知識とすることができる。
11. AT（最適化）基準は、速度性能以外に多様性がある。
12. システム実装において、AT 支援環境が重要である。
13. AT の対象は、必ずしもソフトウェアに限らない。

14. AT の実現方法は、ソフトウェア手法にもとづく。

現状では、このような多面的な属性記述の集積によって、自動チューニングとは何かを示すことが現実的であろう。

1.3 自動チューニング研究会

1.3.1 活動の経緯

自動チューニング（AT）研究会は、2003 年 11 月 4 日に電気通信大学（東京都調布市）において、第 1 回会合を開催した。同会合の出席者は、今村俊幸、片桐孝洋、須田礼仁、直野健、山本有作、弓場敏嗣（50 音順）の 6 名であった。以来約 5 年の間、年間5回程度の研究会を開催し、以下の活動を進めてきた。なお、2008 年 10 月現在の研究会メンバ総数は 18 名である。

1. 科学研究費補助金費など外部資金（ファンド）申請と獲得による自動チューニング技術に関連した研究の推進
2. 学会・研究会における自動チューニング関連セッションの企画と論文発表
3. 国際会議 iWAPT（International Workshop on Automatic Performance Tuning）の企画と実施

1.3.2 今後の展開

AT 研究会のこれまでの 5 年間の歩みは、おおむね順調であったといえよう。その間、自動チューニング技術については着実な進展があり、かつ世間的な注目度も高まってきている。研究会の今後の展開を考えるとき、現時点での技術課題を整理しておく意義は大きい。当面の総合的な技術課題は、以下の通りと考える。

1. AT 技術が適用可能な問題の種類と大きさの分類
2. AT 研究の到達度評価基準の設定
3. AT 技術分野の発展への学術的工程表（ロードマップ）の描出
4. 既提案の AT フレームワークについての有効性評価

一方、自動チューニング研究会の活動体制を強化することも、重要な課題である。現在までは、有志による非公式研究会として運営してきた。上記の活動内容を適切に実施するためには、自由度のある非公式研究会形式の運営という利点は大きい。既存の学会との結びつきを公式なものとするか否かについても、適宜議論を重ねてきた。学会傘下の公式研究会とすれば、世間へのプレゼンスの大きさ、研究会活動記録の保存、事務作業の負担減などの利点が期待される。それらが活動体制の強化につな

がるなら、例えば、情報処理学会、日本応用数理学会などに所属する公式研究会となる道筋を考える意味がある。

国際会議 iWAPT については、自動チューニング研究の世界の中心となることが期待される。過去 2006 年、2007 年、2008 年と 3 回にわたって開催され、AT 研究の成果を発表する国際会議としての国際的な認知が深まりつつある。AT 研究会の今後の活動も、iWAPT の発展を主体的に支援することが重要な責務となる。

第 2 章　自動チューニングのモデル

2.1　自動チューニング機構の構成

自動チューニング機構（AT 機構）には、AT 候補を機構内の指定により呼ぶことが可能な**AT 候補蓄積機構**が存在する（図 2.1）。ここで AT 候補とは、自動チューニングによって高性能をもたらす可能性がある変換後の目的コード群である。AT 候補は、AT 対象の原始プログラムを入力として、AT 仕様記述言語などにより自動的もしくは半自動的に作成される。AT 候補の実行に際して、実行時間、精度、および記憶量などの性能指標をモニタする機構がある。モニタ機構は、AT 候補蓄積機構と連結されている。また、モニタ機構は、自動チューニングデータベース（AT データベース）に、モニタした性能データを自動チューニングデータ（AT データ）として収納する。

学習機構は、AT データを洗練して、高精度のパラメタ推定で活用するための前処理を行う。学習機構での処理の対象は、パラメタ推定機構で必要となる性能モデル自体の決定、性能モデルを決定するために必要なパラメタの決定、および、性能モデルの精度を高めるためのパラメタの決定、である。学習機構での処理は、対象プログラムがインストールされるとき、ユーザが知識を与えたときなど、パラメタ推定が必要となる事前、もしくは、パラメタ推定要求時に行われる。学習機構はモニタ機構と連結しており、必要に応じて AT 候補の実行による AT データを採取できる。過去実行された AT データは、AT データベース上に残っている。学習機構では、データベース上の過去の AT データを利用して、性能モデルの学習、データベース上にあるデータ自体の洗練を行う。**パラメタ値推定機構**では、ユーザ知識やポリシを入力とし、学習機構により決定される性能モデルをもとに最適パラメタ値を決定する。なお、AT ポリシとは、ソフトウェア開発費用、開発デットライン（時間）、開発工数、計算機資源、提供品質（演算精度）などの制約をもとに、実行時間、精度、記憶量などを最適化する際のユーザ側からの要求を **AT ポリシ**と呼ぶ。その他、最近注目されている周波数切り替えによる低消費電力化もAT対象となる。DVS（Dynamic Voltage Scaling）制御による CPU 周波数切り替えにより、低消費電力化という性能チューニングが可能となる。

指定された対象プログラムの AT 実行の流れは、以下のように進行する（図 2.1）。

1. AT 機構もしくはユーザの要求により、モニタ機構を起動する。モニタ機構により、AT 候補の実行データがデータベースに蓄積される。学習機構は、データベース上にある過去の AT データ、および、モニタ機構を通じて得た現在の AT データを参照して、AT データおよび性能モデルを構築する。
2. パラメタ値推定機構は、ユーザ知識とポリシを入力として最適パラメタ値を推

定する。このとき、モニタ機構を通じて獲得した AT データおよび学習によって得た AT データベースを参照して、もっともらしい値を推定する性能モデルを確定する。

3. 性能モデルを利用し、推測される最適パラメタ値を出力する。出力された最適パラメタ値はデータベースに蓄積される。過去の AT データとして学習機構で利用することで、結果のフィードバックを行う。

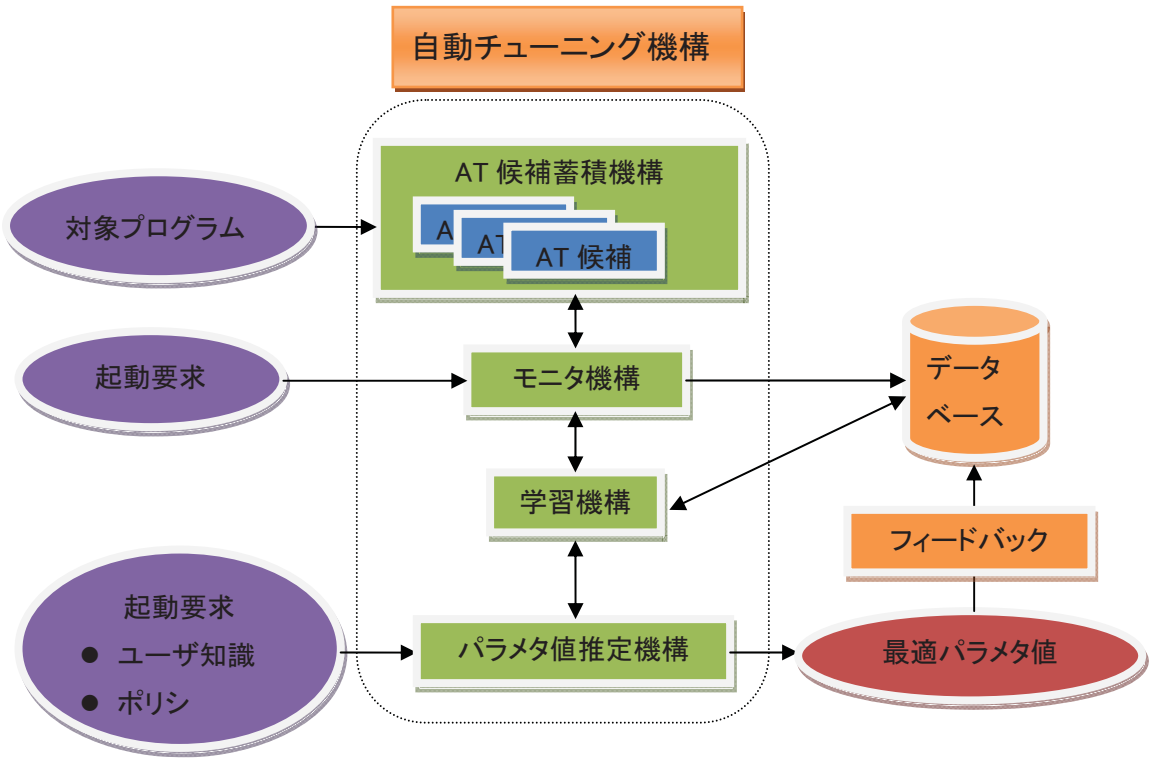

図 2.1 自動チューニング機構の構成

2.2 自動チューニング機構の機能

AT 機構の機能には、以下の 4 つの機能がある（図 2.2）。

1. コード生成機能
2. 対象実行機能
3. モデル化機能
4. パラメタ探索機能

2.2.1 コード生成機能

AT 対象となるプログラムの指定された箇所において、AT 候補を自動生成する機能である。AT 候補は、AT 対象となるプログラムの箇所そのもの、もしくは、AT 対象となるプログラムを処理（例えば、アンローリング処理）して自動生成されたプログラム（目的コード）となる。対象プログラムにおいて最適化パラメタが陽に定義されている場合（例えば、アルゴリズム選択のみ行う場合）は、コード生成機能において AT 候補が自動生成されることはないが、AT 機構でパラメタ推定できるような制御プログラムが付加される。AT 候補はそのコード自体がライブラリ化（関数化）され、AT 機構内で区別できる内部識別番号がつけられる。内部識別番号を制御する変数が、AT 機構での最適化パラメタとなる。その他、必要に応じて、コード生成機能により、AT 機構を実現するための機構が自動生成される。以下に、コード生成機能を形成するために必要な機能をあげる。

1. 対象実行機能において、実行可能なコードを自動生成する。
2. AT 機構内で、適切な AT 候補を参照できるように制御する。
3. パラメタ値を指定することで、AT 候補を実行できる。

2.2.2 対象実行機能

最適化パラメタ値を指定することにより、AT 候補を実行する機能である。AT 候補の増大によるコード量増加を防ぐため、AT 候補のコードを部分的にコンパイルし、実行可能コードを生成する。実行時間計測の必要がない場合、これを省略し、その代わりにコストを定義する関数（目的関数）を評価する。得られた AT データについて、AT 機構内部で参照可能なデータとして AT データベースに蓄積する。以下に、対象実行機能を形成するために必要な機能をあげる。

1. AT 候補のコードをコンパイルし、実行可能コードを生成する。
2. パラメタ推定機構からの出力結果である最適パラメタ値により、最適パラメタを固定し実行する。
3. AT データをデータベース化して蓄積する。

2.2.3 モデル化機能

学習機構は、性能モデルの精度を高めるために、目的関数のパラメタを変更（**モデルフィッティング**）する機能をもつ。以下に、モデル化機能を形成するために必要な機能をあげる。

1. 目的関数中の未定パラメタを、データベースの AT データをもとに決定する。
2. ユーザからの指示、もしくは AT 機構自らの判断により、目的関数自体を修

正・決定する。

3. 目的関数の精度を高める目的で、学習に必要な AT データをデータベースに蓄積する。

2.2.4 パラメタ探索機能

モデル化機能で更新された目的関数をもとに、よりよい最適化パラメタの値を推定する機能をもつ。また、最適化パラメタが、十分によい値であることを判断する機能をもつ。十分に最適化されたと判断する場合は、AT を停止する。以下に、パラメタ探索機能を形成するために必要な機能をあげる。

1. 目的関数をもとに、よりよい最適化パラメタの値を推定する。
2. 推定された最適パラメタ値が十分に高精度なものか判断し、AT を停止するか否かを決定する。

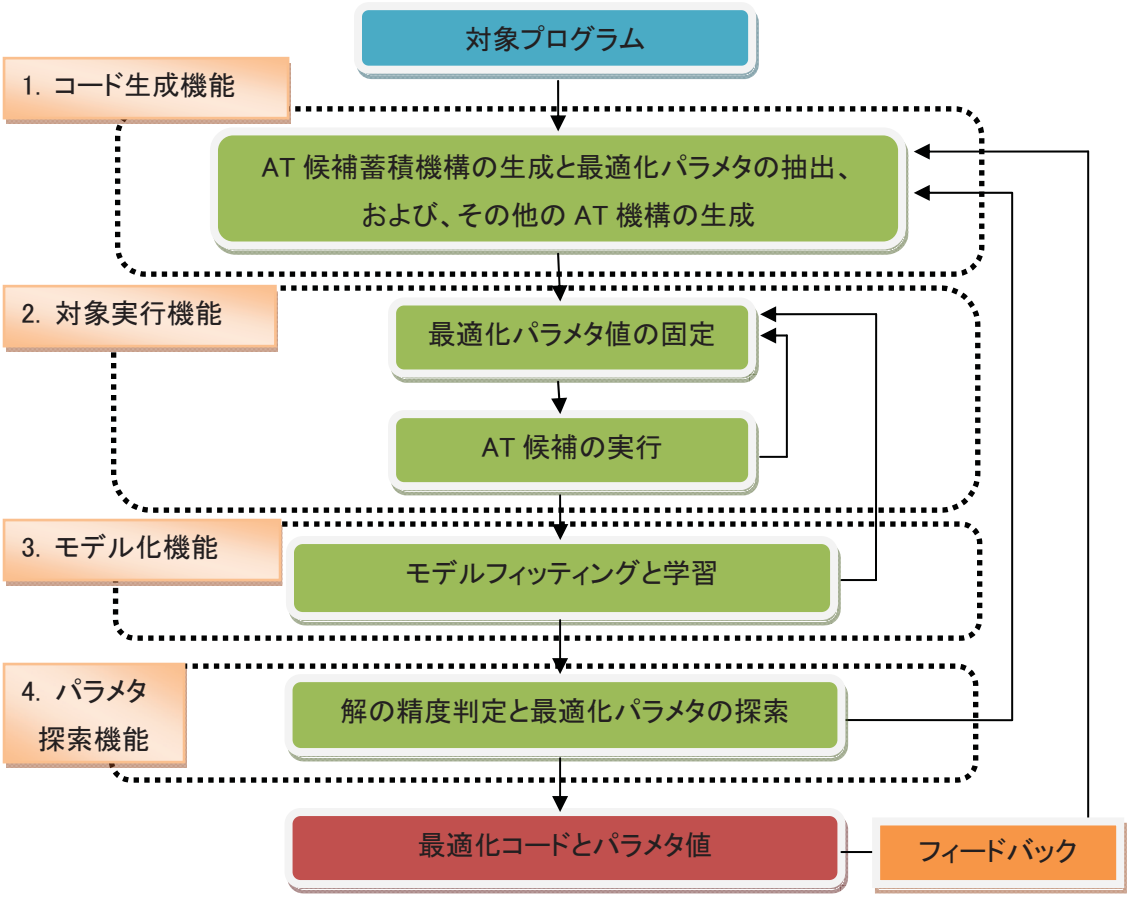

図 2.2 自動チューニング機構の機能と処理の流れ

2.3 自動チューニングにおける時空間階層性

2.3.1 実行タイミングの階層

図 2.2 で示す自動チューニングのすべての過程が、一括して実行される訳ではない。先験的に実行可能な過程の場合は、例えば、インストール時に処理をすませておき、コンパイル時に新たに得られた情報はその都度利用する。AT 実行のタイミングは、対象となるプログラムの実行前に行う**静的自動チューニング**と、プログラム実行中に行う**動的自動チューニング**に大別される。また、対象プログラムの原始コードについて、そのコンパイル時に自動チューニングを行う**コンパイル時自動チューニング**と、コンパイル後の実行時に行う**実行時自動チューニング**に分けられる。一方、AT ソフトウェアをユーザが使うタイミングをもとに、3 通りのタイミングで自動チューニングを分類することがある。すなわち、AT ソフトウェアのインストール時に行う**インストール時自動チューニング**、AT ソフトウェアのエンドユーザが知識情報を与えたときに行う**実行起動前自動チューニング**、AT ソフトウェアの実行時に行う**実行時自動チューニング**である。インストール時、実行起動前、実行時という時間経過的な制約は、図 2.3 で示す時間階層として表現できる。最適化パラメタについても、この制約に従う時間階層性がある。

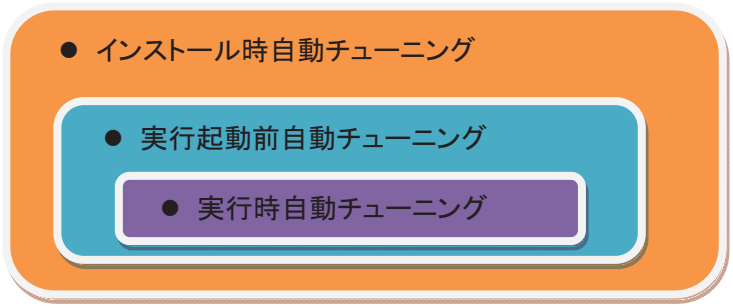

図 2.3 自動チューニング実行の時間階層

2.3.2 AT 対象の階層

対象プログラム自体が、階層化された構造をもつ場合がある。例えば、多くの数値シミュレーションソフトウェアでは、メッシュ生成や行列生成処理などの処理が上位で、ソルバと呼ばれる数値計算ライブラリが中位、数値計算ライブラリで用いられている基本線形代数副プログラム BLAS などの演算カーネルが下位となる構成をもつ。システムソフトウェアを対象とする AT を行う場合、上位からミドルウェア、コンパイラ、OS、ハードウェアの階層における最適化パラメタが存在する。上位の最適化パラメタは、必ずしも下位のパラメタに影響を及ぼさない。しかし、ハードウェアに密着する OS の最適化パラメタが、ハードウェアの最適化パラメタに影響を及ぼすことがある。

対象プログラム上の性能パラメタ、アプリケーションソフトウェアや計算機システムに現れるパラメタ化されたものはすべて AT 対象となる。計算機科学においては、アプリケーションソフトウェアと、計算機のシステムソフトウェアとを区別することが多い。それにならって、AT 対象について両者を分けて考える。計算機科学においては、アプリケーションソフトウェアが上位層、システムソフトウェアが下位層に位置づけられる。さらにシステムソフトウェアにおいては、上位から、ミドルウェア、コンパイラ、OS となり、ハードウェアに近くなるほど下位の階層に位置づけられる。最適化パラメタにおいても、この**空間階層性**を意識して階層化される。

1. アプリケーションソフトウェア上の AT 対象
   - アプリケーションにおける問題レベル： 物理場の初期状態、メッシュ形状、実測情報とシミュレーション情報との差異など
   - 数値ライブラリ上のアルゴリズムに影響を及ぼすパラメタ： 同様の機能を提供するアルゴリズム（ソルバ）の選択、前処理方式選択、ブロック化アルゴリズムにおけるブロック幅、反復のやりなおし間隔など
2. システムソフトウェア上の AT 対象
   - ミドルウェア： 通信ライブラリにおける通信パタンのアルゴリズム切り替えなど
   - コンパイラ： ループアンローリング段数、タイリング幅（キャッシュサイズ）、ソフトウェアパイプライングの実装方式の切り替え、特定の言語やディレクティブが提供する機能の実装アルゴリズムの切り替えなど
   - OS： I/O バッファサイズ（ファイル I/O ストライド幅、メッセージ長）、OS におけるページ管理の単位、ジョブ切り替えの時間間隔など
   - ハードウェア： ハイパスレッドの切り替え、消費電力制御（周波数切り替え制御）など

2.4 階層性のある対象粒度

ハードウェアによるパラメタが先に最適化され、次に OS やコンパイラ階層のパラメタ、最後にアプリケーションによるパラメタが最適化されるというような、計算機科学分類における下位のパラメタが先に決定されるという対象粒度の階層性がある。また、利用時間軸における階層性として、同一のソフトウェア階層における最適化パラメタにおいても、対象アプリケーションのインストール時に最適化パラメタが決定し、次にユーザの情報を提供した時点で定まる最適化パラメタが決定し、最後に、対象ソフトウェアの実行時に最適化パラメタが決定するという階層性もある。

2.4.1 問題レベル

問題レベルの最適化パラメタはアプリケーションごとに存在するため、一般的な記述が難しい。例えば、地震のシミュレーションにおいては、地盤の固さ（密度）に影響を及ぼす地質パラメタが最適化すべきパラメタとなる。観測器が置かれたポイントにおいて、計測された地震波（実測波）とシミュレーションによる地震波（計算波）形状が一致するように、地質パラメタ値を調整する。このとき、実測波と計算波との周期、最大振幅、振幅開始時刻などからなる目的関数を用意して、その差が最も小さくなるようにパラメタを最適化する。地震発生の初期微動から、数秒後に伝わる地震の大きさを予測するシミュレーションなどで適用可能である。

2.4.2 数式記述レベル

上位の問題レベルのパラメタが固定された場合、解くべき数学上の方程式が定まる。一般に、方程式を解く方法は複数ある。数式記述レベルで目的関数を最小にする最適化が、ここでのチューニング対象である。目的関数は、実行時間、必要とする精度、必要とされる記憶量など、ユーザもしくは計算機が要求する指標である。例えば、連立 1 次方程式 $Ax=b$ の解法には、直接解法と反復解法がある。記憶量が十分にない場合は反復解法、十分にありかつ対象の行列が密、もしくは密に近い場合は直接解法が選ばれる。直接解法が選ばれる場合、内積形式ガウス法、外積形式ガウス法、クラウト法の解法などが知られており、それぞれ別の行列操作が数学的に記述できる。このレベルの最適性は、計算機環境に大きく依存する。この数式記述レベルには、MATLAB などの数値計算用言語で記述されたプログラムがあてはまる。チューニング対象としては、記述の冗長性の削減、コピーの削除、よりよい低水準数値計算ライブラリの選択が含まれる。また、内積形式ガウス法などを記述したプログラムから、ブロック形式ガウス法の記述を導出するというようなアルゴリズム導出を含む最適化も、数式記述レベルの最適化に含まれる。

2.4.3 アルゴリズムレベル

アルゴリズムレベルでの最適化は、数値計算ライブラリにおける AT 研究の蓄積のなかに実例が多数ある。ここでは、数値計算アルゴリズムと通信アルゴリズムを取り上げる。数値計算アルゴリズムでは、ブロック化アルゴリズムのブロック幅、反復リスタート周期、前処理方式、直交化方式、行列並び替え方式、行列分解においてゼロとみなせる値、精度を保つために内部に記憶しておく情報のサイズなどが AT 対象の性能パラメタである。通信アルゴリズムでは、通信ライブラリのパラメタとして、通信形態の選択（集中、リング、木、再帰型など）、木構造方式における各ノードの分岐数、同期／非同期通信選択、パケットサイズ、輻輳制御方式選択、通信プロトコル選択（TCP/IP、UDP）などがある。例えば、集団通信方式と 1 対 1 通信方式の選択も、アル

ゴリズムレベルのパラメタとなる。

2.4.4 要素プログラムレベル

要素プログラムレベルでの最適化パラメタは、コンパイラが行う最適化そのものである。AT 機構としてこの最適化を行う理由は、以下の通りである。

1. 高性能なコンパイラが利用できない計算機環境においても、高性能を達成したい。
2. コンパイラが解析不可能なほど複雑な構造をもつプログラムにおいても、高性能を達成したい。
3. コンパイラが理論上解析不可能なユーザ情報を利用しないと、所要の性能を実現する最適化ができない。

要素プログラムレベルで AT 機構が対象とするのは、例えば、以下のパラメタである。

1. ループのアンローリング段数
2. ループのタイリングサイズ
3. ソフトウエアパイプラインにおけるパイプライン段数やパイプライン遅延に起因するパラメタ（例えば、変数の定義と参照の間隔）
4. ループの融合や分離を必要とする実装方式の選択

2.4.5 ジョブレベル

システム全体として高性能を達成するために必要なジョブスケジューリングにおいて、AT対象が存在する。ジョブスケジューリングで必要となる性能パラメタとしては、同時実行数、最大実行可能時間、個々のジョブの優先度設定値などがある。OS によるジョブ切り替えの時間間隔なども、パラメタとなり得る。通信方式レベルでは、通信ライブラリにおいて、どのようなネットワークトポロジを意識して通信処理を行うか、例えば、リング、2 分木、n分木、それらの組み合せなどのトポロジ選択が AT 対象となる。

第 3 章　自動チューニング技術の研究動向

3.1 ソフトウェア自動チューニング技術の誕生

近年の計算機の複雑化およびアルゴリズムの多様化にともない、ソフトウェアの自動チューニング（Automatic Tuning; AT）技術が注目されている。同技術の動向を概観するため、まず AT 技術の生まれた背景、ついで AT 技術の比較の枠組みについて述べる。ついで、比較の枠組みから導出される AT 技術の今後の発展について考察する。考察に際して、AT 技術の適用においてもっとも実績のある行列計算ライブラリ向けの AT 技術を中心にとりあげる。1990 年前後から、計算機の高性能化にともないアーキテクチャが複雑化してきた。主として、キャッシュの多層化、分散記憶と共有記憶によるプロセッサの並列化がその要因である。そのため、行列計算ライブラリでは、性能をチューニングする作業が膨大になった。当該作業の負担を軽減するため、性能チューニングを自動化する技術が研究され、自動チューニングと呼ばれるようになった。行列計算向け自動チューニングの最初の文献 PHiPAC [1]には、直接的に automatic tuning という用語は出てこないが、論文要旨に記された次の文にそれを意味する表現が出てくる。

> Modern microprocessors can achieve high performance on linear algebra kernels but this currently requires extensive machine-specific hand tuning. We have developed a methodology whereby near-peak performance on a wide range of systems can be achieved automatically for such routines.

すなわち、machine-specific hand tuning という状態を克服し、ほぼピークに近い性能を自動的に獲得する方法論（methodology）を開発したという訳である。ほぼ時期を同じくして、データベースの分野でも、AT 技術が注目されるようになっている。データベースシステムの広範囲にわたるパラメタ設定を調整するのに、データベース管理者が多大な時間をかけていることを克服する技術である。行列計算と異なる事情はあるものの、性能に影響をおよぼすパラメタ調整の工数を削減するという目的は共通している。また、両者とも、OS とアプリケーションの中間に位置するミドルウェア階層である。データベースでは、いわゆる Web3 階層での高性能化というシステムインテグレーション上の課題があり、これが行列計算の直面した記憶の多層化およびプロセッサの並列化による高性能実現という課題に相当する。

3.2 自動チューニング技術の比較の枠組み

3.2.1 自動チューニングの定義

代表的な 6 つの行列計算ライブラリ向け自動チューニング技術を比較し、考察する。

比較論を述べる前に、本章で取り扱う自動チューニングの概略的な定義を示す。図 3.1 に、代表的な行列計算である固有値計算における一般的なチューニングプロセスを示す。最初に、実行したい固有値計算のアルゴリズムを選択する。例えば、まず、ハウスホルダ法というアルゴリズムを選ぶ。次に、そのプログラムを記述し、同プログラムをコンパイルして、実行オブジェクトに変換する。最後に、実行オブジェクトを計算機上で実行し、固有値計算の結果を得る。このプロセスでは、性能に影響をおよぼすパラメタの定義すべき組み合せ数が非常に多くなり、さまざまな性能パラメタ、アルゴリズム、記述方法から選択しなければならない。とくに最近の高性能計算機では、それぞれの実行オブジェクトの性能が大きく変わるため、調整すべきパタンの数は非常に多くなる。ここでは、計算内容から実行オブジェクトへの最適な写像の自動化を**自動チューニング**と定義する [2]。

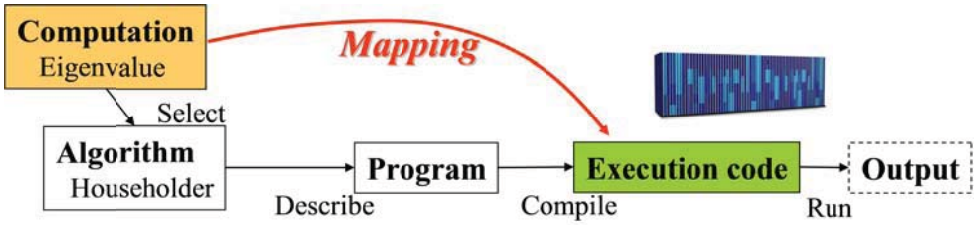

図 3.1 固有値計算のチューニングプロセス

3.2.2 自動チューニング方法の比較

自動チューニングするプロセスの詳細を分析するため、2 つの重要な軸を導入する。1 つの軸は行列計算におけるソフトウェア階層の軸であり、もう 1 つの軸はソフトウェア開発サイクルの軸である。表 3.1 に、行列計算におけるソフトウェア階層の視点を示す。最上位の階層は、方程式を解くという意味をもつソルバ階層である。次の上位の階層は、ソルバのコンポーネントとなるサブソルバ階層であり、行列の何らかの変換という意味をもつ。例えば、ハウスホルダ法による固有値計算においては、3 重対角化がそれにあたる。3 重対角化はある一定の処理の塊であるが、何らかの方程式を解くまでの意味はない。その次の階層は基本線形代数副プログラム BLAS である。これは、行列とベクトルの演算処理を行うという意味をもつ。最下位の階層はループである。これら 4 階層の定義により、AT 技術とコンパイラにおける最適化技術の差異を次のように説明できる。コンパイラにおける最適化技術は、主としてループレベルの最適化情報として、ループ内における変数の相互依存関係を分析する。一方、AT 技術は、行列計算における最大性能を出すため、上記ソフトウェア階層の各々において最適なものを選択する。

表3.1 行列計算におけるソフトウェア階層

ソフトウェア階層	行列計算の例
ソルバ階層	連立1次方程式 $Ax = y$,　固有値 $Av = \mu v$
サブソルバ階層	3重対角化：A（実対称密行列）→ T（3重対角行列） 再直交化：V（ベクトル群）→ V'（正規化ベクトル群）
BLAS	BLAS1　$x = x + y$,　$\alpha = (x, y)$ BLAS2　$y = Ax$,　$y = Ax + A^t x$ BLAS3　$C = AB$,　$A = A - yx^t - xy^t$
ループ	DO　J = 1, 100 　DO I = 1, 100 　Y(J) = Y(J) + A(I, J)*X(I) 　ENDDO ENDDO

ソフトウェア開発サイクルの視点を図 3.2 に示す。計算機上でアプリケーションを作成する際、ソースコードを書き、コンパイルして目的コードを同計算機上で実行し、性能を評価するという手順になる。性能結果を検証した後、プログラミングの方法を改善する。この開発サイクルの各段階において性能に関するパラメタが種々あり、それらを高性能化に向けて調整することになる。自動チューニングでは、従来、人間の手作業による部分も多いこの一連の流れを自動化する。

図3.2 ソフトウェア開発サイクル

ソフトウェア階層とソフトウェア開発サイクルという 2 つの軸で比較する理由は、次の通りである。ソフトウェア階層の軸は、コンパイラの上位への位置づけを意味する自動チューニングの基本の軸である。この軸があるからこその自動チューニングであり、既存のコンパイラの限界を打破するために必須の軸である。もう 1 つのソフトウェア開発サイクルの軸は、チューニングの手順からくる自然な軸である。プログラムを書き、性能を計測し、プログラムのパラメタを制御する。また、性能を計測し、再度パラメタを制御することを繰り返す。このような時間経過を意味する基本的な軸である。

3.3 自動チューニング技術の動向： 6 つの研究事例

■PHiPAC

プログラミング言語 C のループを自動チューニングするスクリプト言語であり、Blimes ら[1]により提案された。行列計算のうち最も基本的な行列 × 行列の演算（DGEMM）の自動チューニングを扱っている。ANSI-C のコンパイラをいかに効果的に利用するかというコーディング技術を集めたライブラリである。PHiPAC では、パラメタ化されたコードジェネレータが、記憶階層に対応する範囲のブロックサイズをもつ C のソースコードを生成する。例えば、mm_gen という行列積のコードジェネレータは、行列積の記憶階層に見合ったいくつかのレジスタ数とブロック数つきのソースコードを生成する。サーチスクリプトは、生成されたソースコードの中で最適なレジスタブロック数（＝アンローリング段数）を定め、次に最適な L1 キャッシュブロック数、最適な L2 キャッシュブロック数と順次定めていく。この順番で最適なパラメタ値を決定できる保証はないが、経験的に良好な結果が得られている。

■ABCLibScript

片桐ら[3]によって提案されたスクリプト言語である。この言語はカーネルプログラムを実行する前に自動チューニングする。図3.3に、ABCLibScriptによって記述されたチューニングの概要を示す。このスクリプトは、多項式などで目的関数を記載することで、適切なループアンローリングを自動化することができる。前述のPHiPACがマシンレベルでBLASのチューニングを行うのに対し、ABCLibScriptでは多項式による最適化モデルを、チューニング仕様を記述する専用言語で与える。

■ATLAS

Whaley ら[4]により提案された自動チューニング機能つき BLAS ライブラリである。Pentium シリーズ、PowerPC シリーズ、Sun SPARC シリーズなど多くの CPU に対し個別にチューニングされた BLAS プログラム集である。ALTAS での AT 技術は、主としてループアンローリングとキャッシュブロッキングである。いくつかの BLAS 関数では、経

験にもとづいたチューニングが含まれている。また,、各計算機へのインストールには、幅広い性能サンプリング情報が必要とされる。しかし、チューニングされた後は、アプリケーションプログラムの開発者は BLAS 関数をチューニングする必要はない。

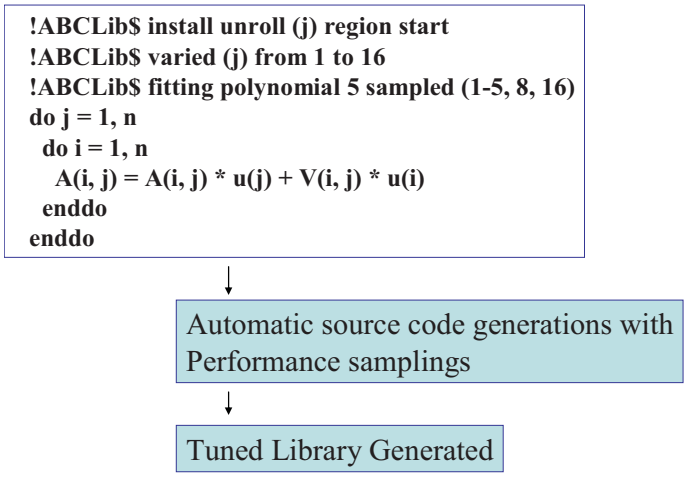

図3.3 ABCLibScriptによる記述例と自動チューニングされたライブラリの生成

■I-LIB

黒田ら[5]により提案、開発された自動チューニング機能つき数値計算ライブラリである。主として、以下 4 つのパラメタをチューニングする。

1. Depth of loop unrolling for matrix-vector multiplications： Depth の範囲は 1、2、 3、 4、 8 など
2. Communication method for matrix-vector multiplications： 主として並列計算機のタイプごとに
3. Gram-Schmidt reorthogonalization： MGS あるいは CGS を選択
4. Preconditioning procedure： いくつかの前処理を実行時に選択

Hitachi SR2201 および SR8000 上での反復解法 GMRES(m)を用いた数値実験では、同等の機能をもつ PETSc に対し、約 4 倍の高速化を達成している。

■FIBER

ソフトウェア開発サイクルを意識した自動チューニングフレームワークであり、片桐ら[3]によって提案された。これは著者らが提案した自動チューニングの定義[1]を、3 つのソフトウェア開発サイクルに拡張した位置づけになっている。FIBER では、性能に影響を与えるプログラム中の調整可能なパラメタを最適化パラメタと呼び、それを以下の 3 つの種類に分類する。

1. インストール時最適化パラメタ
2. 実行前最適化パラメタ
3. 実行時最適化パラメタ

他の自動チューニング方法では、最適化パラメタがインストール時最適化パラメタに限られていた。固有値計算のランク 2 更新の部分では、インストール時最適化パラメタのみでのチューニングに対し、実行前最適化パラメタを適用することで 28. 7%性能向上したと報告されている [3].

■SANS

Dongarra ら[6]によって提案された自動チューニングの枠組みである。数値計算ライブラリを Grid 環境下で管理する要望に対応するよう開発されている。以下の 4 つのコンポーネントからなる。

1. Intelligent agent: Automated data analyzer
2. History database
3. System component: Controlling access to the Grid
4. Metadata vocabulary: User data and performance profile based on common component architecture, CCA, framework

この枠組みでは、データベースとスケジューラがとくにグリッドを意識したコンポーネントとなっている。システムコンポーネントには、実行時適合型スケジューラが含まれており、計算グリッドへのアクセス制御を行っている。CCA 型メタデータ言語によって、計算に使われたデータとアルゴリズムの実行履歴が記録される。

3.4 2 軸による分類と発展の傾向

6 つの AT 技術を上記 2 軸で評価した結果を表 3.2 にまとめる。この表から、3 つの発展の傾向が読みとれる。第1に、自動チューニングの対象が、ループや BLAS などの下位階層からサブソルバを含めた上位階層へ発展しているという傾向がある。PHiPAC は、ANSI C コンパイラのコーディングに対するガイドラインという位置づけである。しかし、そのチューニングをコンパイラで実現しようとすると、コンパイル時間がかかりすぎ現実的ではない。PHiPAC では、1 部の行列演算の意味を変えない範囲で、ループの書き方を変えることでチューニングを実現する。また、I-LIB では、サブソルバを選択することでより高度なチューニングを実現する。このように、PHiPAC から ATLAS、I-LIB となるに従い、徐々に上位階層に AT 対象を移行し、それによって、できるだけ少ないコストで高度なチューニングを行う方向に進展している。

第 2 に、プログラム実装段階での AT から、ソフトウェア開発サイクル全体へ発展しているという傾向が読みとれる。当初の AT 研究では、スクリプトやライブラリというソフトウェア形式を採用していた。最近の新しい研究計画である FIBER と SANS では、AT フレームワークを提案している。このような変化は、AT 技術がソフトウェア開発サイクルをより意識したものになっていることを示している。例えば，疎行列の行列計算ライブラリに対する自動チューニングを検討する際には、密行列の場合と比べて実行時の自動チューニングをより多く考える必要がある。なぜなら、計算性能は実行時の行列データに大きく依存し、従って、フィードバックのある自動チューニングを考えねばならないからである。フィードバックするためには、単にプログラムを実装するだけではなく、性能を評価しさらにその評価の結果、プログラムのパラメタを調整するといった開発サイクルを前提とした考え方を導入する必要がある。そのためにフレームワークが必要となり、上記のような研究動向になっていると思われる。

第 3 に、AT 技術が適用される計算機環境が、単一プロセッサ（CPU）から並列計算機、分散システム、グリッド環境へと発展しつつある。PHiPAC および ATLAS の対象システムは単一プロセッサの性能向上であったが、I-LIB、ABCLibScript および FIBER では高並列計算機 MPP やクラスタシステムを目標にしている。さらに、SANS ではグリッドまでを対象システムに想定している。このように、より幅広い計算機環境を対象とするよう変わりつつある。

表3.2　行列計算を対象とする自動チューニング方法の比較

(P; prepared in detail,　C; considered,　–; not available).

Project Names (Year)		PHiPAC (1997)	ABCLibScript (2003)	ATLAS (1998)	I-LIB (1998)	FIBER (2003)	SANS (2003)
Type		Script		Library		Framework	
Software	Solver	–	–	–	–	–	–
	Subsolver	–	–	–	C	C	–
	BLAS	C	C	P	C	C(ABC.)	C(ATLAS)
	Loop	C	P	C	C	C(ABC.)	C(ATLAS)
Compiling		–	–	–	–	–	–
Running		–	–	–	C	–	C
Evaluating(DB)		–	–	–	–	–	C
Platforms		SinglePE	MPPs,Cluster of SMPs	SinglePE	MPPs,Cluster of SMPs	MPPs,Cluster of SMPs	Grid

参考文献

[1] J. Blimes, K. Asanovic, C.-W. Chin and J. Demmel, “Optimizing matrix multiply using PHiPAC: A portable, high-performance, ANSI C coding methodology,” *Proc. of International Conference on Supercomputing 97,* pp.340-347 (1997).

[2] 直野健, 山本有作, ”単一メモリ型インタフェースを有する自動チューニング並列ライブラリの構成方法,” 情報処理学会研究報告, 2001-HPC-87 (SWoPP2001), pp.25-30 (2001).

[3] T. Katagiri, K. Kise, H. Honda, and T. Yuba, “FIBER: A general framework for auto-tuning software,” *Springer LNCS 2858, The Fifth International Symposium on High Performance Computing (ISHPC-V),* pp.146-159 (2003).

[4] R. Whaley, A. Petitet and J. Dongarra, “Automated empirical optimizations of software and the ATLAS project,” *Parallel Computing, 27,* pp.3-35 (2001).

[5] H. Kuroda, T. Katagiri, and Y. Kanada, “Knowledge discovery in auto-tuning parallel numerical library,” *Progress in Discovery Science, Final Report of the Japanese Discovery Science Project, Lecture Notes in Computer Science 2281 Springer 2002,* pp.628-639 (2002).

[6] J. Dongarra and V. Eijkhout, “Self-adapting numerical software for next generation applications,” *International Journal of High Performance Computing Applications, Vol. 17, No. 2, Summer,* pp.125-131 (2003).

第4章　自動チューニングのための数理的共通基盤

4.1 自動チューニングにおける数理

自動チューニングという言葉のうち、＜チューニング＞の部分は、ソフトウェアにおけるつまみ（パラメタ）を、何らかの意味で結果（目的関数）がよいものになるよう調節することと考えられる。残りの＜自動＞については、いろいろな様態が可能である。しかし、多くの場合において、少なくとも次の4つの処理を、人手の介在ができるだけ少ない形で実現することを目指している。これらは、いずれも数理がかかわってくる余地がある。本節では、これらの4項目それぞれについて、数理の必要性と役割について概観する。

1. 実験：　チューニングパラメタに適当な値を設定し、ソフトウェアを（実際に、あるいはシミュレーションで）動作させる。
2. 測定：　実験において動作させたときの目的関数や、その他の関連するデータを獲得する。
3. 分析：　測定で得られた情報を分析し、目的関数や関連する挙動、チューニングするパラメタ、その他の条件との関連（モデル）を推定する。
4. 決定：　分析において得られたモデルを用い、チューニングパラメタ値に設定する。すなわち、（準）最適なパラメタ値を探索・決定する。

4.1.1 自動チューニングの実験の数理

実験は自動チューニングの基礎である。パラメタに値を与えて実行し、性能を実測するという実験により、パラメタと性能との関連を示す情報を獲得する。このとき、いかに実験するかによって情報獲得の効率が変わってくるので、実験の仕方は自動チューニングにとって非常に重要な問題である。古典的には、実験計画という分野が直接関係している。実験そのもののコストを無視すれば、とりうるすべてのパラメタの組み合せについて、十分な回数、ランダムな順序で実験を行えば、目的関数を最適にするパラメタの組み合せをはじめとする情報が十分にかつ確実に得られる。しかし、一般にこれは現実的ではない。実験のコストを抑えつつ、いかに効果的に準最適解を得るための情報を獲得するかが、**実験計画**の問題である.

実験計画は1種の最適化問題であるが、その目的関数のあり方はさまざまなものがありうる。実験コストをある定数で抑え（制約条件）、その中で得られる情報（いかに最適に近い解が得られるかという指標で評価する）を最大化するという問題設定も考えられる。あるいは、実験により得られる情報から導かれる準最適解の品質（チューニングの結果）をあるレベル以上と制約したうえで、そのような結果を得るために行う実

験コストを最小にするという問題設定も可能である。また、準最適解による実行のコストと実験のためのコストを合計（必要なら重み付けなどはする）して、それを最小化するという問題設定も考えられる。さらには、決定という段階を踏まず、実行時にも実験と測定を繰り返すと仮定し、この実験を含む実行コストの合計を最小にするという問題設定もあり得る。これらの設定は、アプリケーションのあり方によって選択すべきものである。古典的な実験計画をそのまま使うべきかどうか、慎重に判断すべきである。古典的な実験計画では、実験コストを実験回数で置き換えていることが多い。しかし、一般に実験コストは一定値ではなく、しかも事前に知られている訳でもない。古典的にはこれらの情報は予備実験で収集するのであるが、予備実験で得られた情報を最適化に利用していないという問題点がある。

実験計画はモデリングに強く依存することに注意が必要である。例えば、連続値をとるパラメタxに対して目的関数 $f(x)$ が2次関数であることが分かっていれば、適当な3点 $\{x_1,\ x_2,\ x_3\}$ をとり、$f(x_1),\ f(x_2),\ f(x_3)$ を観測し、2次関数で補間すれば最適解が得られる。また、多数のパラメタ $x_1,\ x_2,\ \cdots,\ x_n$ があったとき、目的関数が $f(x_1,\ x_2,\ \cdots,\ x_n) = f_1(x_1) + f_2(x_2) + \cdots + f_n(x_n)$ のように表されれば、各パラメタに関して独立に最適化すれば大域的な最適解に一致する。このように、適切なモデルがあらかじめわかっていれば、実験と最適化が効率的に行える。しかしながら、モデル誤差や測定のばらつきを考慮しようとすると話は簡単ではない。チューニングにおけるパラメタには、離散的なものと連続的なものがある。離散的なものには、量的な意味をもつもの（ループアンローリング回数など）と、名目的な違いしか表さないもの（アルゴリズムの名前など）とがある。量的な意味をもつ離散変数は、連続量に拡張して考えることで便利に扱えることが多い。

4.1.2 自動チューニングの測定の数理

測定とは、設定したパラメタ値で実行した際の、性能に関する、あるいは、それに付随する情報を獲得することである。目的とする性能そのものが重要な測定対象であるが、関連するさまざまな情報（キャッシュミス・ヒット率などの性能情報、平均負荷量などのシステム情報、ライブラリの測定における呼び出し引数などの環境情報、反復法における入力行列の優対角性など計算を要するものなど）も対象たり得る。欲しいデータをどのようにして測定するかという問題は、数理ではなくシステムの問題である。しかし、一般には測定したいものが直接正確に測定できるとは限らない。電力や温度などの物理的な測定には、ばらつきや測定誤差が避けられない。所要時間の場合、測定方法によって精度に限界があるほか、OSの割り込みなどによるばらつきもある。チューニングするパラメタを変化させた直後には、キャッシュなどの状態が異なり、しばらくの間定常状態とは異なる性能挙動を示す可能性がある。さらには、温度などにより水

晶の発振周波数も変化するので、絶対時間という意味では誤差要因となる。このような擾乱要因がある場合には、測定において次の2つの数理が必要となる。

1. 擾乱の大きさを分析し、測定値にどれだけの誤差が混入しているかを定量的に推定することが重要である。擾乱の大きさは、パラメタやその他の条件に依存する場合があり、その依存性が定量的にモデル化することができれば有用性が高い。
2. 擾乱の特性を分析し、測定値から擾乱要因を除去して、本来求めたい値に近づけることが有用である。このためには，事前知識または測定結果から擾乱の入り方を分析し、擾乱を差し引くことも考えられる。また、単に多数の測定の平均をとることにより、ランダムな擾乱の成分を相対的に小さくすることも含まれる。他の情報からそれぞれの測定値に含まれる擾乱の大きさを推定できれば、擾乱が小さいと推定される測定値を重視するなどの方法で擾乱の影響を抑えることもできる。

他方で、アルゴリズムの反復回数など、誤差がまぎれこむ余地がほとんどないものもある。これらについては、とくに測定が問題になるとは考えられない。測定の数理に最も関係が深い古典的分野は、統計的データ解析や時系列解析である。

4.1.3 自動チューニングの分析の数理

チューニングにおける最適化が可能となるためには、チューニングするパラメタや関連する条件パラメタの値と、目的関数との数学的な関係についての知見が必要である。また、目的関数に関連する様々な指標についても、同様の数学的関係の知識が有用であることが多い。以下では、これらの数学的な関係を**モデル**、モデル構築のための数理的処理を**分析**と呼ぶ。分析は、大きく2つの問題に分けることができる。第1の問題は、分析対象の指標とパラメタとの関係を記述するためのある種の仮定である。例えば、n次正方行列の積の所要時間はnの3次関数で表される、あるいは、並列プログラムの所要時間は各プロセッサでの所要時間の最大値で表される、といったものである。逆に、パラメタと所要時間の間に特定の数学的関数関係を仮定しない、というのも仮定の1つである。これらはそれぞれの関数の具体的な中身を明確にせず、数学的な構造だけを示しているのが特徴である。モデリングという言葉は、狭義にはこの問題を指す。モデルは最初から1つに限っておく必要はない。むしろ一般的には、複数のモデルを準備しておき、実験結果からどれが適切であるか評価し選択する方がよい。これは統計で**モデル選択**と呼ばれている問題である.

第2の問題は、数学的な構造を固定したときに、具体的で定量的な関数形を求めることである。これは、モデルが含む未知パラメタの値を推定するという形で定式化される。

フィッティングと呼ばれることが多い。特定の構造をもつモデルを仮定する場合には、必ずしもそれが真の数学的関数関係である必要はない。むしろ、完全には一致しないということが一般的である。このため、多くの場合、フィッティングは、モデルと実測の間の乖離（モデル誤差）を最小にするという最適化問題となる。フィッティングを最適化問題とみなすと、目的関数が何かというのは根本的な問題である。多くの場合、最小2乗法などの安直な手法が用いられているが、これは必ずしも適切ではない。例えば、最適解の付近でモデル誤差が小さいほどよい結果が得られると期待されるから、最適に遠いところでのモデル誤差を犠牲にするということが考えられる。しかし、最適解の付近でモデルと実性能との一致がとくに悪い場合には、最適解の付近で無理にフィッティングしようとすると、全体としてフィッティングの精度が極端に低下する。このように、フィッティングの目的関数をいかに設定すべきかという問題は自明でなく難しい。また、フィッティングにおいて、制約条件を考える必要がある場合もある。例えば、所要時間は負になることはないが、モデルを構築すると負の推定値を与えてしまうことがある。あるいは計算量が増えているはずなのに、モデル上では所要時間が短くなってしまうことがある。このようなモデルを不自然であるとして排除するか、それとも受け入れて利用時に工夫するかという選択も、自明ではない問題である。当然、モデル誤差の大きさと何らかのパラメタや指標の間に関数関係がある場合も考えられる。この場合は、モデル誤差のモデル、さらにはその誤差を考えることができるし、そのような情報が得られれば有効に活用することができる。しかし、モデル誤差に一定の傾向があれば、モデルにとりこむことでモデルの精度をあげることができるから、そのような一定の傾向が誤差に入ってくることは期待しにくい。このためか、モデル誤差の定量的モデル化という既存研究はほとんど見あたらない。

4.1.4 自動チューニングの決定の数理

決定は自動チューニングの最終ステップであって、これ以降はパラメタの変更は行われない。それまでに得られた情報を総合し、パラメタに具体的な値を代入して、できるだけ最適に近い状態を実現する。自動チューニングはある種の最適化である。一般的な最適化の数理からの類推により、制約条件と目的関数の2つの概念が基本的であることが分かる。**制約条件**は、変数が満たさなければならない条件である。もっとも基本的なものはチューニングするパラメタがとりうる範囲である。パラメタがとりうる範囲が、他の変数によって変化する場合も多い。例えば、使うクラスタが異なればプロセッサ数パラメタがとりうる範囲が異なる。これとはやや異なるが、変数 a の値によって変数 b の存在そのものが依存する場合もある。例えば、アルゴリズムAに含まれるパラメタbは、アルゴリズムAが選択されなければ意味をもたない。数学的には目的関数 f の値が変数 b に依存しなくなるとして扱うこともできるが、ソフトウェア的には変数 b を入力できなくなることから、やや扱いが違う。より複雑な制約条件として

は、記憶量などの計算機資源の割り当てや、ハードリアルタイムのような絶対に満たさなければならない制約条件が考えられる。ソフトリアルタイムの場合には、それを満たさないときはペナルティとして目的関数に組みいれることにより数学的に処理できる。**目的関数**は何らかの意味でのコストとして、それを最小化するのが多くの場合よい定式化である。ばらつきや不確定性がある場合、コストの期待値を目的関数にするべきか統計量にするべきかは、実際に使われる場面によって異なる。目的関数を複数の関数に分解して考える場合には、注意が必要である。例えば、並列処理を行う場合には、p 個のタスクの実行時間の最大値の期待値が目的関数になると考えられるが、これは p 個のタスクの実行時間の期待値の最大値とは異なる。

実験と測定をある時点でやめて、その後は実験もデータの収集もしないとしても、対象のソフトウェアが実行されるときの環境や条件が変化する場合には、最適化のタイミングがいくつか考えられる。例えば、ある時点でパラメタを固定してしまう方式、実行するごとにパラメタを設定する方式などである。実行するごとにパラメタを設定する方式では、実行ごとに最適なパラメタ値を探索・計算する手間が無視できない。この手間を短縮するためには、実験・測定で得られた情報のデータ構造としての表現方法と、そこから最適解を求めるアルゴリズムの工夫が重要である。ある時点でパラメタを固定してしまう方式の場合、その後の実行条件・環境とその分布を想定して目的関数を設定することになる。そこで想定した実行条件・環境の分布が実際の分布と異なる場合には、実際には最適解になっていない。このような場合には、さらに利用条件の分布のばらつき（ハイパ分布）を仮定して、目的関数の期待値を評価するという対策が考えられる。一般に、パラメタを固定してしまうよりも、実行時（実行直前）に得られる情報を用いてパラメタを設定するほうが、よいパラメタを選択できると考えられる。ただし、実行時に最適値を求めるコストとパラメタを変更する際のコスト（データ構造やデータ分散の変更などで顕著）を考慮すると、必ずしもパラメタを固定する方法が悪いということはできない。さらに、目的関数が実行前には分からないパラメタや、観測できない（していない）パラメタに依存する場合もあるので、実行時にパラメタを設定することがいつでも十分であるとはいえない。

最後に、決定というプロセスは、実験・測定がこれ以上行われないことを意味しているが、ソフトウェアにおいてこれは必ずしもいつでも自然なことではない。計算機ではパラメタの設定、測定、情報の蓄積（追加）といったことがさまざまな段階で可能であることが多く、それに応じてさらなる最適化が可能と考えられる。このような場合には、決定というプロセスは存在せず、実験・測定のコストを目的関数として最小化することが自然である。しかし、組み込みシステムやゲーム機など専用ハードウェアで動作をするソフトウェアの場合には、事前に実験・測定を完了して、パラメタを固定するのが適

切と思われる。また、システム構築のためのベンチマーキングなどでも、一定の最適化を行ってパラメタを固定したうえで、性能を評価する必要がある。

4.2 数理的共通基盤の現状と問題点

前節で述べた自動チューニングのための数理について、これまでの自動チューニングの研究においてどのように扱われているかを問題点とともに概観する。

4.2.1 実験計画

特定のソフトウェアの自動チューニングでは、モデルが用いられることが多い。定量性のあるモデルをあらかじめ仮定するか、自由度のあるモデルを準備しておいて、何らかの方法でフィッティングし、性能が高いと見込まれるパラメタ値の組み合せだけで実験し評価するのが一般的である。実験対象とすべきパラメタの範囲は、モデル誤差の大きさを定量的に評価することで与えられた危険率を満たすように決定できるはずである。しかし、多くの場合、モデル上で最適解からある定数係数までの性能となるもの、あるいは、性能の高いものからある定数個までのように選択されている。モデル誤差と測定のばらつきの定量評価は、多くの場合ともなっていないが、これは暗黙のうちに実装時の実験で推定しているといえる。

汎用自動チューニングには、モデルが用いられていないことが多い。 ABCLibScriptでは、基本的にはすべての可能性を一度ずつ調べている。モデルを構築する場合には、実験に用いるパラメタ値をユーザに指定させている。この他、機械学習やNelder-MeanのSimplex法などの汎用手法で最適化をするものが多い。これらは明示的に数学的モデルを仮定しておらず、効率や最適性を明確に議論できない。統計的な手段を用いるものとして、直交表などの古典的実験計画、Bayes統計による最適逐次実験計画などの研究がある。また、目的関数の分布を推定することにより、探索を効率化する試みもある。測定のばらつきに対応するため、測定を繰り返して標本分散を計算し、それが閾値よりも小さくなったら反復を停止する手法が用いられる場合があるが、これは統計的には正しくない。正しいのは、予備実験で分散を評価し、その情報から必要な測定回数を決定する方法（Stein の2段解法）である。また、既存の自動チューニングの研究において、測定データを数理的に処理したという話はほとんどない。キャッシュをフラッシュしておくなど、測定のばらつきを少なくするための工夫をしていることがある。

4.2.2 モデル構築

性能のモデル化については、多数の既存研究がある。多くの場合、アルゴリズムを分

析して得られる計算量や通信量を基本として、衝突などのオーバヘッドを加味してモデルを構築する。得られたモデルの誤差についても、議論されていることが多い。しかし、これらのモデルは人間が有効性を確認しており、人手を離れて動作する自動チューニング機構にそのまま組み込むことが適当かどうかは、慎重に判断する必要がある。有効性の自動的な判定と、有効でないと判断された場合の処置の方法を準備しておくなどの工夫が必要である。モデル構築の自動化の研究は、自動チューニングにおいてはほとんどない。また、自動チューニングのために有利なフィッティングについて論じた例も見あたらない。

4.2.3 最適化

自動チューニングにおいて、目的関数がないということはない。しかし、個別ソフトウェアの自動チューニングにおいては、目的関数が明確でないことがある。例えば、ATLASのような行列計算の場合、行列サイズによって最適なパラメタが異なるが、どのような行列サイズ分布を仮定して最適化してあるかが明らかとなっていない。目的関数とユーザが実際に利用する際のコストとが一致しているかどうか、つまり設定した目的関数が適切かどうかという議論も行われていない。多くの既存研究では、そのソフトウェアがどのような条件で高い性能を出すかを示すことにより、ユーザがそのような条件でソフトウェアを使うよう求めているように思われる。しかし、汎用の自動チューニングにおいては、目的関数は必須である。目的関数を定義するとき、場合によっては、複数の評価値の組み合せを考える必要がある。以下の組み合せ方法が考えられる。

1. ある評価値が一定以上の水準となることを制約条件として課す。
2. 複数の評価値の重み付け和などにより1つの目的関数にする。
3. Pareto最適解を出力してユーザに選択させる。

パラメタの違いによる性能値のばらつき（性能安定性）も評価指標の重要な要素と考えられる。しかし、自動チューニングの枠組みでは、このような視点での研究が十分進んでいるとはいい難い。

4.3 自動チューニングのための数理的共通基盤に関する課題提案

4.3.1 高精度な性能モデルのためのモデル選択と自動的モデル改良

自動チューニングにおいて、性能モデルが果たす役割は大きい。適切なモデルが与えられれば、最適なパラメタが確実に求められるほか、そのために必要となる実験の数が大幅に削減できる。このため、高精度なモデルが得られれば、実験コストと実行コストの両方を効果的に削減することができる。しかしながら、高精度な性能モデルを

構築することは、必ずしも自明なことではない。その原因の1つには、計算機の多様性と複雑さがあげられる。多層キャッシュやout-of-order実行などのプロセッサ上の工夫は、平均的な性能を向上させるのには大いに役立つが、個々のソフトウェアの実行性能を予測することを困難にさせている。しかも、これらの特徴は計算機ごとに異なるため、モデルの構築が一段と困難となっている。

これに対し、本課題ではモデル選択を用いて性能モデルの高精度化を目指す。一般に、モデルが複雑になると表現力は増大するが、フィッティングは必ずしもよくならない。その1つの理由は、十分なデータがなければ複雑なモデルが含む多数のパラメタを十分な精度で決定できないからである。このため、得られているデータの量により、適したモデルの複雑さが異なってしまう。別の理由としては、本来は存在しない項や、実際には効果が小さく無視できる項がモデルに含まれている場合、測定の誤差やばらつきがこれらの項にまぎれこみ、モデルの精度を悪化させるということがある。

本研究では、統計的な分析手法を用いることにより、いくつかのモデルの候補から手元のデータにもっとも適したモデルを推定するモデル選択により、複雑なモデルから不要な項をとり除き、モデルによる性能推定の精度を向上させる。さらに、このモデル選択の手法を発展的に用いることにより、もともとのモデルに含まれなかった項を機械的に生成し、それらを追加して精度が向上するかどうかを分析することにより、自動的にモデルを改良することを目指す。本課題を解決することにより、これまで適切なモデルが得られていなかった複雑な挙動を示すソフトウェアに対して、性能モデルを構築することができるようになる。その結果、自動チューニングの適用範囲が飛躍的に拡大し、多様なソフトウェアが高い効率で実装できるようになり、システム全体としての計算効率が格段に向上するものと期待される。

4.3.2 縮小推定を用いた性能モデルの精緻化と挙動安定化

自動チューニングの実験コストの削減を進めると、もっとも高い性能をもたらすパラメタ値以外での実行がほとんど行われず、得られる情報が非常に限られるようになる。その結果、性能モデルの精度がさがり、安定して高い性能を実現することが困難となる。例えば、大多数の場合には高い性能を示すが、ある割合で極めて不十分な性能にとどまってしまう危険性が高くなる。Bayes統計を用い性能モデルの精度が適切に評価できれば、情報を得るために払う適切な実験コストが定量的に求められる。しかし、モデルの誤差を高い精度で評価することは、モデルそのものの精度を確保するよりも格段に難しい。大量の情報を集めないと、モデルの誤差を評価するのは困難となる。

これに対し、**縮小推定**は、独立な異なる対象での測定結果を参照することにより、極めて少数の測定結果しかない場合でも平均的な推定精度を改善することができる統計的な手法である。独立な対象から情報を得るという縮小推定は一見すると不自然な主張のように思われるが、Bayes統計との密接な関連が明らかになってきている。このため、Bayes統計を基礎とする自動チューニングのための数理基盤に適合することが予想される。本課題を解決することにより、性能モデルとその評価の精度と安定性が格段に向上する。それにともない、Bayes統計にもとづく自動チューニングの有効性がさらに高められ、多様な計算が効率よく実現できるようになると期待される。

本課題では、以下の手順で研究を進める。前提として、ソフトウェアの実行時間は、ハードウェアの基本性能とソフトウェアの計算量から第1近似が得られる。従って、ソフトウェアとしては一見非常に異なるものからでも、有益な情報が得られる可能性があることに注意する。

1. ソフトウェアの性能モデルとその誤差の推定に有効な縮小推定の手法を開発する。
2. 有効な縮小推定を実現する高速なアルゴリズムを開発する。ここでは、縮小推定の精度と演算量の異なる複数のアルゴリズムを準備することにより、場合に応じて適切なアルゴリズムが選択できるようにする。
3. 提案した手法をシステムとして実装し、自動チューニングを実現して評価を行う。評価の対象としては、自動チューニングが早くから研究されている数値計算のほか、離散アルゴリズム、最適化、並列処理などを想定する。

4.3.3 自動チューニングのための数理基盤ソフトウェアの構築

自動チューニングにおいて、統計を中心とする数理的技術は極めて重要である。しかしながら、これまでの自動チューニングの研究の多くでは、特定のソフトウェアのチューニング技術に焦点が集中していて、自動チューニングそのものの効率や数学的背景については十分に検討がされてこなかった。このため、多くの研究で実験のために非常に高いコストを払っているなどの弊害も見られた。本研究では、自動チューニングのために必要な数理技術の確立を目指し、得られた数理的基盤に関する知見を数理基盤ソフトウェアという形でまとめて一般に提供する。自動チューニング機能を有するソフトウェアを構築しようとするプログラマが、これらの数理的な結果を簡単かつ適切に利用できるようにする。具体的な数理を解明する研究課題は、以下の通りである。

1. 目的関数と制約条件の適切な設定
2. 性能モデルの高精度化とモデル誤差の定量的評価

3. 時系列的な特性を含む測定の誤差やばらつきを考慮した測定データの解析
4. 最適パラメタを探すための情報を少ないコストで得る実験計画

これらはいずれも簡単な課題ではないが、いずれの問題についても、Bayes統計を核とする数理的手法が有効であることが知られている。これを糸口として、各課題に取り組み、5年程度をかけてある程度の知見をまとめた結果を、数理基盤ソフトウェアとして構築する計画である。本課題が提案する数理基盤ソフトウェアが実現すれば、確たる数理基盤にもとづくソフトウェア自動チューニングが容易に実現できるようになり、多様な計算機環境で安定して効率的に動作するソフトウェアが構築できると期待される。

第 5 章　自動チューニング機構とシステム

5.1　自動チューニング機構のシステム化

自動チューニング（AT）機構は、アプリケーションソフトウェア、ライブラリインタフェース、コンパイラ、通信ライブラリ、OS の各システム階層について、それらの自動チューニング機能を実現するための機構である（図 5.1）。AT 機構は、システム性能に影響を及ぼす性能パラメタ（最適化パラメタ）の自動チューニングを行う。チューニング対象の選択候補となるコード群、アルゴリズム集合、実装方式、スケジューリング方式などについても、パラメタと呼びチューニングの対象とする。これらの適切な選択が、対象アプリケーションの実行において高性能をもたらす。性能パラメタとして AT 機構内でパラメタ化され、その他の性能パラメタと同様に最適パラメタ値の選択が行われる。

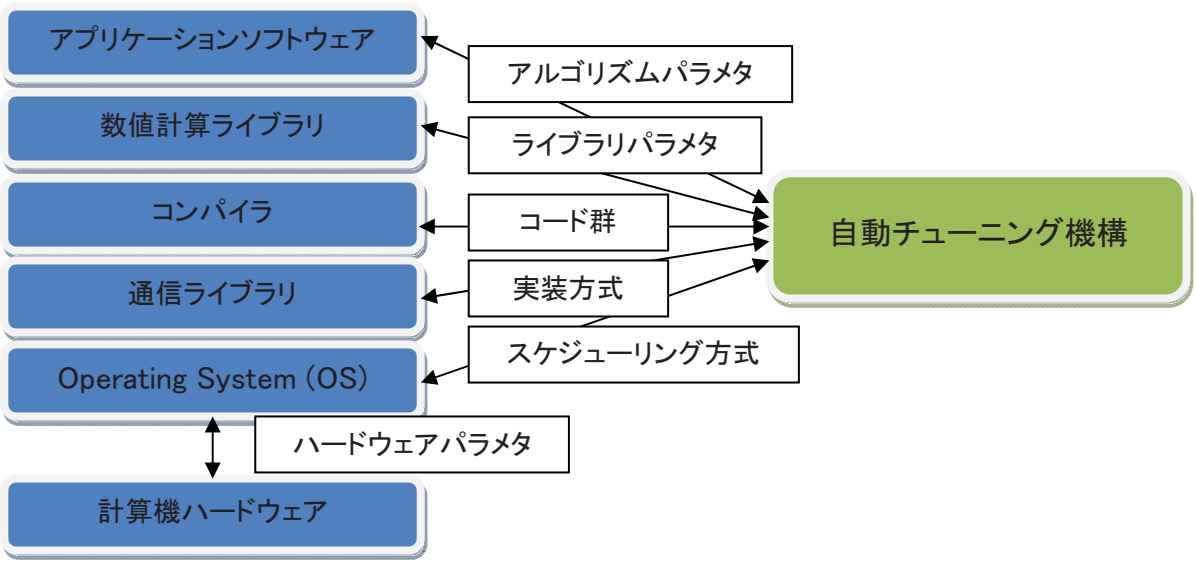

図 5.1　自動チューニング機構のシステム化

AT 機構を実現するために、外部とのインタフェースが必要となる。例えば、モニタ機構を実現するために、OS による計算機環境の情報（使用記憶量、CPU 負荷など）を取得する場合がある。また、場合により、実行ジョブの強制終了の機能が AT 機構の実現には必要とされる。その結果、ユーザプログラムと OS との間のミドルウェア階層において、外部システムを利用するためのアプリケーションプログラミングインタフェース（Application Programming Interface; API）が必要となる。ハードウェアに近い下位層で AT 機構を実現するには、API を規格化することで、上位層の AT に必要なユーザ情報を取得する工夫がいる。この実例として、AT 機構付き数値計算ライブラリにおい

て、ユーザが所有する行列の形状（ブロック、帯など）を与える API の導入があげられる。

5.2 システム化とその事例

AT 機構をシステム化するにあたり、アプリケーションソフトウェアに近いほど多くの事前情報が得られ、それを利用することで自動チューニングの効果が大きい。一方、アプリケーションソフトウェアに近いほど、AT 機構として提供する機能に汎用性がなくなる。すなわち、事前情報と汎用性を鑑みて、適切な階層で、適切な機能の AT 機構を作ることが重要である。AT 機構をシステム化するにあたり、数値計算ライブラリ、通信ライブラリ、言語処理系、OS の各システムソフトウェアレベルにおいて、AT 機構がシステム化されてきた。AT 機構を実装するプログラムは、ユーザレベルに属するもの、システムレベルに属するもの、ユーザレベル、および、システムレベルの両方に属するもの、の 3 通りで実現可能である。また、AT 機構を実装するにあたり、AT 機構を対象プログラムや対象ソフトウェアに付加させる方法として、以下の方式がある。

1. 専用言語を利用しユーザプログラムに付加させる方式（ABCLibScript 方式）
2. OS のデーモンとして AT 機構を常駐させ、割り込みなどの OS 命令の 1 つとして AT 機構を利用する方式（AutoPilot 方式）
3. AT 機構が組み込まれているライブラリとして配布する方式（ATLAS 方式）

今まで、システム化に成功した事例を以下に示す。

5.2.1 数値計算ライブラリ

数値計算ライブラリに AT 機構が実装され、多くの成功例がある。これは、**ライブラリ階層**がアプリケーションソフトウェアとシステムソフトウェアの中間に位置し、適切なユーザ情報を獲得し利用することによって、汎用性の高い AT 機能が提供できたことによる。密行列、疎行列、直接解法、反復解法などの数値計算ライブラリにおいて、AT 機構が導入されている。一般に、ユーザ知識が導入不要な密行列、直接解法ほど、AT 機構のシステム化をしやすい。一方で、疎行列、反復解法ライブラリにおける AT 機構の導入には、多くの未解決の課題がある。

5.2.2 通信ライブラリ

通信ライブラリにおける AT 機構の実現においては、ライブラリインストール時において通信トポロジを考慮した最適な実装方式が選択できる AT 機構がシステム化されている。一方、通信ライブラリにおいて、実行時に最適な実装方式を選択する AT は現在も挑戦的な課題であり、研究レベルでも十分な AT 効果が得られていない。

5.2.3 言語処理系

コンパイラなどの言語処理系においては、コンパイル時に実行して性能プロファイルをとり、それをもとに再度コンパイルする方式で AT 機構が実現されている。しかしながらこのような機能は研究段階であり、実現された商用コンパイラはない。AT 仕様を記述する言語を規定し、それを処理する言語処理系を導入するという方式が研究されている。この方式は既存の計算機言語に AT 機構を自動付加する方式であるが、この方式のほうが実用段階に近い。

5.2.4 OS (Operating Systems)

OS においては、デーモンの常駐プロセスに AT 機能をもたせることで AT 機構をシステム化する方式が一般的である。研究レベルであるが、AutoPilot や Active Harmony などのシステムが実現されてきた。これらのシステムの主たる最適化対象は、システムパラメタである。例えば、ファイルアクセスで性能に影響する入出力バッファサイズ（ストライピングサイズ）の実行時の変更などが、主な AT 機能としてあげられる。いすれも、実行時（動的）に自動チューニングが行われる。

5.3 システム化の問題点

5.3.1 ユーザの関与を想定した半自動化

システム化の現状を見るに、AT のすべての手順において全自動 AT システムを構築することは難しい。エンドユーザ、ソフトウェア開発者の知識や指示をとりいれた半自動化が、AT ソフトウェア構築の際に行われている。AT 機構の全手順において、ユーザからのわかりやすい指示を行える支援ツールが存在しない。

5.3.2 自動チューニングポリシの指定

ソフトウェア開発者、もしくはエンドユーザが設定した AT ポリシについて、自動的に適切な AT を行う方式が確立されるべきであるが、現状ではこのような方式は開発されていない。AT ポリシを設定する API や、そもそも AT ポリシ指定に必要なパラメタ自体がまだ確立されていない。

5.3.3 実行性能の予測

1. 性能安定性の改善： 任意の AT 対象プログラムについて、性能安定性を測定するような尺度が存在しない。また、性能安定性を改善するような、一般化された手法が提案されていない。
2. 目的関数の定義可能性と最適化の誤差検出： 任意の AT 対象プログラムに

ついて、適切な目的関数が定義できるかどうかの評価基準が存在しない。適当な目的関数を設定した場合でも、どれだけ最適な目的関数の結果から悪くなるかの誤差検出手法が確立されていない。

5.3.4 計算量の爆発

AT ライブラリのインストールであっても、長くて 1 日程度で AT 処理は終わるべきである。ユーザにより、最適化時間の許容度が異なる。AT に費やす時間の許容度を設定できる機能が提供されることが望ましい。

5.3.5 自動チューニングインタフェースの標準化問題

AT インタフェースの標準化には、**自動チューニング記述言語**の標準化と、AT の API 標準化という 2 つの側面がある。AT 言語処理系が出力するコードに、標準化した AT の API を入れこむことによって、汎用性を高めることができる。以下の典型的な AT の API について、手順と形式が共通化されるべきである。

1. 自動チューニングする最適化パラメタの指定
2. 自動チューニングのタイミング指定
3. モニタ機構の起動指定
4. 目的関数（コスト定義関数）の定義方法
5. 自動チューニング対象候補の蓄積機構の構造
6. 学習機構の起動指定
7. AT データおよびデータベースの形式
8. 学習機構が提供する機能
9. AT ポリシの指定

5.4 自動チューニング機構とそのシステム化に関する課題提案

5.4.1 自動チューニング記述言語の標準化

AT 記述言語が提供する AT 機能は、何らかの標準化がなされることが必要である。例えば、統一モデリング言語 (Unified Modeling Language; UML)で、AT 機構の記述ができることが望ましい。その UML 記述された AT 機構が、さらにメタな AT 機能として UML で定義されるとよい。そのメタな UML を AT 標準機能インタフェースとするような、階層的な定義が可能となる枠組みを実現する。AT 機構をソフトウェアとして実現する場合、ソフトウェア開発時に使われる AT 機構付加専用の記述言語を開発し、AT 機構の付加を効率化できることが必要である。通常の記述言語の要求に加え、以下の機能が実現されることが望まれる。これらはいずれも、AT システム化研究における重要

な課題である。

1. AT 機構による性能向上の基準を示す性能パラメタを指定できる。
2. 仕様設計→コード実装→機能検証→性能検証→仕様設計というソフトウェア開発サイクルの各段階において、その都度定義が変わる AT 要求性能への対応が可能である。
3. 実装されている AT 機構のモジュールが、他のモジュールと区別されており、要求に応じて削除・追加・変更ができる。
4. AT 性能を検証する機構と手段が容易に与えられ、かつ、検証結果による AT 要求性能の変更を AT 記述にフィードバックできる。
5. AT 機構の付加工程や実証工程が、ソフトウェア開発全般の内訳として、定量的に見積りができる。
6. AT 機構を付加することによる弊害（記憶量増加など）や開発コストの見積りが、AT 記述言語から可能である。

5.4.2 自動チューニング機構のシステム化の方法論

AT 機構のシステム化に際して、システム化の有効性が事前に評価できることが望ましい。個々の AT 方式自体の有効性を事前評価できる、もしくは、工学的に妥当な AT システム化が構築できる方法論が必要である。有用な AT ソフトウェア構築の方法論を明らかにして、生産コストを指定できるソフトウェアの開発方式を提案する。ソフトウェア開発に際して、所要費用、開発期限、開発工数、計算機資源（実行可能時間、記憶量）、製品品質（演算性能、精度）などの制約条件をもとに、実現されるシステムの実行性能、精度、記憶量などが最適化される。同開発方式において、要求仕様は AT ポリシで定められる。AT ポリシをユーザが設定できる AT 方式を実現する。

ATシステム化に際して、段階的な機能の高度化を可能とする仕組みの導入をはかる。AT 記述言語により付加された AT 機構は、ソフトウェア開発の各段階において、要求に応じ段階的に AT 機能が高度化できる手段が提供されていることが望ましい。例えば、初期設計段階においては、インストール時最適化機能のみ必要と見積もられていたが、性能検証段階において実行起動前時最適化が必要とされる場合を考える。この場合は、実行起動前時最適化機能を追加しても、ソフトウェア開発全体として少ないコストで機能の高度化ができることが必要となる。

並列プログラミング環境 MPI、OpenMP、およびそれらの複合環境で、AT 機構がシステム全体の最適化におよぼす影響について、理論的かつ実測的観点から評価する必要がある。このような並列プログラミング環境を統合できる環境を考慮し、そうした

中で有用性をもつ AT 機構の実現をはかる。

5.4.3 自動チューニングの枠組みにもとづくソフトウェア開発方法論の展開

AT は性能チューニングの手間を削減するという意味で、ソフトウェア生産性を高めるための技術と位置づけることができる。ソフトウェア生産性を取り扱う情報技術は、**ソフトウェア工学**として知られている。AT 技術とソフトウェア工学の関係は深い。AT 機構自体もソフトウェア生産性を考慮した構造になっていること、すなわち、AT 機構そのものが、ソフトウェア工学上の方法論を基礎にしていることが期待される。現在のところ、AT 技術がよく適用されているのは数値計算分野であり、同分野が今後もっとも影響をおよぼすと予想されるのは計算科学分野である。計算科学におけるソフトウェア開発プロジェクトには、以下の7つの開発段階があるとされる。AT ベースのソフトウェア開発方法論への展開を目指し、各開発段階に対応させて、以下に示す AT ツールを開発する。同ツールを用いて、AT 機構を半自動的に組み込む AT ソフトウェア開発環境を構築する。

1. 質問と問題の定式化
   AT仕様策定を支援する。決められた仕様について、ソフトウェア開発者、ソフトウェア利用者（計算機管理者、アプリケーションユーザ）の誰が、当該仕様におけるどの最適化パラメタを自動チューニングするかの決定を支援する。
2. 計算科学的アプローチ
   一般的な意味でのプロジェクト管理を支援する。開発対象に現れる最適化パラメタについて、AT を適用する場合に適用可能な AT 記述言語、AT 方式、計算機資源選択の決定を支援する。
3. コード開発
   テキストエディタ、コード可視化、および、ソースコードからわかるプログラム構造情報を自動的にドキュメント化など、コード開発を支援する。開発者レベルに応じ、最適化パラメタや目的関数の情報が陽に示される表示インタフェースが考慮される。コンパイラ性能をあげるため、コンパイラ最適化を行いやすいコードを自動生成する機能、および、開発者がコンパイラの最適化を効果的に用いられるように、プログラム上の適する場所に AT 機能を自由に追加・削除することを支援する。一般的なコードのバージョン管理ソフトウェアの他に、AT 機構のコードへの付加が考慮される機能が必要である。
4. 検証と有効化
   AT 効果が検証できるテストツールが必要である。このテストツールには、性能検証に必要なテストデータが自動的に設定される。AT 性能が仕様を満たない場合、コード開発者の知識をもとに、半自動的に AT 対象を拡張・縮小で

きる。場合によっては、1 や 3 の段階に戻れる仕組みが必要である。

5. 開発ソフトウェアの実行
   開発された AT ソフトウェアを、ユーザの環境にインストールするのを支援する。このとき、各 OS に対応した AT 用インストーラの実装が必要である。AT ソフトウェアのインストールの手間を減らすために、既存のインストーラ（例えば、Windows 用）に容易に開発中の AT ソフトウェア用インストーラを組み込むツールや GUI の提供が必要である。
6. 実行結果の解析
   インストール後の AT ソフトウェアをさらに最適化するため、インストール時、ユーザの指定時、または実行時のチューニングデータを自動的に蓄積しデータベース化する。蓄積されたデータベースを、アクセス制限された局所的、もしくはインターネット上の大域的に公開する。得られたデータについて、ユーザに必要なデータを選別して提供する機能、選別されたデータに対してコードやアルゴリズムを最適化する機能、ユーザに対して自動チューニングの可能性を指示または最適化の指定を支援する機能、AT 結果をビジュアル化する機能などが必要である。
7. 方針決定の支援
   実行結果の解析をもとに、以後における AT 機構の付加方針の決定を支援する。

第 6 章　自動チューニング機能をもつライブラリ

6.1 ライブラリへの自動チューニング機能の組み込みの意義

ライブラリは、もともと汎用目的で利用されてきたものであり、多くの機能を簡便に提供する役割をもつ。現在でもほとんどのライブラリにおいて、ユーザ自身が最適なルーチンを選択し、必要ならば、精度や実行時間に影響するパラメタ値を決定し与えなくてはならない。また、計算規模の拡大にともない、ハードウェアだけでなくライブラリについても、高い性能が求められるようになってきた。コンパイラが扱うことができる性能チューニングの範囲は非常に限定的で、局所的な速度向上にしか対応し切れていない。ライブラリの場合、ソースプログラム（ソースコード）からライブラリを構築する段階では与えられる問題のサイズや性質は未知であるため、問題に特化した高度なチューニングは行うことができない。また、ソースプログラムの形で提供されるライブラリの場合、ユーザがライブラリを実行する計算機環境の情報は利用しようがない。高い性能をもつライブラリを実現するためには、チューニングの仕組みを自動化する必要がある。

6.2 数値計算ライブラリの自動チューニング機能

現存する数値計算ライブラリの中には、自動チューニング（AT）機構をもつものがある。既存の AT 機構をもつライブラリ（AT ライブラリ）は、限定された解法の高速化だけを対象としており、数値計算応用の分野における広範囲な問題を解決できているとはいい難い。一般に数値計算応用の分野では、**基本線形代数副プログラム** BLAS などの基本演算をもとにした階層での処理を、基本構成要素として包含する。しかし、数値計算応用における高性能化、高安定性の要請を考えると、対象を基本演算階層に限定しないで、より高い演算階層での高性能化や高安定性を達成できる方式が必要となる。このような数値計算応用ソフトウェアの階層性を考慮するとき、従来の AT 技術は、適用されているアルゴリズム階層が低水準に制限されている。

数値計算ライブラリに対する AT 機能は、図 6.1 で示す 2 つの機構、チューニング対象パラメタの最適値を決定する機構と最適アルゴリズムを選択する機構で実現される。数値計算ライブラリに関する AT 機構の入力データとしては、性能ポリシ（最大実行時間、使用できる最大記憶量、計算結果の要求精度、性能安定性など）と対象とする計算機環境の特性情報（記憶容量、キャッシュサイズ、プロセッサ数、通信性能など）がある。**パラメタ値決定機構**では、アルゴリズム毎に実行時間がもっとも短くなるようなループ展開段数や、キャッシュ最適化やレジスタ最適化のためのブロックサイズなどのパラメタ値を決定する。

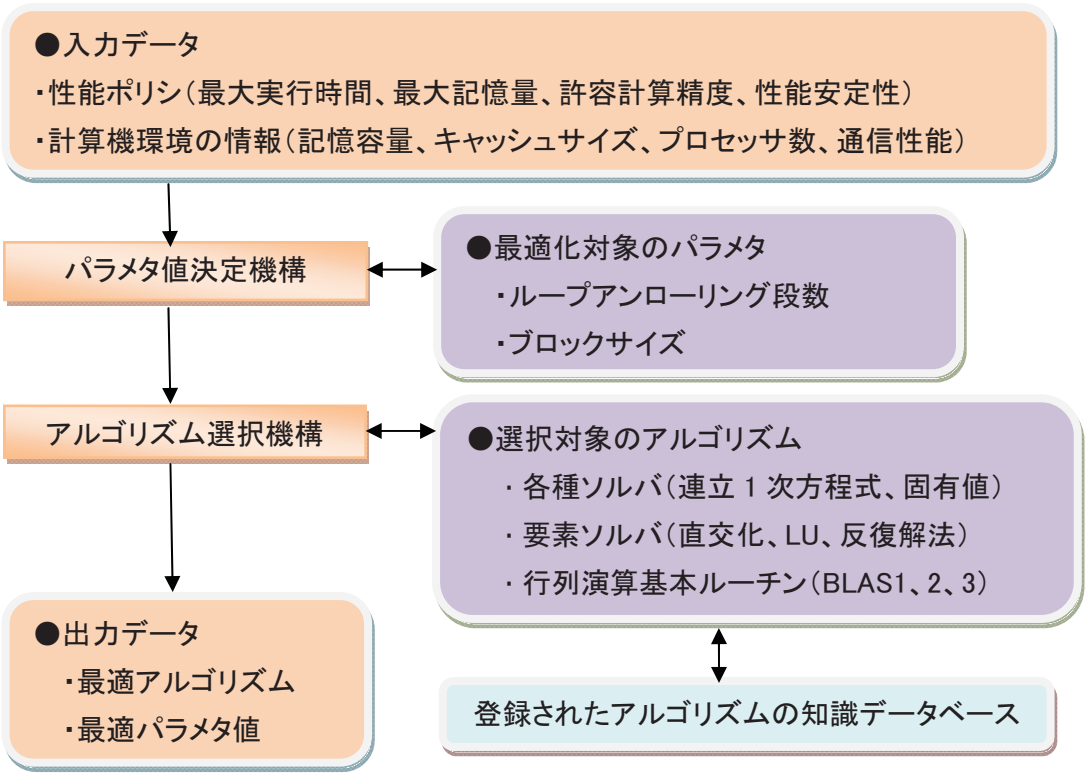

図 6.1 数値計算ライブラリの自動チューニング機構

アルゴリズム選択機構は、次の 3 階層に分解できる。

1. 数値計算応用における各種ソルバ
2. 各ソルバを構成する要素ソルバ
3. 行列演算基本ルーチン

上記の階層ごとに選択すべきアルゴリズム要素を定義し、その性能を決定する調整パラメタと有効範囲を指定する。アルゴリズム選択機構では、プログラムが指定した性能ポリシと、自動チューニングシステムが自動的に取得した計算機環境の情報が入力され、各階層における最適なアルゴリズム要素が選択される。アルゴリズム選択機構においては、知識データベースを利用することでより効率化を行うことが可能である。例えば、ライブラリのインストール時と実行時に、実行時間、キャッシュミス率、性能安定性などのデータを測定し、知識データベースに格納しておく。それを用いて、チューニングの効果を定量的に予測し、予測結果にもとづいてチューニングの判定を行うとことが可能である。ただし、判定基準については、速度と精度、速度と記憶量、速度と記憶量と精度などのトレードオフが存在する。対象が並列計算機の場合、並列

処理オーバヘッドに起因するプロセッサ数と速度のトレードオフがある。これらのトレードオフを勘案して効果を予測し、最適化パラメタ値およびアルゴリズム要素を決定し、チューニングを実施する必要がある。

6.3 自動チューニング機能をもつライブラリの開発と利用

6.3.1 自動チューニングライブラリの開発

数値計算ライブラリの開発者は、自動チューニング機能についての知識とスキルをもち、それを活用して、自動チューニング機構を数値計算ライブラリに実装する。図 6.2 に、数値計算ライブラリの開発者からみた開発手順を示す。作成された自動チューニング機能をもつ数値計算ライブラリでは、対象とする計算機環境は特定されていない。すなわち、この段階では計算機独立な数値計算ライブラリである。その利用者が対象計算機環境の情報（特性を示すパラメタ値）を与え同計算機にインストールすることで、同計算機に最適化された数値計算ライブラリが実現される。

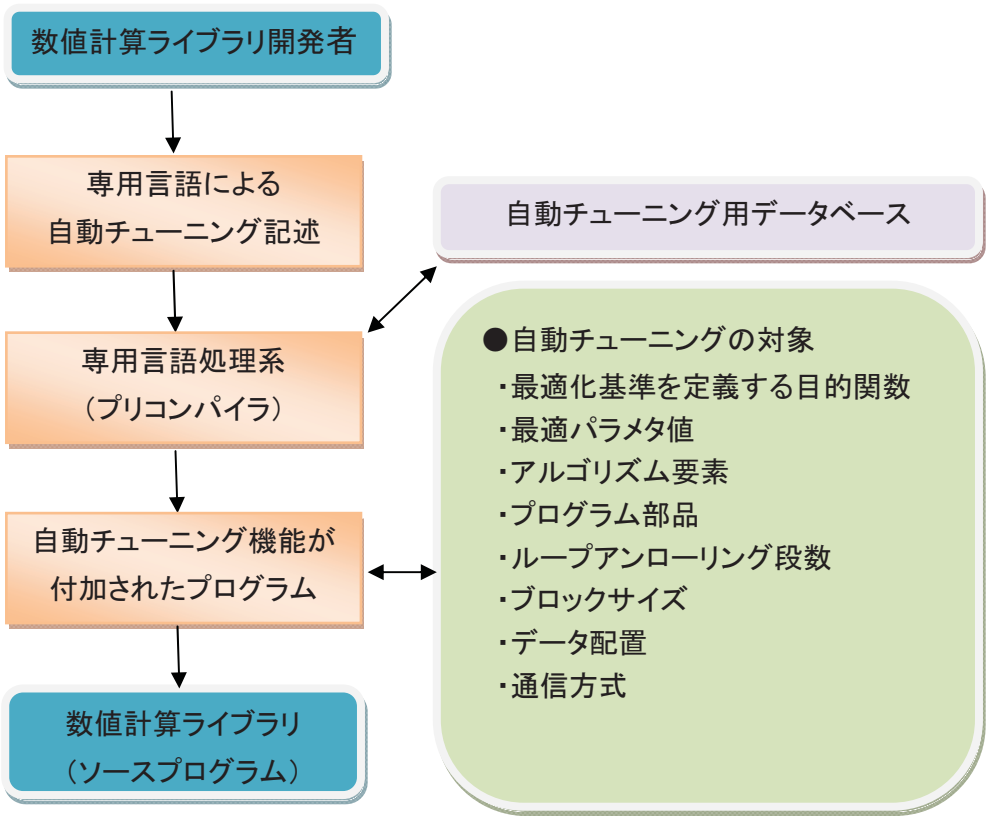

図 6.2 自動チューニング機能をもつ数値計算ライブラリの開発手順

6.3.2 自動チューニングライブラリの利用と実行

数値計算ライブラリ利用者は、数値計算応用プログラムを作成するとき、数値計算ライブラリ開発者が用意した AT 機能をもつ数値計算ライブラリを利用する。ライブラリ利用者は、問題サイズなどの数値計算応用プログラムおよびそれを実行する計算機環境の情報を保持している。これらの情報を性能ポリシとして入力することで、実行時間の最小化などの性能の自動チューニングが行われる。自動チューニングが適用されるタイミングは、対象計算機へのライブラリのインストール時、プログラム実行前の関数コール時、さらに同プログラムの実行時の 3 段階存在する。ライブラリ利用者は最適化が有効なタイミングを指定する。

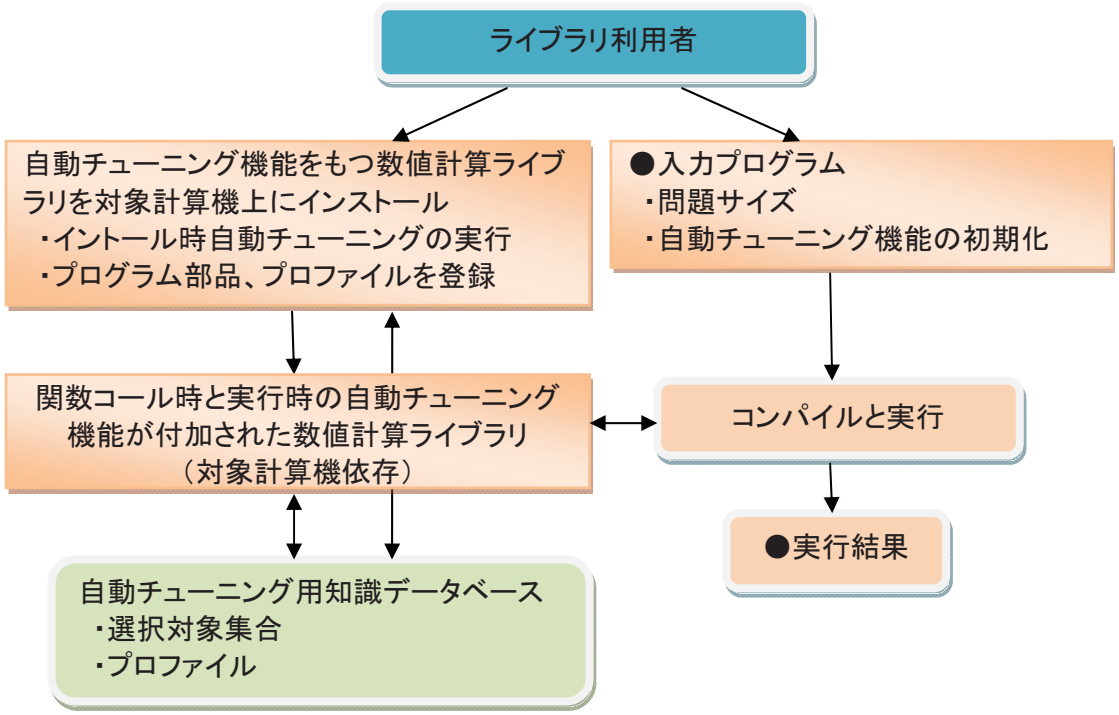

図 6.3 自動チューニング機能付き数値計算ライブラリの利用手順

図 6.3 で示すように、ライブラリ利用者はまず自動チューニング機能をもつ数値計算ライブラリを自分の計算機環境に最適化してインストールする。出力されたライブラリは、性能ポリシによって定まる範囲で対象計算機に最適化されている。このインストール時自動チューニングの過程で、ループアンローリング段数、ブロックサイズ、アルゴリズム要素、プログラム部品などが適切に選択され、それらを含むソースプログラムが生成される。同ソースプログラムは、その後の関数コール時、実行時自動チューニングへの入力となる。変換過程では、自動チューニングシステムが管理する知識データ

ベースに置かれる種々の選択候補、プロファイリングされた性能データなどが利用される。

関数コール時自動チューニングでは、行列サイズ、ループ回数など問題の特性を示すプログラム変数が、利用者によって定数として与えられたあとのプログラムに対してチューニングを行う。例えば、ライブラリ利用者が与えたプログラムの問題サイズにもとづき、最適化パラメタ値の候補の中から適切な選択が行われる。この過程で、必要に応じて知識データベースに置かれたアルゴリズム要素、プログラム部品（候補）、プロファイルなどが参照される。その後、コンパイルが行われ、数値計算ライブラリと結合後、目的プログラムとして実行される。実行時自動チューニングでは、入力されたデータの内容によって決まる特性（例えば、密行列、疎行列、帯行列など）、対象計算機および通信網の動的負荷状態などの実行時情報を考慮して最適化を行う。これら 2 つの自動チューニングについては、性能モニタリング情報を含む知識データベースが適宜利用される。

6.4 数値計算ライブラリへの組み込みの問題点

6.4.1 自動チューニング実行時間の削減

自動チューニング機能をもつライブラリには、自動チューニングの実行に要する時間をいかに減らすかという問題がある。まず、パラメタ値の決定がインストール時に行えるもの、すなわち、計算機の特性情報（命令セット、レジスタ数、キャッシュサイズ、主記憶容量など）のみを利用するものであれば、インストール時にできるだけ時間をかけて性能パラメタ値を決めることで、高い性能をもつライブラリを構成できる。あるいは、パラメタ推定範囲などの決定を行い、実行時の自動チューニングに要する時間を削減することも可能となる。しかし、パラメタ値やアルゴリズムの組み合わせが膨大な数になり、測定データに大きなばらつきが生じることもよくある。この場合には、インストール時であっても膨大な時間をかけることはできないので、第 4 章で述べた AT 数理基盤を利用して、最適なパラメタ値を決定することが必要になる。

6.4.2 実行時自動チューニングにおける選択

実行時にしか行えないパラメタ値の決定やアルゴリズム選択をどのように行うかという問題もある。密行列同士の積や FFT では、入力データの数値は実行時間に影響を及ぼさない。しかし、連立 1 次方程式の反復解法のように、入力データの数値が変わると大きく実行時間に影響するものも多い。この場合、実行時自動チューニングにかかる時間はできるだけ小さい方が望ましい。そのためには、選択するアルゴリズムの候補をユーザ毎にあらかじめ絞っておくなどの工夫が必要になる。例えば、利用する

アルゴリズムとして選択されれば高速化につながるものでも、適用可能かどうかの判定に時間がかかるものであれば、選択候補から外しておくこともあり得る。また、判定にかかる時間が短いものであっても、適用される可能性が極端に低いものについても候補から外しておくことも考えられる。これらは、ユーザや計算機環境によっても大きく左右されるので、知識データベースの利用は不可欠となる。

6.4.3 開発工数の削減

開発者側の視点にたつと、AT ライブラリの開発は大変な作業量を必要とする。実際、プログラムのソースコードの行数が、数万行に及ぶことも珍しくない。これは、多種類の計算機に対応させるため、あるいは、今後出てくる新しい計算機に合わせるために、多くの実装方式をあらかじめソースコードのレベルで用意しておくからである。その結果、ソースコードを自動生成する仕組みの導入が必要になる。もとのソースコードから簡単な指示文を挿入するだけで、いろいろな実装方式のプログラムソースが生成できることが望ましい。

6.4.4 自動チューニング効果の評価

ユーザの視点からは、自動チューニングを行うことでどれほどの性能向上となったのかが気になる。性能向上比は自動チューニングを行わない場合をどう設定するかで大きく変わってくるため、比較に用いる設定値をあらかじめ決めておくことが必要である。例えば、逐次プログラムであれば、Netlib で公開されているベンチマークプログラムを利用する方法が考えられる。並列プログラムの場合は、現状では適切な評価の比較基準がない。

6.5 数値計算ライブラリに関する課題提案

6.5.1 マルチコアに対応した高性能基本線形代数副プログラム

基本線形代数副プログラム（BLAS）に含まれるルーチンは、多くの数値計算ライブラリから呼び出され、アプリケーションによっては実行時間の大半を占める。基本ルーチンなので、このルーチン内においては記憶バンク競合を回避するようなデータ配置の決定やアルゴリズムに関する選択余地はほとんどない。BLAS1 であれば、最適なループアンローリング段数の決定が重要である。また、BLAS2、BLAS3 では、キャッシュヒット率をあげるために、ブロック化アルゴリズムの採用とブロック幅の適正な選択が重要となる。現在、多くの計算機がマルチコア化されていて、1 チップ上に複数のプロセッサが搭載されている。そこで、プロセッサ毎に処理をどう分割するかが重要である。また、計算機毎に記憶バンド幅、キャッシュサイズ、プロセッサ数、プロセッサ性能などは異なっている。このような多様な計算機環境においても、高い性能が出るよう

な BLAS を開発する必要がある。BLAS は入力データの特性にほとんど依存しないことから、インストール時の自動チューニングに適している。

6.5.2 問題に応じて適切なソルバを自動選択する固有値ソルバ

固有値ソルバでは、対象となる行列の種類に応じて、それに適したソルバを選択しなくてはならない。要素ソルバとして、次のような解法が考えられる。

1. 直交化ルーチン： Gram-Schmidt 法、QR 法、Givens 法
2. 相似変換ルーチン： Householder 3 重対角法、Householder 帯行列化法
3. 3 重対角行列求解ルーチン： 2 分法＋逆反復法、Givens 法、QR 法、MRRR 法、分割統治法、Lanczos 法
4. 疎行列解法ルーチン： Arnordi 法

それぞれの要素ソルバに対して、ループアンローリング段数やブロック幅などのパラメタ値の決定機構が必要となる。また、正しい要素ソルバを選択するためのアルゴリズム選択機構も必要である。さらに、クラスタシステムなどの並列計算機環境においては、データ分割方式や通信方式に関するアルゴリズム選択機構が重要となる。

6.5.3 許容残差に応じて解法を自動選択する密行列用連立 1 次方程式ソルバ

係数行列が疎行列であっても、許容残差が小さい場合には、主に密行列解法で使われる直接解法を選択しなくてはならない場合がある。直接解法はまず大きく 2 つに分けられる。与えられた係数行列 1 つに対して右辺ベクトルも 1 つだけの連立 1 次方程式を解く場合には、ガウス消去法が利用できる。同じ係数行列に対して右辺ベクトルが複数存在する場合には、係数行列を 1 度だけ LU 分解する方法が利用できる。LU 分解の要素ソルバとして、さらに次の 3 つの方法がある。

1. 外積形式ガウス法（right-looking アルゴリズム）
2. 内積形式ガウス法（left-looking アルゴリズム）
3. クラフト法

それぞれの方法について、キャッシュ最適化、通信最適化のためにブロック化が行われ、探索対象の性能パラメタの範囲は大きくなる。また、基本計算要素として BLAS が呼ばれるため、その部分の性能によっても全体の性能が大きく左右される。与えられた行列が帯行列やスカイライン行列（可変帯行列）の場合には、通常の LU 分解をそのまま実行すると 0 の部分まで計算対象になり、効率が悪くなる。そこで、帯行列やスカイライン行列では、それらの行列内だけで LU 分解が行える特徴をいかし、それに特化した LU 分解を作成することで高速化を達成できる。これまでは、主に速度性能だけを追求することが多かった。しかし、許容残差に応じて解法を選択することも考え

られる。精度の要求が厳しい場合には、完全ピボッティングを行うことが最適である。しかし、実際の多くの問題では、列方向または行方向だけにピボッティングを行う部分ピボッティングで十分であることも多い。また、あらかじめ与えられた行列が対角優位であるとわかっている場合、あるいは、精度がそれほど重要ではない場合には、最初からピボッティング処理を省くことで高速化を達成できる。行列サイズが大きい場合には、ピボッティングの時間は相対的に小さくなるため余り問題ではない。しかし、小規模な問題でかつ何度も実行するような場合には、ピボッティング処理を省略する効果は大きい。

6.5.4　幅広い問題に対応した疎行列用連立 1 次方程式ソルバ

疎行列用連立 1 次方程式ソルバでは、多くの解法が存在する。直接解法としては、通常、対称正定値行列に限定されるが、スパースソルバなどにも利用できる。反復解法であれば定常的反復法と非定常的反復法とに分けられ、定常的反復法としてはヤコビ法、ガウス・ザイデル法、SOR 法、SSOR 法などがあり、非定常的反復法としては CG 法、GCR 法、GMRES 法、MINRES 法、Bi-CG 法、CGS 法、QMR 法、Bi-CGSTAB 法、SYMMLQ 法、GMERR 法など、数多くの解法がある。反復解法は、各反復では疎行列ベクトル積が行われ、実行時間の中でも大きな割合を占める。この実行時間には、データの分散方式に大きく影響される。疎行列のデータ分散方式は 1 次元行方向ブロック分割と 1 次元列方向ブロック分割の 2 通り分けられるが、選択する解法や並列実行環境によってどちらが最適であるかが変わってくる。また、各ノード内における疎行列の格納形式にも記憶量を削減するための圧縮形式やアンローリング向きの格納形式などがあり、どの格納形式を用いればよいかはプロセッサの特性、キャッシュサイズの大きさなどに大きく依存する。また、GMRES(m)法のように途中でリスタートを行う反復解法では、リスタート周期に応じて使用する記憶量も変わる。リスタート周期の決定は、問題の特性、プロセッサの特性などによって変わるため最適なリスタート周期を決定するための AT 機構も必要である。

反復解法では、前処理を適用することが一般的である。しかし、前処理は問題によってその効果が大きく異なる。この場合、知識データベースの利用は必要不可欠である。例えば、ある特定の分野の問題にはとても効果の高い前処理があっても、あるユーザがそういった問題を解かない場合にはその前処理の適用処理を外すといった工夫が可能になる。前処理には、多くのオーバヘッドがあるためである。このように前処理の適用には、その前処理の効果を調べるのに要する時間、実際にその前処理が適用される確率、実際にその前処理を適用した場合の速度向上など、多くの要素がトレードオフの関係になる。そのため、適切な前処理を決定するようなアルゴリズム選択機構が必要になる。精度に関しては、単精度実数と倍精度実数を組み合せた混合精

度の計算を行う方法も考えられる。単精度実数の計算は倍精度実数の半分の記憶量ですむので、同じキャッシュに倍精度実数の 2 倍の数の要素がはいる。また、単精度しか実行できない SIMD 命令を有効に利用できる。並列計算機環境では、通信が必要な計算に対して単精度実数を用いると、ノード間の通信量は倍精度実数に比べて半分ですむ利点がある。しかし、単精度実数と倍精度実数との間の型変換のオーバヘッドが大きく混合計算に不向きなプロセッサの場合には、逆効果となる場合もある。要求される精度を満たすために、反復の途中で倍精度に切り換えるなどの工夫も必要になってくる。どの時点で切り換えると効果的であるかが、自動チューニングの対象になる。

6.5.5 入力データに適した高速フーリエ変換法を自動選択する知識データベース

高速フーリエ変換(FFT)には、要素数を 2^n としたとき、通常の FFT アルゴリズムの他に記憶量が半分で済む Cooley-Tukey アルゴリズムがよく知られている。この Cooley-Tukey アルゴリズムでは、基底(1 度に処理する要素数)を増やすことで、記憶へのアクセス回数や計算量を減らし高速化することが行われている。しかし、プロセッサに搭載されている浮動小数点レジスタの個数の制約などにより、必ずしも基底が大きければよい訳ではない。また、この基底は組み合せて使うことができるが、組み合せによっても性能が大きく変わる。例えば、2^6=64 要素のときに 8 基底を 2 回使うのと、16 基底と 4 基底を 1 回ずつ使う方法があるが、どちらが早くなるかは実装方法やプロセッサの特性に依存する。さらに、Cooley-Tukey アルゴリズムには、計算量を削減した Split-Radix アルゴリズムがある。計算量を削減しても必ずしも高速になるとは限らず、これについてもプロセッサの特性により、どちらが高速になるかは異なる。その他にも、Stockham アルゴリズムなどがある。それぞれのアルゴリズムにおいて、周波数間引き型 FFT、時間間引き型 FFT という 2 種類の実装方法がある。これらの方法は最内ループの式の形が異なるため、プロセッサによっては一方が他方より性能が出やすいことがある。

通常、FFT アルゴリズムは、処理するデータがキャッシュに載っている場合に高い性能を示す。しかし、キャッシュサイズよりデータが大きくなると、著しい性能の低下をきたす。キャッシュを有効に利用するために、本来 1 次元の要素を 2 次元データとみなして計算することで、キャッシュミスを少なくする方法がある。また、並列化を行う際には、3 次元 FFT とみなして計算する方法もある。このように FFT は実装方法が多種多様である。さらに、通常の要素数は、2 のべき乗に限定されているわけではなく、そういった場合は混合基底 FFT が使われる。このような場合には、アルゴリズムの選択範囲はさらに広がる。高速フーリエ変換の特徴として、入力データの値がどのようなものでも、通常、性能には影響しない。そのため、同じ計算機上であれば、ある要素でもっと

も早い実装方法を見つければ、次から同じ要素がきたときに同じ方法が使える。そのため、知識データベースとして構築しておくことが可能である。

6.5.6 倍精度と 4 倍精度、4 倍精度同士の演算を行う演算ライブラリ

計算機内部の実数の数値表現では、有効桁数に限りがあるため丸め誤差が起きることはよく知られている。現在、計算機が進歩し、計算規模も格段に大きくなった。そのため、一般に使われている倍精度実数の数値表現では計算精度がたりず、正しい結果が得られないという問題が発生している。この場合、4 倍精度実数のような高い精度の数を用いた演算が必要になる。現在のところ、このような高精度演算をハードウェアでサポートしているプロセッサはないので、すべての演算を 4 倍精度で行うことは得策ではない。そこで、計算精度を要求する部分だけに高精度演算を適用する方法などが考えられる。4 倍精度演算は多くの倍精度演算を使って実現するが、浮動小数点演算遅延の影響で、1 回の 4 倍精度演算だけを行おうとすると多くの時間を費やす。そこで、ある程度演算をまとめて行うことで高速化が可能となる。このとき、どういった演算を、どれだけまとめて、どういう順番で行うかを調べる必要がある。これらは、プロセッサの浮動小数点レジスタの数や SIMD 命令の仕組みなどにも大きく左右される。そこで、計算パタンに応じて、最適な実装方式が自動選択されるという高性能なライブラリの開発が重要になってくる。これらの開発には、自動チューニング機構が活用できる。

第 7 章　自動チューニングが対象とするシステム

7.1　対象システムの範囲

自動チューニング（Automatic Tuning; AT）の適用対象となるプロセッサ、計算機、システム環境全体について、対象システムの技術動向ならびに対象システムにおける AT の可能性を考える。AT 適用の対象システムをプロセッサ単体から、並列計算機、分散システム、ネットワーク環境に拡大し一般的な議論を行うことで、AT 技術の潜在的有効性を考察する。応用については、科学技術計算に限定しない。幅広いアプリケーションが利用される多様な計算機環境を、AT の対象システムと考える。対象システムの技術動向については、マルチコアプロセッサ、クラスタシステムに重点をおく。Grid コンピューティング、Web システム、ユビキタスネットワーキング、センサネットワークなどネットワークを基幹とするシステムは、重要な AT の適用対象である。また、組み込み系システムや FPGA（Field Programmable Gate Array）なども、新たな AT 対象システムとして期待される。AT 技術の今後の発展を考えるとき、これらのネットワーク系、ハードウェアに近い対象システムは、多くの研究課題を提供することが予想される。

7.2　対象システムについて検討すべき項目

各対象システムに共通する検討事項として、それぞれの技術動向、AT技術適用に際しての問題点について考察する。自動チューニングの対象システムがもつ最適化パラメタとなり得る項目をあげ、AT の適用範囲、有効性、可能性の根拠を示す。対象システムが AT 向きかどうかを判定するとき、ATにより得られる効用とAT機能を付加するためのコストのバランスを考えなくてはならない。また、自動チューニングの目的は、プログラムの実行速度、所要記憶量、計算精度、消費電力量などの＜性能＞を向上させることであるが、これらの基準はユーザのポリシおよび対象システムによって異なる。対象システムごとに、意義がありかつ測定可能な性能基準を設定する必要がある。

7.3　対象システムの技術動向と自動チューニングの適用

7.3.1 汎用マルチコアプロセッサ

■技術動向

汎用マイクロプロセッサ市場の動向として、クロック至上主義から低消費電力指向に移りつつある。それにともない、プロセッサの多コア化やオンダイ複合コア化が促進されている。また、100 コアを超える多コア化を進める一方で、GPU（Graphic Processing Unit）を統合したオンダイ混載アーキテクチャが提案されている。その背景として、低

消費電力化がクロック向上の必須であったが、微細プロセス化の過程で直面したリーク電流の存在や、量子効果による既存半導体技術で高クロックパッケージの実現性が問題となり、低クロックで高性能化が可能なマルチコアパッケージ化が進んだことが指摘できる。以下では、代表的ベンダごとに、マルチコアプロセッサの技術動向を展望する。

Intel: 2005 年以降、マルチコア版 IA32 は Pentium4 の後継 Pentium-D において初めて市場に登場した。各コアは単一記憶バスを共有し、バスとキャッシュコヒーレンス制御機構の存在を除けば、各コアはほぼ完全に独立している。ついで低消費電力の Core Duo マルチコアプロセッサが出現した。Core に続き Core2 が後継として現れたが、マルチコアの視点から、それ以前のものと大きな変化があった。各コアが独立したローカルなキャッシュを保持していたのに対し、2 コアで共有可能な共有キャッシュが搭載された。4 コア以上のマルチコア構成の場合は、2 コアを 1 組とし多コア化がなされており、コアによりキャッシュ制御が異なる。また、CPU と GPU のダイ混載アーキテクチャ Larrabee のリリースを表明している。これは Pentium 相当の単純コアを高クロック化し多数搭載することで、消費電力あたりの性能の問題を解決することを意図している。さらに、マルチコアプロセッサの将来として数 100 コアを集積した多コアアーキテクチャによるテラスケールコンピューティング戦略を発表している。同戦略では、100 個程度のコアがオンダイインターコネクトにより相互結合する。コアは均一ではなく、シングルコア特化、マルチコア特化、また特定用途コアなどを同一ダイ上に搭載し、オンダイモジュールの周波数制御により低消費電力化する。超並列計算機を、そのまま同一シリコン上に実装したものといえる。

Advanced Micro Devices (AMD): AMD は、オンダイコア間の対称通信可能なマルチコア開発を初期段階から実施していた。AMD の K8 世代以降のマルチプロセッサは、各プロセッサがクロスバー結合をしており、完全結合を目指した設計となっている。AMD 初のマルチコアプロセッサは, Athlon64 をデュアルコア化した Athlon64x2 であった。記憶コントローラを共有し、コア間相互通信をチップ内で実現できる。記憶コントローラをチップ上に搭載し、キャッシュコヒーレンス制御に由来する記憶バンド帯域のロスは少ない。2007 年、4コアプロセッサである PhenomX4 をリリースした。4コアが記憶コントローラならびにレベル 3 キャッシュを共有する対称構造をとっている。マルチコア化の別戦略として、CPU と GPU の統合型プロセッサ FUSION アーキテクチャ開発がある。計算機筐体内に存在する高性能演算器をチップ上に集積し、半導体の効率的利用、高性能計算、高性能グラフィックス処理を同時に実現しようとする。

International Business Machines (IBM): 2001 年に出現した POWER4、それを拡張し

た PowerPC970 が初期のデュアルプロセッサである。POWER4 はレベル 2 キャッシュ以下をコア間で共有するアーキテクチャを採用している。サーバ用途のプロセッサとして開発されたため、大容量のキャッシュと広帯域のチップ間 I/O が備わっている。POWER5 は POWER4 を継承しつつ、高クロック化、SMT（Simultaneous Multithreading）機能を実現したものである。POWER6 は、IBM が 1998 年に POWER3、2001 年に POWER4、2004 年に POWER5 と 3 年毎に大幅な機能拡張と性能強化を実施してきたマルチコアアーキテクチャ・シリーズの 2007 年継承版である。次世代の POWER7 も 2010 年に出荷予定である。また、POWER7 を利用した DARPA の PERCS（Productive, Easy-to-use, Reliable Computing Systems）計画で、10PFLOPS 達成を表明している。同システムは、日本の次世代スーパーコンピュータ（京速コンピュータ）計画の有力な対抗馬である。IBM を高性能コンピューティング分野で世界一に押し上げたマルチコアプロセッサは、現在最高速のスーパーコンピュータ Blue Gene/L に搭載されている PowerPC440 である。組み込み向けに開発されたプロセッサシリーズであり、従来スーパーコンピュータに使用されたマイクロプロセッサとは毛色が異なる。Blue Gene/L は PowerPC440 ベースのコアを 2 つ搭載し、4MB の L3 キャッシュを共有する。低消費電力化と高集積化を狙い、組み込み向けプロセッサを利用するなど、高性能コンピューティングシステム向け CPU の新たな戦略を示した。

Sun Microsystems: 現在 UltraSparc T シリーズで、多コアプロセッサを展開している。初めてのデュアルコアは、UltraSparc IV シリーズとして発表された CMT（Chip Multithreading）機構が対応する。UltraSparc T1 は最大 8 コアが搭載されている。各コアは 4 スレッドを実行可能である。浮動小数点演算器を1つに省略するなど、トランザクションサーバに特化したプロセッサである。引き続きリリースされた UltraSparc T2 は各コア 8 スレッド、計 64 スレッド実行可能である。10Gb イーサチップをプロセッサに搭載し、各コア個別の浮動小数点機能も備わっている。

■AT技術の適用

各社のマルチコア戦略により、プロセッサの種類は爆発的に増加することは間違いない。性能に大きく関わるコア数、キャッシュサイズ、クロック、バス幅などの違いは、少なく見積っても、各要素の（想定しうる）種類の総積に達する。これらをプログラムレベルでいかに吸収するかがATの一番の仕事である。それらをハードウェア的にカバーする柔軟な制御ができれば有効だが、商業的に魅力が少ないので大きな期待はできない。マルチコア化により、プロセッサのキャッシュ容量が増えキャッシュを効率的に使う戦略は非常にやりやすくなるとの見方もできる。しかし、従来単一プロセッサで効率的に活用してきたプロセッサ内のハードウェア資源の競合（とくに主記憶や共有キャッシュのアクセス競合）が発生するため、単純にコア数増加分の恩恵を受けること

は困難である。ラベリング情報を使いキャッシュラインごとの制御を行うなどの機能追加も予定されているが、そのようなキャッシュと過去のキャッシュの整合性をとるのも、最適化を複雑にする大きな要因になる。さらに大きな問題は、プロセッサの複雑化にともない、コア数増加によるプロセッサ性能の線型上昇を阻害する要因の特定が困難となることである。資源競合を適切に制御または最適化するうえで、以下の項目が重要である。

1. 共有キャッシュ方式と記憶階層構造
2. 主記憶ーキャッシュ間（またはキャッシュ相互間）のバス構造と性能
3. 各コア間通信の均質性・非均質性

複数種類のプロセッサのもとでのソフトウェア開発では、プロセッサの多様性をいかに統一的に取り扱うかが重要である。また、共有メモリ型並列プログラミング、スレッドプログラミングを意識したスタイルへの変更を余儀なくされ、現時点ではソフトウェア開発効率において、多くの問題を抱えている。

プロセッサ数が爆発的に増大する対象システムで性能の最適化を実現するためには、ハードウェアやコンパイラのみに頼るのではなく、AT技術の併用を考えなくてはならない。他章とも重複するが、適用すべき AT 技術の例を以下に示す。

1. プロセッサのシステムパラメタを参照し、ABCLibScript などの AT 支援ツールを用いて、ループ最適化を行う。または最適なコードを、インストール時あるいは実行前に選択する。
   - キャッシュサイズ： タイリングサイズ
   - レジスタ数： レジスタブロッキング、ループアンローリング段数
   - キャッシュ共有形態： プロセスマッピングやスレッドオーダリング
   - 記憶帯域： スレッドコアマッピング、使用コア数制御
2. アプリケーションの開発効率を、チューニングの視点から支援する。例えば、抽象的なアルゴリズム表現から、実行時の最適アルゴリズムまたは実装関数を選択する。また、開発時に使用環境の付帯情報から適切なコードを提示する。
3. 過去の実行履歴データベースから、プロセッサ依存の最適化問題をオンラインで動的に解決する。

また、開発環境とハードウェアの相性に関する情報も、ATの支援により性能改善に大きく貢献する。例えば、コンパイラバージョンによって最適パラメタ値が異なる場合が多いが、そうしたプリミティブなチューニングの差異を AT により支援する。

1. 異なる最適化やコード生成の癖に対応した推奨パラメタの提示

- ループアンローリング段数
- 演算順序変更
- コンパイラ指示子の有無・位置など

2. 実行時のランタイムライブラリの癖に応じた推奨パラメタの提示
   - OpenMP の適切な並列化
   - OpenMP の Parallel Region の位置など
   - OpenMP のスレッドスケジューリング
   - 数値計算ライブラリの動的選択など

多コア化と複雑化がさらに進めば、最大の利用効率を得るために、フルコア・フルキャッシュを利用することなくプロセッサの 1 部分のみを計算に利用する戦略も有用である。各社のオンダイ混載アーキテクチャも、それが 1 つの狙いである。そうなると、ユーザが利用するプロセッサ資源は、ある意味で変化するオーダメードプロセッサであるといえる。そのように、多様化が爆発的にかつ時々刻々変化するような状況に対して、戦略的に最適なコア構成を選択する方法論として、動的な実行時自動チューニングが有効である。

1. 実行時の動的スレッドコアマッピング
2. 実行時の数値計算ライブラリへの適切なスレッド（コア）数指定など

さらに、ここでいう利用効率には実行時間だけではなく、消費電力量も対象に含めることができる。例えば、AMDの Phenom プロセッサのように全コアの周波数を制御できる場合には、適切なコアマッピングとともに、周波数制御により実行時間と消費電力量の最小化を同時に行うなどの多目的チューニングが可能となる。

7.3.2 専用マルチコアプロセッサ

■技術動向

本節でとりあげる Cell Broadband Engine（Cell/BE）は、Sony が販売するビデオコンソール PlayStation3 （PS3） 向けのマルチメディア高性能マルチコアプロセッサである。Cell/BE は高精細ディスプレイやリアルタイムレンダリングなどの現実感のある画像を追求するゲーム業界の要請や、マルチメディア家電のメディアセンタ的な役割を果たすため、高スループットのストリーミング処理や近代的 CG に必要な物理シミュレーションが可能な高性能プロセッサとして開発された。PS3 の前機種 PS2 のプロセッサ SE (Synergetic Engine)と同様に、オンダイアクセラレータの形態をとる。すなわち、外部からは通常にシングルコアアーキテクチャだが、内部的には DMA（Direct Memory Access）通信により記憶コントローラを共有するマルチコアアーキテクチャである。ゲームコンソール向けに開発された Cell/BE であるが、その高いコストパフォーマンスか

ら科学技術計算への応用が考えられてきた。とくに、アメリカでは IBM や各研究機関を中心として Cell/BE を倍精度計算用に拡張したアクセラレータボードを Opteron クラスタに搭載した高性能計算機 Roadrunner により、世界で初めて**ペタフロップス越え**を達成している。

■AT 技術の適用

Cell Broadband Engine（Cell/BE）は汎用計算機としての可能性を期待されるプロセッサであり、高性能計算機としての潜在的な可能性を検証する必要がある。Cell/BE の利用には Cell/BE が搭載する SPE（Synergistic Processor Element）を効率的に利用した並列プログラムの作成が必要である。それには IBM が提供する Cell/BE のソフトウェア開発キット（SDK）を利用し、SPE プログラムオブジェクトと PPE（PowerPC Processor Element）、SPE 間を制御するスレッドプログラムを作成しなければならない。APIは、比較的低レベルなものが提供されている。また、SPE は主記憶にはアクセスできないため、DMA 転送により適切なデータ転送を厳しく制限された帯域の中で実施することが求められる。結果として、各 SPE の局所記憶上にロードされたデータの再利用や、ダブルバッファリングなどのプログラミングをユーザに課すことになる。これらを抽象レベルで実現できるプログラミングモデルの提供など、ソフトウェア開発環境面で克服すべき課題は多い。

Streaming SIMD Extensions（SSE）は Element Interconnect Bus（EIB）資源を共有する点を除いて、ほぼ独立したコアと考えてよい。従って、シングルコアレベルでは、汎用マルチコアプロセッサの項で示した適用項目が該当する。SSE 特有の問題としては、スクラッチパッド的な局所記憶の効率的利用、および、AltiVec 命令を効率に使うためのアンローリングや式変形が、ATの適用対象となる。また、専用マルチコアプロセッサの利用では、以下の AT 適用対象が考えられる。未知な部分が多いため、AT の介在範囲はこの他にも多いものと予想される。

1. EIB の帯域を適切に共有できるスケジューリング
2. PPE と複数 SSE の効率的な非同期管理
3. 動的負荷バランス支援
4. 既存の複数バージョン存在する Cell 間の差異吸収
5. 複数Cellプロセッサ間の協調
6. 階層的な記憶構造の有効利用（データブロッキング）
7. 本体CPUとの並列動作時の動的負荷バランス支援

7.3.3　グラフィック処理専用プロセッサ（GPU）

■技術動向

GPU（Graphical Processor Unit)では、 多くのトランジスタを投入して、パイプラインの多重化、高クロック化を進めるとともに、レンダリングパイプラインの高度化がなされている。まだ CPU ほど熱対策が厳しくないといわれており、多コア化やオンダイ複合コア化が Cell 以上に進行している。ビデオカードは比較的古い時代からプログラマブルあった。ただし、プログラマブルといっても簡単にビデオカードをアクセスできた訳ではない。かつては、OpenGL や DirectX などのグラフィクス API から、間接的・模擬的にビデオカードに計算させることができたのみであった。また、より高度なコンピュータグラフィックス（CG）を必要とする場合には、ビデオカードの機械語を直接扱う方法があったが、ビデオカード開発の関係者でなければ知り得ない情報が多いなどの問題が存在した。

2002 年、Nvidia は直接ビデオカード内に Cg 言語（C 言語の拡張版）プログラムをアップロードし、独自のシェーディングプログラムを使用できるような環境を整備した。Cg 言語の出現により、GPUによる汎用計算（General Purpose GPU; GPGPU）が可能になった。ただし、ビデオカード独特のパイプラインアーキテクチャを理解したうえでプログラムを作成する必要があり、GPU を汎用機械として利用するにはまだ敷居が高かった。Nvidia が 2006 年に公開した CUDA 言語は、プログラマブル GPU のアーキテクチャを抽象化し、そのモデル上で汎用プログラムを効率的に開発可能にする統合開発環境である。CUDA は G8 世代以降の比較的最近のアーキテクチャでのみ動作する。CUDA によってハードウェアレベルでの抽象化がなされたため、プログラマはパイプラインの構造などを知る必要がなく、CUDA 特有のスレッド制御や記憶階層などに注意をすれば、簡単に並列プログラムを GPU で走らせられる。GPU の標準言語仕様となりつつある。

■AT 技術の適用

CG では倍精度計算の需要が少なく、GPUに搭載される浮動小数点演算器は基本的に単精度演算器である。そのため、条件数の大きな計算には向かないか、精度上のケアをしたプログラム作りが要求される。近年、倍精度演算器が搭載されて計算精度の問題もなくなり、さらにコア増強などで単体性能（単精度演算で）テラフロップスを記録するに至った。安価で高性能な演算が GPGPU で可能になった。また、GPU プログラミングモデルが、技術的に固まっていない問題がある。各ベンダによりプログラミング言語が異なり、共通に使えるGPGPU向け言語はサポート体制にはない。とくにプログラミングモデルならびに言語が未整備であるために、有効な AT 技術の確立が遅れている。ほぼ毎年変更されるGPUアーキテクチャに対応できるアルゴリズムの選

択も、極めて困難な作業といえる。

チューニング経験の蓄積の少なさが大きな問題である。コンパイラによって異なる最適化機能やコード生成の癖に応じて、推奨するパラメタを提示することが求められる。ループアンローリング段数、演算順序変更、ストリーミングデータ再利用促進などの最適化は、GPU アーキテクチャの世代およびコンパイラ（SDK）によって最適パラメタ値が異なる。そういったプリミティブなチューニングの差異を AT によって支援できる。汎用マルチコアプロセッサで示した AT 技術の適用項目は、ほとんどが対応する。また、GPUは汎用マルチコアプロセッサよりも数 10 倍のコア資源をもつ。GPUの複雑なアーキテクチャを考慮すると、適切なコア利用（スケジューリング）、クラスタリング、グルーピング、階層データのブロッキング、データ配置など、ATが介在する余地は大きい。

7.3.4　クラスタシステム（並列計算機・分散システム）

■技術動向

クラスタシステム、並列計算機では、それを構成する各要素プロセッサについてすでに述べた自動チューニングの適用内容があてはまる。ここでは、要素プロセッサ相互間の複合的な自動チューニング要因について議論する。まず、均質クラスタについて考える。**均質クラスタ**とは、システムを構成するCPU、主記憶、ネットワークカードなどすべて同一であり、要素プロセッサ（ノード）の性能が均一なクラスタをいう。多段ネットワークなどの結合トポロジが原因で、ネットワーク距離に非均一性が想定される場合がある。**非均質クラスタ**とは、ノード性能が不均一なクラスタを指す。各ノードを構成するCPU、主記憶、HDDの他、ネットワーク媒体、ネットワーク結合方式によって多種多様なクラスタ構成が考えられる。代表的なネットワーク媒体としては、Classical Ethernet/10M,100M、1GbitEther、10GbitEther、Myrinet、Infiniband などがある。これらを束ねる技術の組み合せも考えられる。ネットワーク結合方式では、FatTree、多段スイッチ、トーラス、クロスバーなどが実システムで利用されている。並列プログラミングのユーザインタフェースであるメッセージパッシングモデルでは、通信ライブラリの実装系ならびに、使用されているデバイスドライバによって性能が変わる。

■AT 技術の適用

通信ライブラリの性能は実装系またはAPIさらには、内部アルゴリズムの選択に依存する。実際に大規模クラスタでの通信ライブラリでは、メッセージサイズに応じた最適な通信手法を選択し Send/Recieve 関数を構成する。集団通信関数は、メッセージサイズ、使用プロセス数、ネットワーク（遅延、バンド幅）に依存して、最良アルゴリズムが異なる。均質クラスタでは、動的な変動項目は少なく、静的なパラメタ推定にもとづ

いた動的なチューニング手法が存在する。実行時負荷やメッセージサイズのばらつきに応じた自動チューニングも考案されている。実装系の作りこみ具合によっては、チューニングにより数倍性能向上する余地がある。また、プロセススケジューリングと各ノードの周波数制御による低消費電力化も提案されている。非均質クラスタでは、ノード間の性能格差を考慮したプロセスマッピングやスケジューリングによる性能向上を実現することができる。

7.3.5 複合分散システム

■技術動向

複合分散システムとは、非均一クラスタの定義に加えて、異なるアーキテクチャ、例えばスカラ、ベクトル、アクセラレータを1つに結合したシステムである。最近の大規模な高性能計算機環境の多くは、このような複合分散システムを構成している。Cell の項であげた Roadrunner や東工大の TSUBAME システム、スカラ計算機とベクトル計算機を統合する次世代スーパーコンピュータ（京速コンピュータ）などがこの例である。GRAPE-DR などの複数アクセラレータボード混載サーバのクラスタなども、複合分散システムに含めることができる。

■AT 技術の適用

複合分散システムの利用には、混成要素からなるため複数のプログラミングモデルを駆使しなくてはならないなど、ハードウェア資源を使い切る高い技術力が要求される。また、TSUBAME システムの Linpack ベンチマーク報告にもあるように、システム性能のアンバランスなどを考慮した負荷分散技術が必要である。プログラムの部分ごとにアーキテクチャの向き不向きを考慮して適切に選別し、並行動作スケジューリングをするなど、高性能最適化全般にAT技術の適用が不可欠である。

7.4 対象システムに関する課題提案

7.4.1 マルチコア対応の性能安定化自動チューニング

マルチコアプロセッサはマイクロプロセッサ技術の流れであり、ソフトウェア開発者はその性能を出し切るためのチューニングは避けることができない。一方、性能を決定する要因、キャッシュサイズ、記憶バンド幅について、1コアあたりの増加率はあまり大きくない。逆に、減少する可能性も予想される。また、複数コアによるバスやキャッシュの共有形態が多種類存在し、それぞれの理由による性能の不安定化が予想される。いいかえるならば、性能不安定性の原因が複数存在し、かつ、それらが複合的に発生する。例えば、単体コアの性能不安定性は、単純なライン競合によるキャッシュスラッシングが原因である。それが共有キャッシュをダーティにし、他コアの記憶ア

クセス性能を悪化させることが考えられる。今後、これらの性能不安定化の要因に対して、その解消が大きな研究課題となる。複数コアによる排他的利用と協調的利用を制御しつつ、性能の最高点を探ることがこれからのATに必要である。今後の研究の中心になると期待される課題と、その内容を示すキーワードを以下にあげる。

1. 複数コアの最適配置問題
   - 目的関数は記憶アクセストラフィック量とキャッシュ間トラフィック量
   - コア数、キャッシュサイズ、バス幅、非対称バスなどを考慮したモデル化
   - それらの実装と性能解析
2. キャッシュスラッシングが与える影響ならびに解消策
   - コア数やコア配置との相関
   - 単一コアのキャッシュスラッシングとの同等性分析
   - マルチコア化による新種スラッシングの可能性探索
3. 非対称バス利用によるパイプラインストールに起因する性能不安定性
   - コア配置と記憶へのデータ配置
   - スクラッチパッド記憶（ユーザ制御可能なキャッシュライン）の有用性

マルチコア出現の要因でもある低消費電力化についても、検討すべき点は多い。数値計算ライブラリの観点からは、高速化＝全コア使用ではない。また、高速化が最大周波数の利用時であるとは限らない。このような場合に、各コアの周波数を制御し適切なコアマッピングとともに、周波数制御により実行時間と消費電力量の最小化を同時に行うことが可能であろう。例えば、BLAS レベルでの電力を考慮した最適コアマッピングや、より上位の API であってもよい。電力消費量を監視・制御するコアを搭載したアプリケーション開発フレームワークのように、単に実行性能のみならずマルチコアの性質を利用した複合的なチューニング手法も今後の課題となる。

7.4.2 自動チューニング技術を用いた大規模複合分散システムの運用管理

■AT 技術を利用した低消費電力化

従来、計算機の高速化と低消費電力化は相反する事象と考えられてきた。近年、Blue Gene/L や Roadrunner に見られる低クロック動作の低消費電力CPUを高密度に集積するアプローチ、または、TSUBAME に見られる低消費電力アクセラレータを搭載するアプローチなど、ハードウェア面からの提案がなされ、システムの低消費電力化に向けた性能改善がなされている。運用上の戦略として、休眠ノードを停止状態にして省電力化を図る方法や、高消費電力ノードの使用を極力おさえ低消費電力ノードを優先利用するなどの運用ポリシ面からの効率管理方式もある。冷却効率に関して、空調や計算ノード配置、ならびに、稼働ノードパタンの流体シミュレーションによる検証なども実施すべき課題である。

アプリケーションの特性に合わせたノードプロセスマッピングや、異機種計算機に対して利用可能なライブラリが登場した時代に議論があったように、関数もしくはプロシージャレベルでのプロセッサマッピングを行い、場合によってプロセスマイグレーションや RPC などによる異なる計算機資源を積極的に切り替えて使用する方式が意味をもつ。それらを採用し、複数ユーザに透過的に、仮想化したレベルで提供することで低消費電力を達成する枠組みを開発し、実際に提供することがスーパーコンピュータ運用管理に求められる。

古典的なキュー管理に加えて、ポリシ優先の新たな管理方式の導入が必要である。ノード内の各コアのクロック制御を行う方法があるが、一般ユーザに委ねるのは危険が大きい。低消費電力指向のAT技術をとりこんだ数値計算ライブラリを、スーパーコンピュータセンタが提供することにより実現する。低消費電力指向 AT に関する課題を以下に示す。

1. 複合ハードウェア利用による低消費電力化の効率測定
   - 対スループット効率、対価格効率など
2. プロセス配置・動的クラスタ構成
   - クラスタ内・クラスタ間の周波数制御
   - ハードウェア利用権設定
3. 冷却に関する流体シミュレーション
   - 導入時の評価
   - 実行時のキューイングシステムと流体シミュレーションの連携
4. 仮想化技術によるハードウェア共有
   - 資源独占・共有における低消費電力化の評価
5. ユーザによるポリシ制御を可能とするキュー管理方式
6. スーパーコンピュータセンタによる省電力ライブラリのサポート
   - 省電力数値計算ライブラリ
   - 省電力メッセージパッシングライブラリ

■京速コンピュータへの自動チューニング技術の適用

マルチコア、超並列、複合分散の技術要素をもつ次世代スーパーコンピュータである京速コンピュータでは、これまでに示したすべての適用項目の和集合として AT 化を捉えられる。AT適用対象は、極めて多岐にわたる。また、余りある計算機資源の独占的利用はありえない想定であり、複数のビッグユーザの同時利用が一般的な利用形態である。そのような環境下で有効に資源管理を行い、資源衝突を少なくし、さらに低消費電力化までも視野に入れつつ、世界最高性能での運用を考えねばならない。

ここでは、特定のユーザプログラムだけではなく、同時に実行される複数のユーザプログラム間で協調制御により、ATが実現される。大筋の研究課題は低消費電力化であげたものと共通するが、ハードウェア資源量が大幅に増加することを考慮する必要がある。

7.4.3 自動チューニングシステムのアーキテクチャ設計と有効性評価

自動チューニングベースのシステムアーキテクチャ設計は重要な課題である。複数のユーザプログラム間での協調動作を可能にするシステムがある。そのような環境下で確実にATを可能にするためには、あらゆる動作時のプロファイリング情報をAT基盤ソフトウェアに提供できるシステムが必要である。先にあげたマルチコア対応の性能安定化自動チューニング、低消費電力化とスーパーコンピュータ管理方式の課題を実現するに最小限必要なモニタリング機能、プロファイリング機能の選択とシステムへの実装が課題である。また、ATを指向したソフトウェアとAPIの関係を整備することなども、今後の課題としてあげられる。

これまでAT技術で曖昧かつ非客観的に扱われてきたのが、性能評価の方法と評価基準である。技術革新により AT の評価項目が変化することは間違いなく、不連続な事象を組み込みつつ時間的に連続して評価可能な指標が利用できることが望まれる。並列計算機の性能評価では、通常ベンチマーク方式が用いられる。もっとも有名な Linpack ベンチマークでは、持続的最高性能が測れるのみである。また、HPCCベンチマークも、マルチコアのクラスタまでの評価に意味があるベンチマークであろう。多様なシステムにおけるATの有効性評価で利用可能なベンチマークが必要である。性能のみならず、低消費電力化や運用ポリシ達成度を測定するベンチマークを整備する必要がある。また、測定するのみではなく、測定データや運用ポリシが定めるところの性能評価法に対する評価基準をどのように定めるかも、重要な課題である。さらに、上記方法で得られた結果の総合評価方法や総合評価基準についても検討しなくてはならない。

第 8 章　自動チューニングを支援するソフトウェアツール

8.1 自動チューニング支援ツールのモデル

8.1.1 AT支援対象のユーザ

自動チューニング（AT）に関連するユーザを分類する。考慮すべきユーザ層として、次に示す 3 階層がある。これらのユーザ層のそれぞれにとって、有用な自動チューニングを支援するツール（AT 支援ツール）の存在が求められる。

1. エンドユーザ層： AT 機能をもつソフトウェアを利用する人
2. システム開発者層： AT機能をもつソフトウェアを開発する人
3. 研究者層： 自動チューニング技術を研究する人

3 階層のユーザ以外に、自動チューニングに一般的な関心をもつ人々の層が存在する。この層のユーザに対して、AT 機能の利用を促進するAT支援ツールやAT機能をもつソフトウェア（AT ソフトウェア）の利用環境の整備が必要である。なお、この一般的関心層には、自動チューニング機能をもつライブラリ（AT ライブラリ）などの AT ソフトウェアをAT機能の存在を知ることなく使用しているユーザ、アプリケーション分野で性能パラメタに対するハンドチューニングの経験をもつユーザなどがいる。

8.1.2 AT 支援ツールの支援内容

各々のユーザ層が、AT 支援ツールに対して何を期待するかを考える。各ユーザ階層が求める主な支援の枠組みは、以下の通りである。

1. AT ソフトウェアの利用に対する支援
   AT ソフトウェアのエンドユーザに対して、AT機能の利用を容易にするように支援する。
2. AT ソフトウェアの記述に対する支援
   AT ソフトウェアの開発や拡張を行うとき、システム開発者は ABCLibScript のような AT 仕様を記述するための専用言語を必要とする。開発者は、同言語の支援をうけて AT の対象、範囲、目的関数などを記述する。
3. AT ソフトウェアの開発に対する支援
   ATソフトウェアの開発には、多くの試行錯誤が繰り返される。例えば、求解する問題に対して、適切な数値計算アルゴリズムが選択される状況を考える。開発途上でアルゴリズム選択方法が不明であるとき、適切な選択を行うのに必要な情報を収集し提供することが期待される。この場合、支援をうけるのは AT 機構の開発者である。なお、アルゴリズム選択において、エンドユーザの関与がある場合は、エンドユーザも支援の対象となる。

4. AT ソフトウェアの評価に対する支援
   AT 技術に関連して、AT ソフトウェアの有用性、AT モデルの妥当性や有効性など、多くの評価対象がある。そうした評価を効率的に支援するツールの存在は不可欠である。この場合、ツールによって支援を受けるのは、AT モデルの妥当性や有効性検証と改良を目指す研究者、開発者と、AT ソフトウェアの有効性を評価するエンドユーザである。

8.1.3 AT支援のタイミング

AT支援ツールのモデルには、基本的に人が介在する。支援を必要とする人と支援するシステムが存在し、人はシステムから適切な支援を、適宜、受けながら所望の性能を自動的あるいは半自動的に獲得する。支援を受けるタイミングは、対象ソフトウェアを開発するユーザ層によって異なる。例えば、ATライブラリの開発者は、AT 仕様記述言語のディレクティブを手作業で対象プログラムに挿入する。このプログラム記述過程における支援ツールは、AT 仕様を記述する専用言語である。ベクトル化コンパイラ、並列化コンパイラの例に倣うと、ATプログラミング支援環境は重要なAT支援ツールのモデルである。

AT ソフトウェアのインストール時の支援も、不可欠なAT支援ツールのモデルである。対象とするシステムの静的環境条件を考慮して、高性能が得られるようインストールを適切に支援する。ATデータベースの構築を含めて、通常インストールには多くの時間を要する。効率的にインストールし、かつ人手の介在が少なくなるように支援する必要性は高い。単なるプログラム構造の静的な解析だけでは十分な性能が得られない。解くべき問題の入力データやシステムの動的環境条件によって、適切な求解アルゴリズムは異なる。プログラムの実行中にユーザと交信することで、更なる性能向上が可能となる。ATソフトウェアの実行時のAT支援モデルでは、ユーザとの交信の過程で、適切な情報提供とプログラムの修正を考慮しなくてはならない。その場合、修正過程での試行錯誤を前提とする必要がある。

8.2 自動チューニング支援ツールの機能

AT 支援ツールの機能を考える。AT 支援ツール群を用いて自動チューニングを実現するとき、支援ツールに求められる基本機能は性能計測と評価を支援する機能と情報可視化機能である。自動チューニングにおける支援ツール群は、これらの機能を実現しなくてはならない。例えば、アルゴリズム選択の自動チューニングでは、目的とする求解に向けたアルゴリズムは数多く存在し、問題に対してどのアルゴリズムを適用するべきかという判断が迫られる。自動的に選択することが難しい場合は、ツールの

支援を得て半自動的に判断する。最良のアルゴリズムを選択できない場合は、不適切なアルゴリズムの優先度をさげるなどの対応を支援する。以下に、必要とされる AT 支援ツールの機能をあげる。

8.2.1 AT 記述言語

ユーザが対象プログラムに AT 機能を付与する **AT プログラミング**を行う上で、ATソフトウェアを記述する専用言語を用意することは、AT 支援のみならず AT 研究の出発点である。AT 記述言語 ABCLibScript は、それを代表するものである。

8.2.2 AT プログラミング環境

AT ソフトウェアを利用する観点からのプログラミング支援ツール、および、AT 記述言語を用いて AT ソフトウェアを開発する観点からの支援ツール、すなわち、AT プログラミング環境が必要である。有効性の分析・評価を支援するツールの利用においても、AT プログラミング環境が前提となる。これらの AT に特化した記述を容易にする AT プログラミング環境として、ABCLibScript 用の視覚化ツール VizABCLib がある。ABCLibScript のディレクティブ挿入は基本的に手作業であり、AT プログラミング環境としては不十分な状況にある。さらに、ユーザマニュアル、AT ソフトウェアの仕様書・設計書や、一般向けの AT 技術の解説書などのドキュメンテーションの整備も重要である。また、AT ソフトウェアの標本プログラムや、デモンストレーションプログラムなどが必要である。

8.2.3 ATデータベース

性能に影響するパラメタや性能情報を抽出する機構を付加し、そこから得られたログをデータベースに格納する必要がある。このようなログ抽出にはオーバヘッドがかかるため、この機能の利用はユーザが自由に選択できるようにする。AT プログラミングの中で目的とする計算スキーム、アルゴリズムなどが複数あるとき、どれを選択したらよいか指針を示す情報が必要である。ATデータベースは、選択対象となる計算スキーム、アルゴリズムなどの性能パラメタとそれらのデータ値を永続的な知識として保持する。分析・評価結果の情報も、ATの知識としてデータベースに格納する。

8.2.4 性能アナライザ

性能と最適性を追及する AT プログラミングは、動作仕様を満たすことのみを目的とする通常のプログラミングは異なる。AT ソフトウェアの開発、利用の過程において、定量的および定性的分析・評価のフェーズが必要となる。性能アナライザはATデータベースの情報を分析し、その評価結果を再びデータベースに格納する。ここでは、確立された分析・評価手順に従い処理される。性能アナライザのインタラクティブ操作や

分析により生成された情報は、ログとしてATデータベースに格納される。

8.2.5 品質管理手法

AT 支援ツールのデータ分析・評価では、数理的手法、データ計測手法、情報可視化手法などが用いられる。AT を象徴する品質特性は速度と精度である。情報可視化をともなう速度と精度の分析・評価では、旧来からの統計的解析、多変量解析、OR 手法の他、種々の品質管理手法が用いられる。

8.3 自動チューニング支援ツールの現状と問題点

現状として、AT 記述言語 ABCLibScript の開発を支援するための情報可視化ツール VizABCLib を中心に述べる。AT 記述言語 ABCLibScript のATプログラミング環境は、VizABCLib と呼ばれる可視化ツールである。同ツールは、プログラムテスト支援システム、ビジュアルエディタ、性能アナライザから構成される。

8.3.1 プログラムテストの可視化

AT 記述言語に連携してプログラムテストを行う方法は、ソフトウェア工学の一般的手法として知られている。ここでは、AT 機構を付加するための言語と連携したプログラムテストの可能性と、その可視化について述べる。ABCLibScript では、アンローリング段数やブロックパラメタ値の決定において、抽出された最適化パラメタ値をプログラマが必要と思われる範囲、もしくは、AT システムが指定する範囲について、網羅的に探索する。実行時間に相当する対象の出力範囲が明確に定義できれば、プログラムの挙動をコード開発時にチェックできる。問題は、出力結果が異常かどうか判定する手段である。また、出力値の増減や周期性などの動作傾向分析において、入力となる最適化パラメタが1つで結果の出力が1つであれば、図示は単純である。一般には、最適化パラメタが多数で出力も複数となり、その場合の図示は単純ではない。AT におけるプログラムテストの可視化には、以下の機能が要求されるが、現状ではまだ実現されていない。

1. 入力となる最適化パラメタが多数、出力となる値が複数の場合にも、直観的な可視化ができる。
2. ユーザが興味をもつ表示次元に、簡単に切り替えられる。
3. 問題がありそうなパラメタ値の範囲を自動的に検出し、自動的もしくは半自動的に表示の次元を縮小し、3 次元以下のグラフで表示する。
4. テストセットが組み合わせ爆発する場合、現実的なセットを自動生成する。
   - 対象の最適化パラメタ数を削減する機能

- 適切な標本点を自動的にとり、出力形状の概略がつかめる機能

8.3.2 ビジュアライザ

ユーザに解りやすいビジュアライザを提供することは、AT ソフトウェアの開発効率を高める。AT 支援の可視化ツール VizABCLib には、以下の表示機能がある。

1. ABCLibScript の記述がある領域（AT 領域）ごとに見出しを自動的につけ、ユーザがクリックすると指定個所に飛ぶ。
2. ABCLibScript の指示文を色表示する。

8.3.3 実行結果の可視化表示

AT 結果をプログラマへ可視化表示する機能は、ユーザ、開発者がATの有効性を知るために有用である。AT 記述言語処理系により目的コードが生成され、自動実行テストされ得られたログから、AT 効果の可視化情報がプログラマへ伝えられる。VizABCLib では、AT 記述言語 ABCLibScript を用いて、AT 機能を付加されたコードの実行ログを表示する機能がある。また、ループ中に現れる問題サイズNをシステムが自動決定し、その値から、ユーザが見たいログの箇所を指定することができる。最適化パラメタが複数ある場合は、最終的に 2 次元グラフになるまで、ユーザが最適パラメタ値を選択させる機能が付加されている。

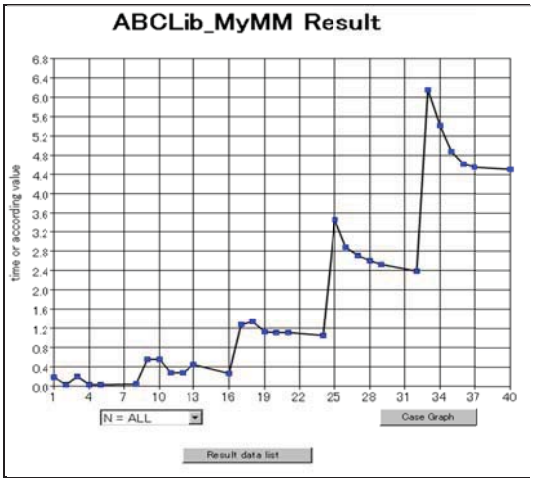

図 8.1 VizABCLib による自動チューニング実行履歴

また、目的関数による予測と AT による実行時間との差異を知ることは重要である。VizABCLib では、多項式の目的関数と問題サイズ N を変動させたときの実行時間を

表示する機能がある。図 8.2 で、目的関数による予測値は点線で、実測値は折れ線グラフで表示されている。

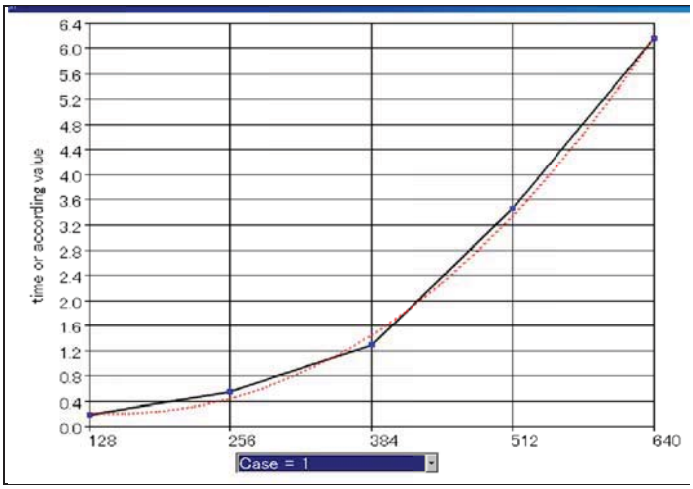

図 8.2 VizABCLib における目的関数による予測と実行時間

8.4 自動チューニング支援ツールに関する課題提案

8.4.1 品質管理手法を用いた線形方程式求解プロセスの分析・評価

AT 支援ツールの構築に不可欠な技術として、品質管理手法を用いた線形方程式求解プロセスの分析・評価を提案する。線形方程式の求解において AT が直面する問題点を、品質管理の方法論にもとづいて分析・評価し、問題解決へと向かわせる。工業製品の製造プロセスと製品の品質を、線形方程式求解のプロセスと計算結果に対応させる。ここでユーザの要求とは計算時間、求解精度などを指しているが、それ以外に所要の計算機資源などもATポリシとして与えられる。

自然現象や工学現象の解析においては、多くの問題が大規模な行列計算に帰着され、この求解に多くの時間が費やされる。一方で、行列計算の求解アルゴリズムは数多く存在する。ここでアルゴリズムの選択を間違えると、数値解が得られるまでに何倍もの計算時間を費やし、最悪の場合は数値解が得られないことがある。このような無駄な時間と労力を削減し、数値計算における生産性を向上させるためには、求解アルゴリズムの性能や特性をあらかじめ戦略的に分析・評価しておくことが重要である。そして、これらを解決する具体的な方法論が必要である。線形方程式求解の求解過程においても、さまざまの問題が生じる。例えば、得られた数値解の精度が悪い、異常に計算時間がかかる、求解途中でエラーとなり解が得られない、どのアルゴリズムを選ぶのが適切なのか、既存のアルゴリズムで解けるのかなどである。このような

場合、実測したデータを分析・評価し、求解アルゴリズムに対して有用なアルゴリズム、あまり有用ではないアルゴリズムなどの優先度をつけることが重要である。あるいは、線形方程式の特性に着目し、実行する計算機環境に対して、アルゴリズムの適・不適を評価することが必要である。これらの情報を知識データベースに蓄積しておき、所要のATが実施されるとき適切に利用されねばならない。

図 8.3 は、線形方程式求解に対する特性要因を示す。ここでは、反復解法の求解状況が悪いときの品質特性が問題点として指摘されている。A～E の各項目は品質特性に影響を与えている要因であり、A～E の各太矢印に向かっている細矢印は要因を組立てている要素を示す。要因分析に際して、品質管理手法、数理手法を用いて、実測データの分析・評価を行っている。さらにこのマトリックス・データを用いてマトリックス・データ解析したとき、数値計算特有の分析・評価方法が新しいツールの提案へとつながる。図 8.3 では、汎用的な求解アルゴリズムや計算プログラムを念頭におき、物理方程式、離散化などをモデル化と見なしている。実際の大規模問題に対する数値計算では、SMASH (Science, Modeling, Algorithm, Software, Hardware)の各要素が重要であり、それらを多軸的に考慮すべきであるという中島研吾の提案がある。物理方程式とは自然法則の定式化であり、品質管理の対象となる領域は人手を介してモデル化された後から始まると考える。すなわち、自然現象の解明を目指す大規模数値計算においては SMASH、人手による求解の活動においては品質管理を念頭におく。

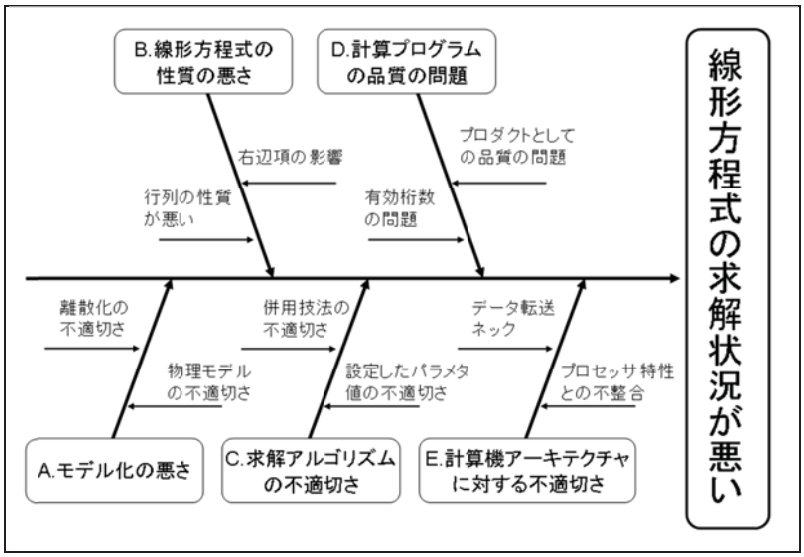

図 8.3 線形方程式求解に対する特性要因

数値計算に対する品質管理的な考え方では、問題、アルゴリズム、計算プログラムの全体を見通して問題点の分析・評価を行う。単なる与えられた問題の分析・評価にとどまらず、これまでに認識されていなかった問題を発見しようとする。従来の数値計算の研究では、理論にもとづく考察が中心に行われ、その正当性は理論に対する実験すなわち数値シミュレーションのレベルで完結させてきた。抽出されたデータのさらなる分析や、体系的にデータを分析・評価する実験科学としての取り組みは、理論科学の不十分な部分を埋めるものである。数値計算は、数理的な根拠を与える数学、実際の実行形態であるプログラム、計算を実行するプラットフォームである計算機の3つの観点にまたがって位置づけられねばならない。これらを踏まえたうえで、森口繁一がいうように数値計算のやり方を工学的に研究する**数値計算工学**として取り組むことが重要であり、自動チューニング研究にも不可欠な研究姿勢である。

8.4.2 AT支援ツール群とその統合システム

ユーザの AT ソフトウェア開発を支援するプログラミング環境、AT 支援ツール群を開発する。これらの設計思想、機能については 8.1 節で説明した通りである。具体的な AT 支援ツール群として、AT 記述言語およびそのプログラミング環境、AT データベース、性能アナライザなどがある。とくに、プログラムテストの自動化であげたように、入力となる最適化パラメタが多数で、かつ出力となる値が複数の場合にも、直観的な可視化ができる機能が望まれる。性能アナライザでは、問題がありそうなパラメタの範囲を自動的に検出し、自動的もしくは半自動的に表示の次元を縮小し、3 次元以下のグラフで表示する機能と、ユーザが興味をもつ表示次元に簡便に切り替えられる操作性を実現する。

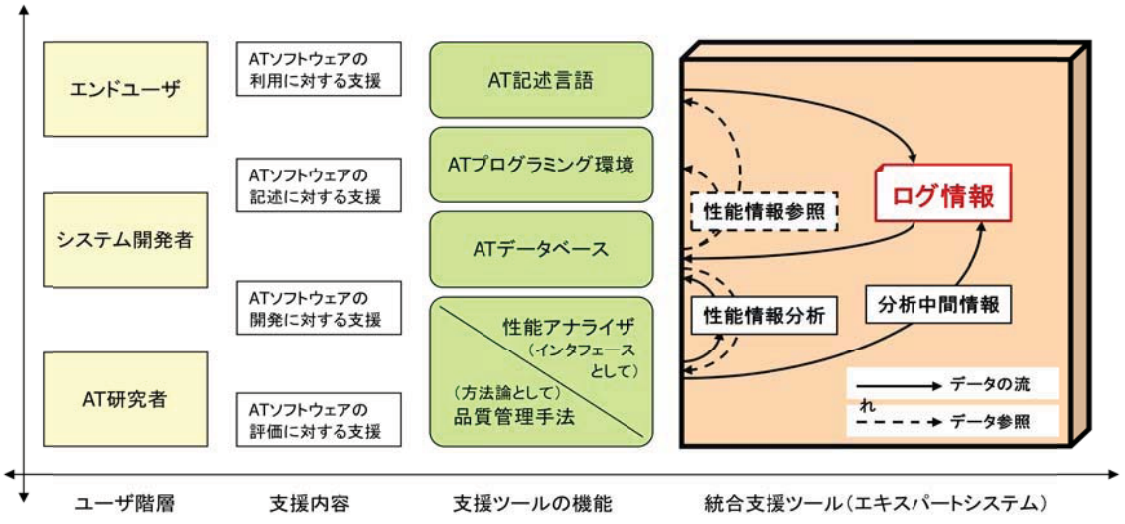

図 8.4 ユーザ層を拡げる AT 支援エキスパートシステム

AT支援ツール群を相互補完し相乗効果を生みだす統合システムを提案する。これは、ユーザ層を拡げる **AT 支援エキスパートシステム**（図 8.4）であり、ユーザの要求を満たす AT 支援ツール群の要素が互いに機能しあい、有用な情報を抽出する。数値計算アルゴリズムの自動選択を目標にしているが、それ以外の分野における性能チューニングに対しても同様のシステムが考えられる。統合システムにおいては、ATに関する情報の管理を行う。実線の矢印はデータの流れ、破線の矢印はデータの参照を表している。データベースに格納された情報量と品質が高まるにつれて、当システムは徐々に性能改善から性能予測に向けたAT機構へと位置づけが変わる。

第 9 章　自動チューニングが適用可能な応用分野

9.1　目的と背景

本章では，自動チューニング技術（以下、AT 技術）の適用対象を、応用（アプリケーション）全体、すべての計算機環境、情報技術全般に広げて、AT 技術の潜在的な可能性を明らかにする。AT 技術は、現在、限られた分野でしか利用されていない。本来は、情報技術に関連するさまざまな分野での活用が考えられる、大きな可能性を秘めた技術である。実際、世の中には、プログラムの自動的な最適化が必要とされている応用分野が数多くある。例えば、すでに見てきたように科学技術計算においては、適切な性能チューニングによって、実行時間や所要記憶量を大幅に削減できる例が多い。また、最近では計算機の低消費電力化が大きな話題となっている。適切なチューニングを行うことで、実際に利用する計算機の規模を縮小したり実行時間を短縮することによって、消費電力量を削減できる場合が多い。このようなチューニングを自動化し低コストで行う AT 技術は、有力なソフトウェア基盤技術として期待される。

AT 技術の適用対象を広げる場合、次の 3 つの方向への拡張が考えられる。

1. 対象とする応用（アプリケーション）の範囲を広げる。
2. 対象とするプラットフォームを広げる。
3. 新しい目的関数を考える。

本章では、主として 1.の方向への拡張を考える。具体的には、アプリケーションとして数値計算応用、信号処理応用、データベース応用、Web 応用、人工知能応用、実世界コンピューティング応用、バイオインフォマティクス応用をとりあげ、それぞれについて AT 技術適用の現状と問題点、さらに今後の可能性を議論する。なお、2.の方向への拡張としては、プラットフォームを単体プロセッサ、マルチコア型プロセッサ、クラスタシステム、分散システム、広域分散システムへと広げることが考えられる。これについては、自動チューニングが対象とする計算機環境について議論する第 7 章で扱う。3.としては、数値計算応用で主に目的関数として使われている実行時間、精度、所要記憶量に加え、性能安定性、消費電力量を含めるなどが考えられる。

9.2　個々の応用において検討すべき事項

9.2.1　検討項目

個々の応用（アプリケーション）への AT 技術の適用を考えるにあたっては、以下の項目について検討し、明らかにしなくてはならない。

1. アプリケーションの目的と課題
2. 最適化を行うにあたっての目的関数、最適化すべきパラメタ
3. AT 適用対象の特性
4. AT 技術として支援すべき対象
5. AT により得られる効用
6. AT 適用にあたって予想されるコストと障壁
7. アプリケーションから見た AT 支援を実現する手段（ツール）
8. 当該アプリケーションは AT 向きであるといえるか？
9. 今後の方向性と研究課題

9.2.2 AT 向きか否かを判断する根拠

上記の 5.から 8.について、もう少し詳しく説明する。あるアプリケーションが AT 向きであるといえるためには、AT により得られる効用と AT 適用のためのコストを比べたとき、効用がコストより大きいという関係が成立しなければならない。各アプリケーションに AT 技術を適用した場合の効用と、適用する際のコストを定量化する必要がある。アプリケーションによっては、効用があっても、現状ではコストの方が大きく、AT の適用が引き合わないこともある。その場合でも、AT 適用のコストをさげる新しい技術あるいはツールの開発により、適用できるようになる可能性がある。

AT の効用とコストを評価するための基準について考える。まず、AT の効用が大きいのは、次のような条件を満たす場合である。

(E1) プログラムの実行時間が非常に長い、あるいは多くの計算機資源を必要とする。
(E2) プログラムの実行時間、あるいは計算機資源の制約が非常に厳しい。
(E3) パラメタあるいはアルゴリズムの選択により、性能（実行時間以外の目的関数も含む）が大きく変化する。
(E4) プログラムが異なる計算機環境で使用される。
(E5) プログラムが長期間にわたって使用される。

例えば、AT の適用事例としてよくあげられる行列計算ライブラリは、条件(E1)、(E3)、(E4)、(E5)を満たす。そのため、AT の効用が非常に大きく、AT 向きであるといえる。コストについて考えると、AT 適用のコストが小さくなるのは、次のような場合である。

(C1) 計算（あるいは処理）の大部分が少数の計算カーネルに集中している。
(C2) パラメタを変化させることで、プログラムを異なる計算機環境向けに最適化することができる。

(C3) 異なる計算機環境に適するさまざまなアルゴリズムを、系統的に自動生成することができる。
(C4) プログラム構造が明確で、ドキュメントが整備されている。逆に、すでに開発者がいなくなっているレガシコードなどでは、AT 化のコストが大きい。

例として、行列計算ライブラリの場合は、条件(C1)、(C2)を満たすため、AT 適用のコストが小さい。また、高速フーリエ変換 FFT の場合も、条件(C1)、(C3)を満たすため、コストは比較的小さくできる。

最後に、AT 適用のコストを引き下げるための技術について考える。まず性能測定のための実験計画法、性能モデリング、最適化など、AT 全体に共通する数理的基盤があれば、個々のプログラムに AT を適用するコストは小さくなる。また、既存のプログラムに簡単なディレクティブを挿入することで AT 適用プログラムへの変換を容易にする AT 記述言語(ABCLibScript など)の存在も、AT 化のコストを大きく引き下げる。AT 化の障壁や対象プログラムの特性は分野ごとに異なるため、各アプリケーションに特化した AT 化のための支援ツールを開発することで、AT が適用可能な範囲をさらに広げることができる。以下では、各アプリケーションに関して、これらの点も検討する。

9.3 狭義と広義の AT 応用

AT 技術の潜在的適用範囲は極めて広い範囲におよぶが、これを狭義の AT 応用と広義の AT 応用とに分類する。**狭義の AT 応用**とは、プログラムの機能(入力と出力の関係)は変えずに、内部のアルゴリズム、実装方式、パラメタなどを与えられた計算機環境にあわせてプログラム自体が最適化することにより、速度向上、所要記憶量削減、消費電力量削減などを実現する応用例である。ATLAS を初めとする多くの自動チューニング型数値計算ライブラリやデータベースの自動チューニングなどは、狭義の AT 応用に相当する。本章では、数値計算、信号処理、バイオインフォマティクスにおける分子動力学法、データベースなどの応用が狭義の AT 応用を扱っている。

広義の AT 応用とは、入力に対するプログラムの出力も変化することを許し、内部のアルゴリズムやパラメタなどを計算機外部の環境(入力も含む)に合わせてプログラム自体が最適化することにより、よりよい出力結果を得ようとする応用である。本章では、Web システム応用におけるリコメンデーションシステム(ユーザの好みに合った検索結果を出力)、ゲームなどの人工知能応用(対戦相手人間のレベルを判断して互角になるよう強さを自動調節)、実世界コンピューティングにおけるロボット制御(与えられた環境のもとで目的関数を最大(小)とするように自動調整)などが、広義の AT

応用に相当する。このような広義の AT 応用では、チューニングを計算機が自動的に行うことが本来の研究目的であることから、わざわざ＜自動チューニング＞と呼ぶべきかについては議論がある。しかし、使われる技術や数理に共通性があることと、ATの適用範囲をなるべく広げて考えるという方針から、本章では広義の AT 応用についてもとりあげることにする。

9.4 数値計算応用

さまざまな分野における数値計算をとりあげる。ただし、AT 技術の重要な適用先である数値計算ライブラリについては第 6 章で扱うため、ここでは対象から除外する。本節では、必ずしもライブラリ化が適切ではなく、アプリケーション自身に AT 機能を組み込む必要がある応用を中心とする。数値計算が必要とされる分野は極めて幅広いが、本節ではそれらを網羅的にとりあげることはしない。AT 活用により大きな成果が得られている分野として**量子化学計算**をとりあげ、なぜ成功したかを 9.2 節の視点に基づき分析する。また、同様の方法論により効果が期待できる分野としてコンピュテーショナル・ファイナンスをとりあげ、新しい AT 応用の具体的提案を行う。

量子化学計算のうち、分子軌道法による計算をとりあげる。**分子軌道法**は、応用分野の専門家である量子化学者と計算機科学者との協働により、AT 応用の実用化に向けた研究が進められている数少ない例の 1 つである（G. Baumgartner et al.: Synthesis of High-Performance Parallel Programs for a Class of Ab Initio Quantum Chemistry Models, Proceedings of IEEE, Vol. 93, No. 2, pp. 276-292, 2005）。分子軌道法における処理は、電子軌道の基底関数からハミルトニアン（全エネルギ演算子）を計算する多電子積分と呼ばれる部分と、ハミルトニアンの固有値・固有ベクトルを計算する部分からなる。通常、前者が計算時間の圧倒的な部分を占めるため、以下では前者のみを対象とする。

9.4.1 目的と課題

多電子積分では、複数個の電子軌道とポテンシャルの積に対して多重積分を行うことにより A_{ijkl} のような高階テンソルを求め、さらに複数個のテンソルに対して縮約を行う。例えば、次の式は 4 個の 4 階テンソルを 6 個の添字に関して縮約し、1 個の 4 階テンソルを求める計算を示す。

$$S_{abij} = \Sigma_{cdefkl} A_{acik} \times B_{befl} \times C_{dfjk} \times D_{cdel}$$

これを単純にプログラム化すると添字が 10 種類なので 10 重の DO ループになるが、各添字は数 10～数 1,000 個の値をとるため計算量が膨大となる。この計算を高速か

つ使用可能な記憶量の範囲で行うことが課題である。

9.4.2 目的関数と最適化パラメタ

上記の課題を達成するには、以下の点を考慮する必要がある。

（o1）縮約の計算量を削減する。
（o2）縮約演算の中間結果を記憶するための領域を削減し、できれば主記憶上に格納できるようにする。
（o3）中間結果が主記憶上に載らない場合でも、ディスクには収まるようにする。
（o4）分散メモリ型並列計算機上で実行する場合、プロセッサ間通信を削減する。

これらが目的関数および制約条件となる。最適化パラメタは、以下の通りである。

（p1）ループの順番（ループの実行順序、ループの入れ替え）
（p2）ループの融合
（p3）ループのタイリング
（p4）分散メモリ型計算機におけるデータ配置

この問題では、それぞれの目的関数に対し、最適なパラメタ値が一般には異なる。従って、計算とディスクアクセスの速度比、計算とデータ転送の速度比など、対象とする計算機の特性に応じて各目的関数に重み付けをし、実行時間が最小となるようにする。ただし、(o3)は絶対的な制約であり、優先的に考慮する必要がある。

9.4.3 適用対象の特性

多電子積分は、対象とする系あるいは量子化学の手法ごとに計算式が異なる。従って、あらかじめ決めた何種類かの計算式を最適化し、ライブラリとして提供するだけでは解決にならない。ユーザが定義した任意の式の計算を最適化する仕組みが必要である。一方、計算自体はテンソルの縮約という形式のきまった計算であるため、計算式を表現する専用の高水準言語を設計することが可能である。このような高水準言語は AT のために役立つだけでなく、計算式の記述と解読を容易にするので、多くのユーザに受け入れられる素地がある。

9.4.4 AT として支援すべきこと

AT としては、ユーザが計算式を入力するだけで、上記(p1)～(p4)のパラメタを自動的に最適化し、目的(o1)～(o4)を満たすようなプログラムを自動生成する機構を提供することが期待される。

9.4.5 AT により得られる効用

多電子積分は、分子軌道法において計算時間の 95%以上を占めるといわれる負荷の重い計算である(9.2.2 の項目 E1)。また、計算法によっては、必要とする中間結果の大きさがディスク容量をこえてしまい計算不可能になるなど、計算機資源に関する制約が厳しい計算でもある(E2)。また、計算法、とくにループの順番により、計算量と所要記憶量は数 1,000 倍以上の幅で変動する(E3)。これに対する最適化は、従来、経験の豊富な量子化学者が長時間かけて手作業で行っていたが、これを AT 技術により自動化できればその意義は大きい。さらに、分子軌道法は広く使われているため、汎用的な AT システムが構築できれば、その恩恵は広い範囲におよぶ(E4)。

9.4.6 適用コストと予想される障壁

多電子積分では、ユーザが定義した任意の計算式に対して最適化を行わなければならない点に、行列計算ライブラリなどとは違った難しさがある。この場合、最適化を行うにあたって、ループの順番変更や融合の可能性をシステムが自動的に抽出しなければならない。しかし、計算式が通常のプログラミング言語で記述されていると、複雑なループに対しては解析が困難となり、これら最適化のための情報を抽出できない可能性がある。また、パラメタ数が多く、各パラメタのとりうる値の数が多いため、パラメタの組み合せの数が爆発的に増大し、最適パラメタ値の探索が困難となるという問題点もある。これは、多くの AT 応用と共通する問題点である。

9.4.7 AT 支援ツール

これらの問題点を解決し、AT の適用を可能にするツールとして、多電子積分の記述のための高水準言語、パラメタ最適化エンジン、コード生成装置の 3 つを組み合せることが考えられる。実際、Baumgartner らの TCE(Tensor Contraction Engine)では、これらの機能をもつ AT 支援ツールを構築している。多電子積分の記述のための高水準言語を用いて計算内容を記述することにより、通常のプログラミング言語で書いたプログラムからは抽出困難なループの順番変更や融合の可能性などに関する情報を抽出することを可能としている。組み合せ論的爆発に対処するため、高水準言語で書かれたプログラムをもとに、階層的な最適化、各階層での動的計画法あるいはメタヒューリスティクスの利用などを行う。演算量最小化、演算量を保ったままでの使用記憶量削減、データ分散方式の決定、中間結果がディスク容量をこえる場合はディスク使用量削減(演算量増加を許容)、データ参照の局所性の向上という各段階を、この順に(必要なら前の段階にもどって)行うという階層的な最適化の枠組みを用いている。コード生成については、最適化エンジンで求めたパラメタ値にもとづき、通常のコンパイラに入力可能なプログラムを自動生成する。

9.4.8 AT 向きかどうかの検討

多電子積分を AT により最適化することができれば、その効用は非常に大きい。一方、AT 適用にあたっては先に述べたような障壁があるが、それらは支援ツールにより解決の目途がたっている。以上により、多電子積分の計算は極めて AT 向きといえる。

9.4.9 今後の方向性

TCE グループによる研究は非常に進んでいる。すでに分散メモリ向けの並列化やキャッシュ向けの最適化などの機能を備えたプロトタイプシステムが量子化学者によって使用され、それを用いて新たな量子化学の計算手法が開発されている。また、TCE は、NWChem や UTChem 2003 などの量子化学計算ソフトウェアに組み込まれて配布されている。今後の方向性としては、対象を SIMD 型超並列システム、FPGA、非均質クラスタ、Gridなど多様な計算機環境へと広げること、目的関数として計算精度、消費電力量なども扱えるようにすることなどが考えられる。

9.4.10 提案: 多期間ポートフォリオ最適化への AT 適用

数値計算分野における新しい AT 応用の具体的課題提案として、コンピュテーショナル・ファイナンスにおける**多期間ポートフォリオ最適化問題**をとりあげる。本問題は、与えられた資金をn種類のリスク資産(株式など)と無リスク資産(現金など)に分散投資し、時間 T 後の期待効用を最大化することが目的である。ただし、期間 T を m 個の区間に分割し、区間ごとにポートフォリオの構成を変更可能とする。本問題は、状態空間を離散化して動的計画法により解くことができるが、計算量は n の指数関数のオーダと極めて大きい。この計算を、高速・高精度に少ない記憶量で行うことが課題である。本問題における計算の主要部は、状態空間の各点における期待値計算、すなわち、多次元の数値積分である。計算におけるパラメタは、数値積分手法(格子点数、格子点位置、重み)、数値積分の加速手法(高速多重極子展開法など)、ループのタイリングなどである。これらにより計算量、精度、計算の効率性(キャッシュミス、通信量など)が大きく変動する。従って、問題の特性、計算機の特性、要求精度に応じてこれらのパラメタを最適化することが AT の課題である。

AT 適用にあたっての障壁としては、まず、パラメタ選択の自動化のための知見が未だ不十分であることがあげられる。例えば、適切な数値積分の手法や高速多重極子展開法は、リスク資産の収益率の分布関数の形により大きく異なる。従って、これらを自動的に定めるには、経験にもとづく知識データベースや、理論にもとづく計算式を整備する必要がある。一方、AT 機能を備えたプログラムの実装においても課題がある。アルゴリズム選択やループのタイリングを最適化する機能を備えたプログラムを書くことは、ABCLibScript など AT 記述言語を使えば原理的には可能である。しかし、

本問題の場合は、数値計算ライブラリの場合と異なり、プログラマが一般には高性能コンピューティング（HPC）の専門家ではない。そのため、 ABCLibScript が使いこなせない可能性が高い。これらの問題点を解決し、AT の適用を可能にする 1 つの方法として、量子化学計算の例にならい、次の 4 つを組み合せることが考えられる。

1. パラメタ選択のための知識データベース
2. 問題と解法を高水準に記述するためのモデリング言語
3. パラメタ最適化エンジン
4. コード生成装置

このようなシステムの実現性については、知識データベースの構築・利用には、連立 1 次方程式の反復法向けの AT システム SALSA（http://icl.cs.utk.edu/salsa/）の開発で得られた知見が大きく役にたつ。また、モデリング言語に関しては、ポートフォリオ最適化のユーザが LINGO、NUOPT などの数理計画ソフトウェアに用意されているモデリング言語に慣れていることから、比較的受け入れられやすいと考える。

研究計画としては、まず短期的な課題として、知識データベースの構築と、それにもとづくパラメタ値選択の自動化という基礎技術を確立する必要がある。これにより、上記で述べたような支援ツールの開発が可能となる。一方、今後の計算機の能力向上やアルゴリズムの進歩により、長期的に見ると動的計画法による最適化がより広く普及し、ポートフォリオ最適化のソフトウェアパッケージに組み込まれる可能性も高い。その場合は、ライブラリと同様、AT を組み込む主体はパッケージの開発者となるので、AT 支援ツールを利用することで、より柔軟な AT 機能が搭載できる。

9.5 信号処理応用

信号処理で用いられる計算は多岐におよぶが、ここではもっとも基本的な計算として、離散フーリエ変換をはじめとする**線形変換**をとりあげる。離散フーリエ変換は、AT の応用に関して活発な研究がなされてきた分野であり、FFTW、UHFFT、SPIRAL など多くの研究成果がある。また、AT 応用の効果が期待できる計算として、近年注目を集めている独立成分分析をとりあげる。**独立成分分析**は高次元テンソルのある種の正準分解として解釈できるが、計算量が多くかつ演算集約型の計算であるため、最適化の効果は大きいと考えられる。また、計算時間や計算機資源に制約のある実時間応用で使われることが多いため、その点からも AT の効用は大きいと期待される。

9.5.1 目的と課題

信号処理における線形変換では、入力ベクトル $\boldsymbol{x}$ に対して定数行列 F をかけることに

より、出力ベクトル yを求める。変換の具体例としては、離散フーリエ変換、ウェーブレット変換、ウォルシュ・アダマール変換、フィルタなどがある。x, y のサイズをそれぞれ m, n とすると、単純な行列ベクトル積として計算した場合の演算量は O(mn)である。しかし、多くの線形変換では、F を対角行列、置換行列、サイズ 2 のデータに対する行列の直和など、基本変換行列の積として表すことが可能である。基本変換行列による乗算の演算量は一般に O(max(m,n)) なので、Fを少数個の基本変換行列の積に分解できれば、それらを順に x にかけることで変換の演算量は大きく削減できる。また、この分解の仕方により記憶へのアクセスパタンも異なり、精度も異なる。そこで、与えられたサイズの線形変換を与えられた計算機上で行うに際して、なるべく演算量が少なく、同計算機に適したアクセスパタンをもち、かつ精度の高い分解を見つけることが課題となる。さらに、変換を FPGA などのハードウェアで実現する場合は、回路規模、消費電力量を小さくすることも課題となる。

9.5.2 目的関数と最適化パラメタ

上記の課題を達成するには、以下の点を考慮する必要がある。

（o1） 変換の演算量を削減する。
（o2） 対象とする計算機に適したアクセスパタン（連続参照など）を使う。
（o3） 変換の精度を一定値以上に保つ。
（o4） 変換の回路規模を小さくする（ハードウェアによる実現の場合）。
（o5） 消費電力量を小さくする（ハードウェアによる実現の場合）。

これらが目的関数および制約条件となる。最適化パラメタは、以下の通りである。

（p1） 変換の基本変換への分解法
（p2） 基本変換の実装法（ループの実行順序、ループの入れ替え、ループ融合）
（p3） ハードウェアによる実現の場合のパラメタ（パイプラインの深さ、並列性）

この問題では、それぞれの目的関数に対し、最適なパラメタ値が一般には異なる。従って、用途に応じて目的関数に重み付けをする必要がある。例えば、実時間応用では、高速化のために(o1), (o2)を優先し、指定された時間内に処理が終わるようにする。一方、ハードウェアで実現する場合は、回路規模を指定された大きさ以内に収めることが絶対的な制約となるため、(o4)が優先事項となる。

9.5.3 適用対象の特性

線形変換に関しては、フーリエ変換のように広く使われる変換は高性能なライブラリが提供されており、ライブラリに対する自動チューニングの研究も早くから行われてい

る。しかし、用途によっては自分で線形変換を設計して使う場合もあり、そのような場合にも適用できるより柔軟な AT の枠組みが必要である。また、線形変換は実時間応用で使われ、ハードウェア化されることも多いため、AT 技術を用いた最適なハードウェアの実現が期待される。一方、計算自体は単純な行列ベクトル積であり、数式を使って簡単に記述できる。従って、変換を記述するための専用の高水準言語を設計し、それを入力として AT を行うことが可能である。

9.5.4 AT として支援すべきこと

ユーザが線形変換の定義式を入力するだけで、上記(p1)～(p3)のパラメタを自動的に最適化し、目的(o1)～(o5)を満たすプログラムあるいはハードウェアロジックを自動生成する機構を提供することが望まれる。

9.5.5 AT により得られる効用

線形変換は、物理現象のシミュレーションのような大規模計算から音声認識のような実時間応用に至るまで、さまざまな応用において使われる極めて汎用性の高い計算である(9.2.2 項の項目 E5)。シミュレーションにおいては、FFT 演算が計算時間の半分程度を占める場合も多く、高速化の効果は大きい(E1)。一方、実時間応用においては、定められた時間内に計算を終えることが要求される(E2)。さらに、ハードウェアで実現する場合は回路規模に制約があり、また、消費電力量の削減も求められる場合が多い。このように、線形変換はさまざまな計算の中でも、最適化が強く求められている計算であるといえる。また、アルゴリズムによって記憶アクセスパタンなど計算の特徴が大きく変化し、速度が大きく変わる計算でもある(E3)。従来、線形変換に対する最適化は、標準的なフーリエ変換に対して FFTW などの AT ソフトウェアがある他は、人手で行われてきた。従って、より広い種類の線形変換に対して適用でき、速度以外にもさまざまな目的関数が扱え、かつハードウェアロジックの生成も可能な AT システムが構築できれば、その恩恵は大きい。

9.5.6 適用コストと予想される障壁

信号処理ではユーザが設計した線形変換に対して最適化を行う必要があり、この点に行列計算ライブラリとは違った難しさがある。さらに、線形変換の行列 F の基本変換行列への分解は極めて自由度が高い。従って、これらの自由度を適切に表現し、問題サイズに対して指数関数的に増える分解候補の中から、最適な分解を選ぶことが必要である。これは、きめられた数のパラメタを最適化するのに比べ、より難度の高い問題である。FPGA などで実装を行う場合は、これに加えて、最適なハードウェアロジックの自動生成に関する難しさがある。

9.5.7 AT 支援ツール

以上の問題点を解決し、AT の適用を可能にするツールとして、以下の 4 つを組み合せることが考えられる。

1. 線形変換の記述のための高水準言語
2. 変換行列のさまざまな分解を系統的に生成するシステム
3. 分解集合の中から目的に応じて最適な分解を選ぶ最適化エンジン
4. コード生成装置

既存研究の SPIRAL（Markus Pueschel et al.: SPIRAL: Code Generation for DSP Transforms, Proceedings of IEEE, Vol. 93, No. 2, pp. 1-42, 2005）では、この組み合せにより線形変換のための AT 支援ツールを構築している。高水準言語 SPL は、意味のわかりやすい形で線形変換を記述することを可能にする。離散フーリエ変換、ウェーブレット変換、ウォルシュ・アダマール変換、フィルタなどの基本的な線形変換が要素として用意され、ユーザはそれらを組み合せることで、自分の望む線形変換を記述できる。計算内容を高水準で記述することにより、次項(b)で述べる分解の自動生成を可能とする。高水準言語で定義された線形変換に対して、あらかじめ定義された分解規則を適用し、基本線形変換の積に分解する。分解規則としては、例えば、サイズ kl の FFT をサイズ k の FFT の l 個の直和、ひねり係数の乗算、サイズ l の FFT の k 個の直和、置換の積に分解する規則（Cooley-Tukey FFT の原理）などがある。基本線形変換への分解の仕方は一意的ではなく、どのような順序でどの規則を適用するか、また、規則に内在するパラメタをどうとるかにより、さまざまな分解が可能である。各分解を木に対応させ、可能な木の群を系統的に生成することにより、多様な分解を生成する。得られた分解の中から、目的関数（計算量、記憶アクセスパタン、回路規模など）に応じて最適な分解を選ぶ。分解は木によって表現されることから、階層的な最適化あるいは動的計画法の利用が可能である。また、探索効率をあげるため、強化学習や進化的計算などの手法も利用している。得られた最適な分解に対応するアルゴリズムを生成する。また、ループの順序変更、融合など、局所的な最適化を行う。これにもとづき、通常のコンパイラに入力可能なプログラムを自動生成する。ハードウェア実装の場合は、局所的な最適化にもとづき、ハードウェアロジックを生成する。

9.5.8 AT 向きかどうかの検討

一般の線形変換を AT により最適化することができれば、その効用は非常に大きい。AT 適用にあたっては 9.5.6 で述べたような障壁があるが、それらは 9.5.7 で述べた支援ツールにより解決の目途がたっている。以上より、線形変換の計算は極めて AT 向きといえる。

9.5.9 今後の方向性

SPIRAL グループによる研究は非常に進んでおり、単一プロセッサ向けの最適化に加え、分散メモリ向けの並列化、Cellプロセッサ向けの最適化、FPGAによる実装のためのハードウェアロジック自動生成などの成果が得られている。SPIRAL のホームページでは、ユーザが指定した線形変換に対し、さまざまなプロセッサ向けの最適化コードを自動生成し、提供している。今後の方向としては、SPIRAL で培われた技術を、線形計算などより広い分野に適用することが考えられる。

9.5.10 提案: 独立成分分析への自動チューニングの適用

信号処理分野における新しい AT 応用の具体的提案として、独立成分分析をとりあげる。**独立成分分析**とは、m 本の統計的に独立な入力信号 $\mathbf{x}(t)$の線形変換として m 本の出力信号 $\mathbf{y}(t) = A\mathbf{x}(t)$ が与えられた場合に、線形変換 A に関する知識を用いることなく、$\mathbf{y}(t)$から $\mathbf{x}(t)$を復元する手法の総称である。独立成分分析は比較的新しい手法であるが、音声信号分離、移動体通信における信号分離、気象データの解析などさまざまな分野で用いられ、その有効性が確認されている。主成分分析などの伝統的手法に比べて計算量が多く、また実時間応用も多いことから、処理の高速化が求められている。

独立成分分析のアルゴリズムは、さまざまなものが提案されている。重要なアルゴリズムとして、高次のキュムラントが存在する定常過程向けのキュムラントテンソルの対角化にもとづくアルゴリズムと、弱定常過程向けの相互相関行列の同時対角化にもとづくアルゴリズムがある。前者では、テンソルをある次元方向にスライスしてできる行列の組の特異値分解を、スライス方向を順次変えながら繰り返す。後者では、複数の時間差に対する相互相関行列を、ヤコビ法の拡張版を用いて近似的に同時対角化する。両手法とも 3 次元以上の配列に対する処理であり、多大な計算を要する。その分、ブロック化、ループアンローリング、ループの順番変更、演算順序の自由度など、AT により最適化可能なパラメタも多い。

AT の適用にあたっては、可能な数多くの演算順序を系統的に生成する手法の開発、それらの演算順序の中から演算量、所要記憶量、キャッシュ利用効率などの目的関数を最適化する順序を選ぶ手法の開発などが必要となる。これには、線形変換や多電子積分を計算するために開発された手法が応用できる。さらに、プログラムの最適化にとどまらず、FPGA 向けの最適ハードウェアロジックの設計への展開も考えられる。音声信号分離、移動体通信における信号分離などの応用では、限られた計算機資源のもとでの実時間処理が要求されるため、AT 技術により高速化・省記憶量化が実現できれば効果は大きい。

9.6 データベース応用

クラスタリングによるデータベースの性能チューニングは、1990 年代初頭から活発に研究されている。研究プロジェクトは多数存在し、いくつかの製品化の例もある。データベースチューニングの手法としては、インデックスチューニング、バッファリングによるチューニング、クラスタリングによるチューニング、並列化によるチューニングなどがある。これらのうち、本節では、近年、重要性を増しているクラスタリングによる性能チューニングを扱う。

9.6.1 目的と課題

データウェアハウスの増加にともない、一時的に利用されるデータベースも増加傾向にある。ところが、一時的なデータベースには、従来の静的クラスタリングでは性能チューニングが不十分であることが分かってきた。また、データベース全体をチューニングするのはコストが高い。そこで、部分的なデータベースに対し、動的な統計情報を活用したクラスタリグによる性能チューニングが必要になってきている。N 個のもっとも頻繁にアクセスされるオブジェクトを計測し、それを他のデータよりも高速にアクセスできるようにクラスタリングする。このとき、N の大きさ、データベースの範囲、アクセス頻度、計測間隔などが自動チューニングにあたっての課題となる。

9.6.2 目的関数と最適化パラメタ

関係データベースは行と列の 2 次元でモデル化される。クラスタリングによるデータベースチューニングは、レコードクラスタリング（水平方向）と属性クラスタリング（垂直方向）の 2 種類がある。レコードクラスタリングでは、次式で与えられる探索要求 Q(i)の発生頻度 F(Q(i))とページ数 P(Q(i))の積分を目的関数として、これを最小化する。

$$(o1) \qquad C = \sum F(Q(i)) \times P(Q(i))$$

別の目的関数を用いた最適化としては、単一オブジェクトがアクセスされる頻度を Heat (o2)、また、ペアとなっているオブジェクトで一連のトランザクション中、同時にアクセスされる尤度を Tension (o3)と呼び、それぞれを最小化する場合がある。これらの目的関数を最小化するよう、レコードを再パーティショニングする。属性クラスタリングでは、属性集合をアフィニティにもとづいてグループに分割する。属性 i と属性 j の同時アクセス頻度を a(i, j) としたとき、次式を最小化する。

$$(o4) \qquad ME = \sum a(i, j) \times \{a(i, j+1) + a(i, j-1) + a(i+1, j) + a(i-1, j)\}$$

9.6.3 適用対象の特性

上記の一時的データベースは常に変わるため、一度最適化したパラメタがすぐに無

効になる。また、運用実態にあわせたチューニングが求められるため、運用実態をモニタリングしつつインデックスパラメタを最適化する、いわゆる実行時自動チューニングが必要となる。また、実行時の自動チューニングは、前提としてシステム運用管理を念頭におく必要があり、そのため、本来はシステムを運用管理する人間も含めたチューニングを行うべきと考えられる。

9.6.4 AT として支援すべきこと

上記のような最適化を行うため、実際の運用時において性能をモニタリングしつつ、その場で最適化することが AT として支援すべき内容になる。

9.6.5 AT により得られる効用

番号	効用	本件の場合
E1	プログラムの実行時間が非常に長い、あるいは多くの計算機資源を必要とする場合	○（一般に DB システムは大規模）
E2	プログラムの実行時間、あるいは計算機資源に関する制約が非常にきつい場合	△（現実にはゆとりをもって実装される）
E3	パラメタ／アルゴリズムの選択により、性能（実行時間以外の目的関数も含む）が大きく変化する場合	◎（大幅に変わる）
E4	プログラムが異なるハードウェア環境で使われる場合	ー（無関係）
E5	プログラムが長期間にわたって使われる場合	ー（無関係）

9.6.6 適用コストと予想される障壁

番号	効用（少ないコストでできる）	本件の場合
C1	計算（あるいは処理）の大部分が少数の計算カーネルに集中している場合	ー（無関係）
C2	パラメタを変化させることで、プログラムを異なるハードウェア環境向けに最適化することが可能な場合	◎（一時 DB に適用可能）
C3	異なるハードウェア環境に適する多様なアルゴリズムを系統的に自動生成することが可能な場合	◎（一時 DB に適用可能）
C4	プログラム構造が明解でドキュメントが整備されている場合（すでに開発者がいなくなっているレガシコードなどでは、AT 化のコストが大きい）	◎（一時 DB に適用可能）

9.6.7 AT 支援ツール

実際の運用時に、性能をモニタリングしつつ、上記で定義された最適化のための目的関数の入力情報を計測する支援ツール、運用管理者向けの支援ツールが必要である。

9.6.8 AT 向きかどうかの検討

90 年代からアクティブな研究分野であり、また、データ量の増加やデータベースの種類、活用の幅が大変広く、現在も極めて AT 研究に向いている分野といえる。

9.6.9 今後の方向性

今までの研究活動により、最適化のための種々のメトリックスが定義され、初期段階の高性能化は実現されている。しかし、具体的にどれくらいの頻度でシステム状態をモニタリングすべきかについては未解決である。モニタリングにおいては、具体的な実運用システムを目標にした実証的研究が望ましい。また、その際には、システム運用管理の手順や実態を考慮した(半)自動チューニングとするように、目的関数を設計すべきである。

9.7 Web システム応用

9.7.1 目的と課題

Web 技術の発展により、電子商取引(Electronic Commerce)の分野が拡大している。Web 環境のもとで、書籍や旅行などの莫大なアイテム候補から、推薦する商品、パッケージ、組み合せを選んで提示して欲しいというニーズがある。情報爆発の時代といわれるようにアイテムの選択肢が数限りなくあり、ユーザがどれを選べばよいのか、どうやって自分の好みにあった商品を選ぶべきかという悩みが生まれている。そのような状況で、Web ユーザが満足を得られるアイテムを提案する**Web リコメンデーションシステム**を実現することが、本分野の重要な研究課題となる。

9.7.2 目的関数と最適化パラメタ

本分野は適用範囲が広く、さまざまなパタンが考えられる。ここでは、目的関数をきめる指標として、以下の項目をあげる。

(o1) ユーザの満足度を高める。
(o2) 多すぎる提案とはならない。
(o3) アイテムのランキングを表示する。

これらを数学的な目的関数とするため、例えば、以下のパラメタを入力とする関数とし

て定義する必要がある。

（p1）提案商品のアイテム数
（p2）各アイテムのコスト
（p3）問い合せリードタイム
（p4）各アイテムの組み合せ（例えば、オプショナルツアーなど）

9.7.3 適用対象の特性

最終的にユーザが満足する提案ができることが目的だが、ユーザの満足度という指標が曖昧であり、それをいかに数学的表現にするかが課題となる。また、最適化の目的関数が多層的になることも特性としてあげられる。例えば、問い合せリードタイムは商品問い合せ過程での目的関数になり、かつ、ユーザの満足度の最適化パラメタにもなる。このように、最適化が多層になっている。

9.7.4 AT として支援すべきこと

曖昧な満足度を、いかに数学的に表現するかが課題となる。そのため、AT としては業務プロセスの中から重要なパラメタを抽出し、目的関数を明らかにするフレームワーク化が支援すべき課題となる。

9.7.5 AT により得られる効用

番号	効用	本件の場合
E1	プログラムの実行時間が非常に長い、あるいは多くの計算機資源を必要とする場合	○～△（Web 環境の規模による）
E2	プログラムの実行時間、あるいは計算機資源に関する制約が非常に厳しい場合	△（現実にはゆとりをもって実装される）
E3	パラメタ／アルゴリズム選択により、性能が大きく変化する場合	○（大幅に変わる）
E4	プログラムが異なる計算機環境で使われる場合	ー（無関係）
E5	プログラムが長期間にわたって使われる場合	ー（無関係）

9.7.6 適用コストと予想される障壁

番号	効用（少ないコストで可能）	本件の場合
C1	計算（あるいは処理）の大部分が少数の計算カーネルに集中している場合	ー（無関係）
C2	パラメタを変化させることで、プログラムを異なる計算機環境向けに最適化することが可能な場合	◎（適用可能）

C3	多様な計算機環境に適するさまざまなアルゴリズムを系統的に自動生成することが可能な場合	◎（適用可能）
C4	プログラム構造が明解でドキュメントが整備されている場合（すでに開発者がいなくなっているレガシコードなどでは、AT 化のコストが大きい）	○（目的関数が明確であれば適用可能）

9.7.7 AT 支援ツール

ユーザの満足度を高めるための支援ツールとして、リコメンド内容のスコアリングなどがあげられる。

9.7.8 AT 向きかどうかの検討

現時点では、ユーザの満足度という関数が必ずしも明確ではないため、確度の高い判断は難しい。ただし、情報システムは、人間という要素を含めた広がりのあるものに発展しつつある。リコメンデーションという分野が情報学で注目されているように、チューニングという対象を人間の満足度という曖昧な領域も包含する方向に向かうのであれば、AT の対象として検討すべき分野になると思われる。

9.7.9 今後の方向性

ユーザの満足度を数学的な目的関数として表現し、その後、AT で提案された情報が、本当にユーザにとって満足のいくものか、ユーザからのフィードバックを得る必要がある。このフィードバックをいかに科学的に評価するかが、よりよい目的関数を構築するとき重要となる。この点が、本分野が AT の研究領域として成立するうえで重要であり、今後の研究課題である。

9.7.10 提案： Web システムへの適用を目指した自動チューニングパラダイム

Web システムは、計算機だけに閉じた系ではなく、それを使う人間も含めた最適化が求められている。従って、行列計算ライブラリという狭い範囲を対象とした自動チューニングパラダイムを、人間系を含めたパラダイムへと発展させる必要がある。とくに、Web システムではユーザが多岐にわたるため、より広い範囲の計測とより高度な制御が必要になる。そこで、2 つの観点での AT 新パラダイムに関する研究課題を提案する。下記の 1.は計測技術、2.は制御技術に関する提案である。計測と制御という両輪がうまくかみ合うことが、本提案の実現における重要なポイントとなる。

1. コンテキストの数学的表現
   人間系を含めた最適化を行う際に、各種の機械化されたシステムのモニタリングはもとより、人間の意図をくむ、いわば、気が利くモニタリング技術が必要であ

る。また、システムを使う人間が置かれている状況を正しく表現する技術も必要である。この意図や状況といった、いわゆるコンテキスト情報を数学的に表現する方法が重要である。

2. Web リコメンデーションにおける自動チューニングパラダイムの適用
これまで、計算機環境の複雑化ゆえに、AT 技術が生まれてきたという背景がある。AT 技術は、これから先、以下のような発展段階を経ることが予想される。

段階	コンピューティングパラダイム	動機・理由
−1	Parallel Computing	ベクトル処理、キャッシュ、SPMD などの計算機アーキテクチャが出現
0	Automatic Tuning	並列プログラム実装パタンの爆発的増加による実装コストの低減化
1	Policy-based Computing	条件やポリシ（If…then…else…）に適応する自動チューニング
2	Context-aware Computing	ユーザの意図や状況にもとづいたポリシの入力が可能な自動チューニング
3	Recommendation in Context	ユーザの意図や状況を察知し、ユーザより先回りして、ユーザが望む情報を提案

9.8 認識システム応用

認識とは、与えられたデータから意味づけされた特徴を抽出し、それが属する先験的範疇に分類することである。ここでは、自動チューニングの人工知能分野への応用として、学習、推論、認識といった人間の行為を複合的に利用するパタン認識を扱う。

9.8.1 目的と課題

パタン認識では、人間に代わって計算機が音声データや画像データを見て特徴を抽出し、あらかじめ与えられた範疇に分類（識別）する。しかし、人間であれば瞬時に認識できるようなものでも、計算機ではいくら時間をかけても認識できないといった事例がよくある。文字認識であれば、それぞれの文字の特徴をデータベース化してそれを利用することが可能であるが、画像の場合には特徴をデータベース化すること自体が非常に難しくなる。また、単に対象物の名称を識別するというのではなく、人物の画像を示して、計算機にその人物の年齢や顔の表情から、怒っているとか笑っているといった感情をいいあてさせることはさらに難しい。ここ数年、防犯カメラや監視カメラが小型化し安価になってきたことから、人物画像の認識が急速に普及し始めている。これまでは、1週間あるいは1ヶ月間といった一定期間の映像をビデオ媒体に保存し、

何か事件や問題が起きて始めて過去の映像を見直すという使われ方が多かった。このような場合、犯罪の抑止にはなっても、事件の解決には結びつかないことがある。最近は、映像をハードディスクに保存するタイプも増え、映像をリアルタイムに解析することも可能となってきた。犯罪や事故の発見を即時に行えるようにすることが課題としてあげられている。

9.8.2 目的関数と最適化パラメタ

パタン認識では、一般にパタン観測、前処理、特徴抽出、パタン識別、後処理といった5段階の処理を行う。パタン観測の部分では、カメラからの映像を計算機にとりこむ部分が含まれる。このとき、計算機の処理能力に応じて、映像の解像度や単位時間のフレーム数を適宜変更するが、その過程が自動チューニングの対象となる。最適化を行うにあたっての目的関数は、認識率の高さである。しかし、1 回のパタン認識に多くの時間を費やせる訳ではないので、一定時間内における認識率の高さが目的関数となる。最適化すべきパラメタは 5 段階の処理それぞれに含まれるが、それらの処理にどれだけ時間を使うかといった時間配分も最適化パラメタに含まれる。また、利用する計算機の能力を最大限に発揮するプログラムの自動選択も必須である。さらに、認識に使うアルゴリズムや知識データベースの方式が複数存在する場合には、どのような組み合せで利用するかということも自動チューニングの対象となる。

9.8.3 適用対象の特性

パタン認識の対象はさまざまである。同じ画像認識にしても、例えば、文字、車両、人の顔の認識では、その手法は大きく異なる。しかし、対象が異なったとしても、基本技術で共通化できる部分は多い。

9.8.4 AT として支援すべきこと

前処理や特徴抽出の部分では、FFTを用いた周波数スペクトル分析や各種統計手法、行列の固有値解析などの処理が行われる。これらは汎用的な数値計算として考えられるので、数値計算における自動チューニング技術が適用可能である。また、対象物が予想できない場合に、同時に複数の対象物を認識する状況も考えられる。例えば、テレビに映し出されている映像はある瞬間だけをとらえれば、そこに何が映っているかは予測がつかない。このような場合は、考えられるすべてのパタンと一致しているかどうかを調べなくてはならない。ある程度、認識対象がきまっている場合には、処理を高速化することも可能である。また、対象として調べる範囲が多い場合には、すべてのパタンをきめられた順番で調べるよりも、出現確率の高い順に調べることで高速化を実現することも可能である。この場合にも、自動チューニング技術における知識データベースが活用できる。

9.8.5 AT により得られる効用
それぞれについて、効用が大きい場合を〇、小さい場合を×とする。

（E1）扱うデータ量が多いためプログラムの実行時間が長い。◎
（E2）実時間処理しなければならないため、時間制約は厳しい。〇
（E3）パラメタやアルゴリズムの選択により、性能は大きく変わる。◎
（E4）プログラムが異なる計算機環境で使われることは余りない。×
（E5）プログラムが長期間にわたって使われる。〇

9.8.6 適用コストと予想される障壁
それぞれについて、適用コストが小さい場合を〇、大きい場合を×とする。

（C1）処理の大部分が少数の計算カーネルに集中しているとはいえない。×
（C2）異なる計算機環境に適用させることは可能である。アプリケーションと計算機環境が一体となっている場合が多いので、本件の処理に適した計算機環境を選択することで、さらにコストがおさえられる。〇
（C3）異なる計算機環境に適するさまざまなアルゴリズムを自動生成することは可能である。〇
（C4）プログラム構造は明確とはいえないため、コストがかかる可能性がある。×

処理の高速化は、試行回数を増やすことで最適な手法を選択していくことで実現できる。しかし、正しい認識が行われたかどうかの判定までを自動化することは難しい。また、認識が正しく行われなかった場合の知識データベースの自己補正手段に関しても、自動化は簡単ではない。

9.8.7 AT 支援ツール
パタン認識の 5 段階の処理をそれぞれ高速化するためのツール、間違った規則を除去するためのツール、知識データベース化のための特徴量を抽出するためのツールなどが必要である。

9.8.8 AT 向きかどうかの検討
各処理の内部では汎用的な数値計算に帰着される部分も多いため、処理の高速化に関しては AT 向きである。また、これまで人手で行っていた処理を自動化でき、処理時間の大幅な短縮と認識率の向上につながるため AT 向きである。

9.8.9 今後の方向性
膨大な数の静止画像や膨大なデータ量となる動画像から、検索キーワードを入れてそれに関連する画像を抽出できるシステムへの応用が考えられる。

9.9 ゲームコンピューティング応用

ゲームは、機械学習や推論という人工知能分野の典型的事例である。人が自然に行っている学習という行為を機械に行わせる機械学習によってゲーム機は強くなり、学習の結果得た規則を知識として、未知の入力に対して何らかの結論を推論により導き出す。具体的には、多くの標本データを与えて機械がそれを解析し、そこから有用な規則を抽出して蓄える。蓄えた規則を知識として利用し、以降の推論に活用する。

ゲーム機で使われる計算機の性能が大幅に向上し、ゲーム機の市場規模も大きくなり、ゲーム企業間の競争が激しくなってきた。ゲーム機の機能と性能が増すにつれて、**コンピュータゲーム**の開発には多くの時間と人件費がかかるようになってきている。ゲームにおける自動チューニングとして求められるのは、計算機の機能と性能を最大限に活用したゲームの開発を、短時間かつ低コストで容易に行えるようにすることである。しかし、一番重要なことはゲームの面白さである。ゲームの面白さは、ストーリ、難易度、サウンド効果、映像効果、操作性、意外性、話題性などのいくつもの要因が複雑に絡み合っているが、これらを自動チューニングによって決定できることが期待される。ただし、面白いと感じるかどうかは人によっても大きく異なるため、数値化することが難しい問題でもある。ゲームにはいろいろなジャンルがある。その中でもチェスやチェッカなどのテーブルゲームは、人工知能の研究の始まりとともにその研究対象として扱われており、古い歴史がある。テーブルゲームの研究では人に勝つような強いプレイヤを育てること、パズルゲームでは問題をいかに早く解くかが研究の主な対象とされてきた。しかし、ここではゲームの面白さという観点から、自動チューニングの枠組みがゲームコンピューティングに適用できるかどうかを考えたい。

9.9.1 目的と課題

現在、**計算機将棋**は十数年前と比較にならないぐらい強くなっており、今ではほとんどの人が計算機に勝てない状況になっている。そこで、対局している人の棋力にあわせて計算機側の棋力を調整できるプレイヤを実現することを考える。人の棋力の指標として段や級が使われるが、ここではチェスの世界で広く使われているレーティングという言葉を使用する。計算機将棋の目的は、指定するレーティングの棋力をもつプレイヤの実現ということができる。問題は、いかに人と見分けがつかないように指すかということである。単純に考えれば、すでに計算機は十分強いレベルに達しているので、何回かに 1 回、乱数などで指し手を決めるといったことで棋力の調整ができるが、それでは不自然である。例えば、ある棋力の人と対戦するときに 2 回に 1 回は本気を出し、2 回に 1 回はわざと負けるようにすれば勝率は 5 割となり一見、同じ程度の

棋力と判断されることになるが、これでは人間らしさとはかけ離れている。

9.9.2 目的関数と最適化パラメタ

将棋では、駒の損得、駒の効率、玉の固さ、手番の重要性などが評価の要素項目となる。目的は、同じレーティングの人間と対戦して序盤から終局までほぼ互角に対戦できるようにすることである。各レーティングに応じた目的関数の各パラメタ値、次の手の探索における読みの深さ、実行する読み手の選択などが最適化すべきパラメタとなる。

9.9.3 適用対象の特性

将棋のような評価関数を用いた探索問題では、多くの場合、評価関数のパラメタ値の決定において、機械学習の手法がとりいれられている。将棋であれば、対局の棋譜データなどからパラメタ値を自動調整する方法などが用いられている。

9.9.4 AT として支援すべきこと

計算機環境の特性を最大限に活用するために最適なプログラムコードを選択する部分は、従来の AT 技術が利用できる。また、最適化パラメタ値の決定については、完全に自動化することが理想であり、そのために AT 技術が有効である。

9.9.5 AT により得られる効用

それぞれについて、効用が大きい場合を〇、小さい場合を×とする。

（E1）数値計算と異なり、通常の計算機は探索を得意としていない。そのため実行時間が長い。〇
（E2）人との対戦のため、応答速度は早いほどよい。また、1 台の計算機で複数の人と同時に対戦できる方がよい。◎
（E3）パラメタ値やアルゴリズムの選択により、性能は大きく変わる。〇
（E4）プログラムが多様な計算機環境で使われる。〇
（E5）プログラムは長期間にわたって使われる。〇

他の効用として、将棋の初心者から上級者までが楽しめるアプリケーションとなることがあげられる。また、それぞれの棋力の人にとって学習効果が期待でき、棋力の向上にもつながる。

9.9.6 適用コストと予想される障壁

それぞれについて、適用コストが小さい場合を〇、大きい場合を×とする。

（C1）処理の大部分が少数の計算カーネルに集中しているとはいえない。×

（C2）多様な計算機環境に適用させることが可能である。〇
（C3）異なる計算機環境に適するさまざまなアルゴリズムを自動生成することは可能である。〇
（C4）プログラム構造は明確とはいえないため、コストがかかる可能性がある。×

勝ちそうになったから急に弱い手を指すというのでは不自然であり、最初から最後まで一貫して同じ強さで指すことが求められる。それをどう実現するかが課題である。また、戦法や玉の囲い方には数多くの種類があるが、どれをどのくらいの頻度で利用するかも考慮しなくてはならない。ただ、どのパラメタ（値）が勝敗に起因したのかを判断することが難しいため、多くの標本データが必要となる。

9.9.7 AT 支援ツール

棋譜データから目的関数のパラメタ値をチューニングするためのツール、棋譜データから読みの深さを計るためのツール、棋譜データから次の読み手の選択をどう行ったかを推測するためのツールなどが必要である。

9.9.8 AT 向きかどうかの検討

計算機環境の特徴を調べて、それにあった性能向上を追求する点は従来の自動チューニング技術の得意とするところである。例えば、思考ルーチンでは、同じルーチンを何度も実行することになる。この性質は、特定のループで実行時間の大部分を占めるといった一般的な数値計算と共通するものであり、性能チューニングには適している。最適化すべきパラメタの自動調整については、何度も試行を繰り返すことで可能である。また、計算機同士が対戦することでパラメタの自動調整も可能である。実際、強い将棋ソフトウェアにおいては、自分同士で何度も対戦してパラメタを自動調整する方法がとられている。これらのことからも、自動チューニング技術を適用しやすい分野であるといえる。

9.9.9 今後の方向性

計算機囲碁に関しての研究は、まだまだこれからである。囲碁の場合は、チェスや将棋以上に探索空間が広いため、探索を深くすることが現実的には難しくなる。そこで、盤面上の碁石の配置をパタン化して次の指し手を生成するといったことも行われている。しかし、目的関数におけるパラメタの自動調整に関しては、将棋における自動チューニング技術が囲碁にも適用できる。

9.10 実世界コンピューティング応用

実世界コンピューティングの分野は、ロボティクス、画像・音声などのメディア処理、図形表示・描画の処理を含む。いずれも最近の計算機の高性能化によって、高機能化が実現されており、さらなる技術革新に向けた社会的な要請も強い。ここでは、**ロボティクス**における自動チューニングの適用について考える。

9.10.1 目的と課題

ロボティックス分野において、自動チューニング技術はロボット制御におけるさまざまな達成目的（経路計画、障害物回避、揺れ止めなどの物理的な安定化）を達成するための基礎技術として利用されている。機械制御のために導入したファジィ制御、および、ニューロ制御における学習において、自動チューニングが適用される。ファジィ制御においては、どの入力変数が、どの程度、出力値に影響するのかを定義した内部パラメタ値制御用の**メンバシップ関数**が重要な役割をもつ。デルタ関数や三角関数など、出力関数に影響する入力変数の影響範囲を限定した関数を採用することで、メンバシップ関数の形状決定に要する計算量を削減することが多い。

自動チューニングのパラメタの実現値として、推論規則（ルール）が適用されることも多い。どのように操作するかの規則は if-then の並びで与えられる。規則は、経験則やアドホックな実装で事前決定されることが多い。メンバシップ関数と推論規則は別の概念であるが、お互いが出力に影響する。適切なメンバシップ関数と推論規則の設定は、一般には容易でない。メンバシップ関数を推論規則と独立して設定する方式もある。いま、x_1,…,x_m を入力、y を出力とし、推論規則を次のように与える。

Rule i: if x_1 is A_{i,1} and … and x_m is A_{i,m}
then y is w_i (i=1,…, n)

A_{i,j} は前件部のメンバシップ関数、w_i は後件部の実数値である。ここで、A_{i,j}は、推論規則ごとに独立している点に注意する。この方法では、性能パラメタの調整のための計算量が削減される。推論結果 y は次式となる。

$$y = A_{i,1}(x_1) \cdot A_{i,2}(x_2) \cdot \cdots \cdot A_{i,m}(x_m)$$

9.10.2 目的関数と最適化パラメタ

■ファジイ制御

ファジィ制御を行う場合、与えられた入力に対しある目的関数を定め、この目的関数値が高くなるように制御する内部パラメタを調整する。メンバシップ関数を制御するパラメタ、および規則を形成するためのパラメタが、ファジイ制御におけるチューニング対象パラメタとなる。与えられた入力データから、適するメンバシップ関数の形状に影

響するパラメタを決定する。それにともない、出力値が調整され、適切な規則が自動的に選択される。この規則選択のための調整、すなわち、メンバシップ関数の形状と適切な関数パラメタの選定が、ファジィ制御における自動チューニングとなる。

■ニューロ制御

ニューロ制御では、入力と出力との間に、重みづけのある**ニューラルネット**を定義する。ある目的関数を定め、その目的関数値が大きくなるように重みを定める。この重みが、チューニング対象のパラメタとなる。与えられた入力データから、目的関数の値が大きくなるように重みを自動調整することが、ニューロ制御における自動チューニングとなる。

■ファジイニューロの融合方式

デルタルールによるファジィ推論の自動チューニング手法と障害物回避への応用（野村 博義, 林 勲, 若見 昇、日本ファジィ学会誌, Vol.4, No.2, pp.379-388,1992）では、入力層、中間層、出力層からなるネットワークを形成し、各層はニューラルネットで結ばれている。また、同一の層に相当するノード内は、ファジィ制御にもとづく推論規則を適用する。従って、メンバシップ関数との結合による推論規則が形成される。入力層→中間層にはメンバシップ関数のパラメタが存在し、中間層→出力層にはニューラルネットのパラメタが存在する。すなわち、ファジィとニューロを融合した方式となる。移動ロボットの動的障害物回避問題においては、以下のパラメタがある。

1. 入力パラメタ： 移動ロボットとの距離、移動ロボットと目標物の角度、移動ロボットと障害物の距離、移動ロボットと障害物の角度
2. 出力パラメタ： 移動ロボットのハンドル角の変化量
3. 目的関数： 目的点に達した時間 f（入力パラメタ群）、メンバシップ関数

9.10.3 適用対象の特性

ファジイ、ニューロ制御ともに、すべてのパラメタに対する組み合せを求めると計算量が爆発し、現実的な時間でパラメタのチューニング、すなわち学習ができない。そのため、ルール独立のメンバシップ関数の形状を定めるパラメタと、ニューラルネットの重みを決めるパラメタの同時決定を行う手法などの導入により計算量を削減し、実時間処理を可能としてきた。メンバシップ関数の組み合せを限定する方式が、計算量削減のため提案されている。計算量の削減方式の開発が自動チューニング以前の課題となっており、パラメタの自動チューニングをしないと手法自体が実現できない。

9.10.4 ATとして支援すべきこと

メンバシップ関数やニューラルネットの重みづけを自動化しないと、汎用性のある実用システムが作れない。汎用的な学習機能を実現するための要素技術として、自動チューニングは必須である。

9.10.5 AT適用により得られる効用

汎用性が高く、目的を達成するロボット制御システムを構築できる。

9.10.6 適用コストと予想される障壁

推論規則の生成方式やメンバシップ関数の形状を固定すると、効率のよい自動チューニングが実現できる。一方、初期規則の作成の際に高度な経験知が必要となり、この設定自体が人、時間、費用面などのあらゆるコスト高へとつながる。これを緩和するためには、開発者が半自動で規則生成できる支援ツールの開発が必要となる。

9.10.7 AT支援ツール

ツールに期待される支援内容は、以下の通りである。

- 初期状態の推論規則の設定
- ニューラルネットの最適化
- メンバシップ関数の最適化
- 探索空間の削減

9.10.8 AT向きかどうかの検討

AT の効用について、事由(E3)、(E4)、(E5)が相当する。(E3)は、自動チューニングにより、目的達成の質が向上する。(E4)は、ロボティクスにおいては、個々のロボットにおいて、まったく同一の状況(実世界の物理状況、ロボットの部品規格、ロボット内部の制御状況)はない。汎用的な手法を開発しなくてはならない。工学的理由から、一度開発した制御アルゴリズムやプログラムは、長期間利用する(利用されるべき)なので、(E5)は相当する。AT のコストについて、事由(C2)が相当する。(C2)は、ニューロ、ファジイの目的関数の重みなどの性能パラメタは、異なる実世界の環境に適用するために定められたものである。

9.10.9 提案: 推論規則の自動生成

ファジイやニューロを用いた制御手法について調べることは、自動チューニングの適用手法を開発する際の参考となる。論文(家電製品へのファジイ制御の応用状況、林勲、特集解説、T. IEE Japan, Vol.113-C, No.7, 1993)によると、家電製品へファジィ制御を適用する際の規則、すなわち制御方針である if-then の並びの調整、および学習

や制御対象の変化に対する適応性を確保する方式は、以下の 3 通りに大別される。

1. 設計者が設計支援ツールを用いて、初期状態の推論規則を設定する。
2. ファジイ制御を用いて、オンライン繰り返しの学習により自動調整する。
3. ニューラルネットワークとの融合により、自動的に推論規則を決定する。

1.は、自動チューニング機能付きのソフトウェアを作る際の開発者の方針（ポリシ）を半自動的に決定し、自動チューニングシステムに自動設定する方式に類似している。2.は、実行起動前時、および実行時自動チューニングで扱う自動チューニングに類似している。3.は、推論規則の自動調整であり、ソフトウェア自動チューニングの究極的な機能に類似しているが、ここでの規則は導出方式が固定化されたものである。一般にファジィ、ニューロ制御において導出される規則の生成方式に対して、汎用性のある手法はあまり提案されていない。

ファジィ規則の学習に関する研究は、以下の 2 つに分類できる（遺伝的退化アルゴリズムGA[d]によるファジィルールの構造学習における退化速度の自動調整に関する検討、知能と情報、日本知能情報ファジィ学会誌、Vo.16, No.1, pp.33-43, 2004）。

1. 固定されたファジィ規則構造において、メンバシップ関数の形状や規則のファジィラベルなどの規則パラメタを最適化
2. 規則パラメタだけでなく、必要に応じて規則を追加・削除するなどの操作により、規則数に代表される規則構造を最適化

ファジィ制御における自動チューニングはすべて 1.の範疇に入るものである。また、2.の方がより柔軟であり汎用性はあるが、自動チューニング対象のパラメタが変動したり、組み合せ爆発をおこすなど、定式化が難しい。

現状でのニューロやファジィ制御においては、固定された規則構造におけるメンバシップ関数の形状情報を自動チューニングするという範囲にとどまっている。より抽象度の高い推論規則生成に関する研究が、今後は推進されていくものと予想される。新しい研究の方向性として、規則自身の自動生成、規則を生成する方式の自動生成が考えられる。

9.11 バイオインフォマティクス応用

バイオインフォマティクスの分野では、さまざまな手法、アルゴリズムが利用され、それらを一般的に論じることは難しい。以下では、バイオインフォマティクスの各分野に

おいて用いられている 6 つの手法について、その内容と AT 適用の可能性を検討したのち、AT の適用が有効と考えられる分子動力学法について考察する。**分子動力学法**は、生体分子を対象とした数値シミュレーションの代表的手法である。他の手法に比べて、AT 適用にあたっての論点が明確であり、関連した研究例も見られる。

■配列解析

バイオインフォマティクスの分野において、類似配列の検索は、遺伝子やタンパク質の機能を調べるのに用いられる。とくに、ホモロジ検索では、動的計画法による最適アラインメントや、BLAST、FASTA などの発見的アルゴリズムが用いられる。また、モチーフ検索では、オートマトンやプロファイル法、さらに機械学習を用いたパタン認識法が用いられる。これらの手法を用いた単発的な検索では、プログラムの実行時間がそれほどかからず、アルゴリズムも個々の目的に特化しているため、AT の効果はそれほど大きくない。AT が有効となる可能性があるのは、全ゲノム的な、膨大な数の配列群を対象にしたゲノム解析である。例えば、配列群のクラスタリング、配列群からの特徴的パタン抽出（ギブスサンプリングや MOTIF 法）など、膨大な数の配列群を繰り返し走査する処理があげられる。また、上記の処理も含めて、配列から機能を予測・推定するという枠組みの中では、配列の特徴、要求される精度、機能の性格によって、最適な手法・アルゴリズムを自動的に選択する AT が考えられる。ただし、この場合、結果の正しさや精度を自動的に判定することは困難であり、それをもとにしたパラメタ最適化については多くの課題が残されている。

■ゲノム情報解析

アセンブルは、ゲノム配列決定における重要なバイオインフォマティクスの技術である。アセンブルには、反復配列除去、オーバラップ検出、クラスタリング、コンティグ形成などの処理が含まれる。これらは一般に長時間の計算を必要とし、実際に多くの計算機資源が投入されている。しかしながら、使用されるアルゴリズムは比較的限定されており、並列計算も重要ではあるが、独立した操作を並列に実行する処理が多いため、AT の効果はそれほどない。現在、次世代シークエンサの研究が注目されており、そのためのバイオインフォマティクス技術の必要性も指摘されている。しかし、手法そのものが模索されている段階であり、AT の適用は今後の課題であろう。遺伝子発見やゲノム特徴解析（繰り返し配列、トランスポゾン解析など）には、配列解析と共通する点が多く、機械学習などの手法が用いられている。しかし、性能はそれほどクリティカルなものではなく、AT の効果はそれほどない。ゲノム進化解析・機能解析については、長時間の計算を必要とすることが多く、アルゴリズムによっては、パラメタや内部のアルゴリズムによって性能が大きく変わるものがある。しかし、それぞれ目的が限定されており、コストをかけて AT を行う効果がそれほどあるとは考えられない。近年、注

目されているメタゲノム解析では、微生物間の相互作用解析、さらには複雑系の解析が必要となり、そのためのバイオインフォマティクス手法が検討されている。膨大な計算量を必要とし、複数のアルゴリズムを組み合せて適用する必要性があるため、利用者が希望する知識抽出を行ううえで、適切なアルゴリズムを自動的に選択・調整できる AT 技術が望まれる。しかし、そのための実践的知識と技術の集積は、今後の課題である。

■トランスクリプトーム解析・プロテオーム解析
マイクロアレイのデータ解析に関する処理は、多変量解析、クラスタリングなどの統計処理、機械学習が主体である。性能はそれほどクリティカルなものではなく、むしろ、遺伝子の関連性の解析やネットワーク推定などでは、いかに実験データから意味のある情報を抽出するかが重要である。データの性質から、適切なデータマイニング手法を自動的に選択・調整できる AT 技術が望まれる。ただし、結果の妥当性と精度を評価するための知識が不足しており、自動的に判定することは現状では困難である。

■ネットワーク情報解析
ネットワーク特徴解析やネットワーク比較・探索などについては、プログラムの実行時間がそれほどかからず、アルゴリズムも個々の目的に特化しているため、AT の効果は、それほど大きくない。細胞シミュレーションでは、モデルやパラメタの設定などについて、さまざまな手法が提案されている。それぞれにATが有効となる可能性もあるが、使用されるアルゴリズムは個々の目的に特化しているため、AT 導入のコストに見合った効果を得ることは難しい。

■タンパク質立体構造解析
タンパク質の構造比較については、プログラムの実行時間がそれほどかからず、アルゴリズムも個々の目的に特化しているため、単発的な比較・探索では、AT の効果はそれほど大きくない。配列比較の場合と同様、全ゲノム的な多数の構造群を対象にした解析にATが有効となる余地がある。タンパク質の構造予測のうち、2 次構造予測、アミノ酸残基の埋もれ度予測、不規則領域予測、相互作用部位予測では、配列比較や機械学習の手法が用いられ、その基本的な枠組みは共通している。しかし、性能はそれほどクリティカルなものではなく、AT の効果はそれほどない。立体構造予測については、代表的な手法がいくつか存在し、どれも多大な計算時間を必要とする。しかし、結果の妥当性と精度を自動的に評価することが難しく、そもそも、アルゴリズムおよびパラメタの選択と結果の関係が明確ではなく、現時点では AT の有効性は小さい。
■複合体構造予測

ドッキング予測とも呼ばれるこの予測では、geometric hashing や clique search などの局所的な形状マッチングや高速フーリエ変換などの手法が用いられる。これらが実行時間のほとんどを占めているため、AT の可能性は基本的な手法の適用によるところが大きい。タンパク質の立体構造予測、ドッキング予測とも、その構造精密化の段階では、より精度の高いポテンシャルエネルギ関数を用いた最小化計算が行われる。分子シミュレーションは膨大な時間を必要とするため、シミュレーティッドアニーリングなどの手法が用いられる。最急降下法やニュートンラフソン法は、極小解を求めるだけでよい場合や簡易な精密化の手段として用いられているが、これらの手法に対する AT の適用は必然的に限定される。シミュレーティッドアニーリングでは、徐々に探索の範囲を限定する温度スケジューリングが重要であり、そのパラメタは最適化の対象となり得る。側鎖モデリングなどの離散的な探索問題については、遺伝的アルゴリズムやそれに類する手法が用いらる。これら同手法の計算が実行時間のほとんどを占めており、AT の適用はこれらに対して一般的に有効と考えられる。

9.11.1 目的と課題

分子動力学法は、原子の運動を、原子間に働く力を計算しながら、個々の原子に対するニュートンの運動方程式を積分することにより求める手法である。半導体プロセスシミュレーション、高分子材料の物性解析・設計、熱流体現象の解析などにも用いられる。その基本的な手順はＮ体問題と呼ばれる問題の数値解法である。天体シミュレーションなどにも適用されるが、ここでは生体分子のシミュレーションに的を絞って論じる。生体分子の機能の解明を目指す分子動力学法は、以下の機能をもつ。

1. 生体分子構造のモデリング: 生体内に近い環境での生体分子の構造モデルを求める。
2. 生体分子のダイナミクス解析: 生体に近い環境での生体分子や周辺の溶媒分子の動きを求める。
3. フォールディングシミュレーション: タンパク質のフォールディング過程の再現とその原理の解明を目指す。
4. 原子間相互作用の解析: 生体分子の分子間相互作用を原子レベルで解析する。
5. 物理化学量の計算: 多数のコンフォメーションから、統計力学の公式を使って、自由エネルギ変化、定温圧縮率、定圧比熱などを計算する。

材料系などにおける応用に比べて、定常状態でのシミュレーションが多く、力場が複雑であり、長時間のシミュレーションを必要とすることが多いという特徴がある。分子動力学法では、内部で使用されるアルゴリズムの選択肢、パラメタの数は膨大であり、精度、実行時間、記憶量、適合する並列化手法などは大きく異なる。分散システム、

大規模並列計算機、クラスタシステム、専用ボードなど、さまざまな計算機環境で実行されるため、それぞれに適した最適化を行う必要がある。また、精度、実行時間の他、シミュレーションの目的や対象によって異なる目的関数による最適化を行うことも重要である。

9.11.2 目的関数と最適化パラメタ

分子動力学法の基本的な手順は、有限差分法であり、離散的なタイムステップで、原子に働く力、原子の座標を繰り返し計算する。タイムステップの間隔、マルチタイムステップ（力の種類に応じて、タイムステップの間隔を変える手法）の設定、力の計算のカットオフの設定、電荷の計算法およびそれに関連したパラメタなどが最適化の対象となる。例えば、CrockerらのMDSimAidでは、Particle Mesh Ewald法の並列化において、与えられた精度の制約条件のもとで、分子レベルのカットオフ値、グリッドサイズ、グリッドレベル、レベル間の補間次数などのパラメタの自動最適化を行っている。さらに、力場パラメタの選択、アンサンブルの設定と計算、効率的な構造サンプリング、溶媒効果の計算モデルなどを含めると、対象や目的に応じたアルゴリズムの選択肢と、最適化の対象となり得るパラメタが多数存在する。最適化を行うにあたっての目的関数は、シミュレーションの精度と実行時間である。一般に、シミュレーションの目的や対象とする系の大きさなどにより、必要な精度を維持しつつ実行時間を短縮することが求められ、両者のトレードオフが存在する。実行時間については、データの参照局所性を意識した高速化や並列計算が重要である。例えば、力の計算をプロセッサ間で分散させることや、分散メモリ型並列計算機上で実行する場合はプロセッサ間通信を削減するなどの最適化が考えられる。

9.11.3 適用対象の特性

長時間の繰り返し計算を含むため、実行時における自動チューニングも効果的と考えられる。原子の空間上の移動、原子間の相互作用のパタンの変化などに対応させて、プロセッサへのデータ配置や力の計算のカットオフ値、マルチタイムステップにおける時間間隔の適用範囲の調整などを行うことができる。

9.11.4 AT として支援すべきこと

最適化すべきパラメタは多数存在するので、性能に影響を与えそうなパラメタを選択し、モデル化する枠組みが重要である。Nelson らは、パラメタ最適化にあたり容量パラメタ、バランスパラメタ、機能仕様パラメタの 3 つのパラメタを用意し、それぞれ最適値を探索するアルゴリズムを提案している。分子動力学法では、一時データのキャッシュサイズ（記憶階層を考慮したバッファサイズ）、分割セルのサイズ（プロセッサの負荷とプロセッサ間通信のコストとのトレードオフ）、系のプロセッサへの割り当て法（プ

ロセッサの物理的な配置を考慮した負荷分散の方式）をパラメタとして設定し、自動最適化を行っている。また、実行時最適化を行う場合は、計算機の特性や性能をモニタリングする機能が必要である。

9.11.5 AT により得られる効用

番号	効用	本件の場合
E1	プログラムの実行時間が非常に長い、あるいは多くの計算機資源を必要とする場合	◎
E2	プログラムの実行時間、あるいは計算機資源に関する制約が非常に厳しい場合	○（精度を得るには、長時間実行が必要）
E3	パラメタ／アルゴリズムの選択により、性能（実行時間以外の目的関数も含む）が大きく変化する場合	○（大幅に変わる）
E4	プログラムが種々の計算機環境で使われる場合	○
E5	プログラムが長期間にわたって使われる場合	○

9.11.6 適用コストと予想される障壁

番号	効用（少ないコストでできる）	本件の場合
C1	計算（あるいは処理）の大部分が少数の計算カーネルに集中している場合	○
C2	パラメタを変化させることで、プログラムを種々の計算機環境に最適化することが可能な場合	○
C3	異なる計算機環境に適する種々のアルゴリズムを系統的に自動生成することが可能な場合	△（計算機に応じたアルゴリズム選択が可能だが、精度に影響）
C4	プログラム構造が明確でドキュメントが整備されている場合（すでに開発者がいなくなっているレガシコードなどでは、AT 化のコストが大きい）	○

9.11.7 AT 支援ツール

通常、シミュレーション結果の妥当性は、実験で得られた物理量が計算によって再現できていること、系のエネルギ値の総和が一定していること、シミュレーションによって得られた構造群と参照すべき構造との差が一定していること、溶媒分子の分布が既知の動径分布関数と一致していることなどによって総合的に判定される。こうした判定はある程度自動化できる可能性がある。判定の自動化のためには、これらの値を

モニタリングし、必要に応じてユーザに提示する機能が必要となる。従来ユーザが行ってきたチェックや判断を、自動的に行う手段が求められる。AT を支援する手段としては、目的関数の定義を容易化するツールや、モニタリングした結果を可視化するツールなどが考えられる。

9.11.8 AT 向きかどうかの検討

分子動力学法は、バイオインフォマティクスの他の手法に比べて、AT 導入の論点が比較的明確で、関連した研究例も見られる。大規模な系、精度の高い長時間のシミュレーションに対する要求は強い。地球シミュレータや京速コンピュータなど、各時代の最高性能の計算機において主要な応用の 1 つとして位置づけられ、AT の適用を検討する価値は十分にある。一方で、分子動力学法は研究室レベルのクラスタシステムや専用ボードでも実行され、さまざまな規模や形態の計算機環境で使われている。従って、計算機環境に応じた最適化を AT で行う効果は大きい。その基本的な手法は、材料系や天体シミュレーションなどにも適用されるので、こうした手法の汎用性も AT 向きであることの根拠としてあげられる。

9.11.9 今後の方向性

分子動力学法では、ユーザの要求に応じて、一定の精度を維持しながら、性能の向上を目指す必要がある。しかし、精度の具体的な尺度は必ずしも明確ではなく、さらには自動的なモニタリングとパラメタの最適化が難しく、現実的には人手の介在が必要となる可能性もある。従来、人手で行ってきたチェックをある程度自動的にモニタリングし、アルゴリズムの選択やパラメタの最適化に適用できることが望まれる。シミュレーションの対象や目的に応じて、今後、以下のような研究の方向が考えられる。

1. 力場パラメタや溶媒効果の計算法を自動的に選択し、性能チューニングする上位レベルの最適化
2. 計算機環境にあわせた性能チューニングを行う下位レベルの最適化
3. プログラムの動的特性に応じた実行時の最適化
4. folding@home などの広域分散システム環境の活用