アルゴリズム 32－7
（1993. 3. 18）

多種計算機上での適応的アルゴリズム選択法 (AASM)

須崎 有康　栗田 多喜夫　田沼 均　平野 聡

{suzaki,kurita,tanuma,hirano}@etl.go.jp

電子技術総合研究所

本論文では我々が提案した動的なソフトウェア最適化手法である「適応的アルゴリズム選択法 (Adaptive Algorithm Selection Method; AASM)」の異種計算機への適応実験について報告する。AASM は、ソフトウェアがライブラリを呼び出すとデータや計算機アーキテクチャに合わせて最適なアルゴリズムを選択し、処理する。最適アルゴリズムの判断は、事前に各アルゴリズムの性能測定をターゲット計算機で行ない、その結果に基づいてニューラルネットにより学習される。本論文では、複数文字列検索問題を対象とした実験を Sparc-Station 2、Cray X-MP/216、FACOM M 1800/30 で行なった。その結果、AASM が計算機ごとに適応が可能であることを確認した。

Adaptive Algorithm Selection Method (AASM) on various machines

Kuniyasu SUZAKI, Takio KURITA, Hitoshi TANUMA and Satoshi HIRANO

Electrotechnical Laboratory

1-1-1 Umezono,Tsukuba-city,Ibaraki,305,Japan

This paper presents the rerults of the experiments of the dynamic software tuning method, named Adaptive Algorithm Selection Method (AASM) on various machines. When the library is called, the AASM selects the algorithm which fits for the data and the machine from the algorithms registered in the library, and then makes it execute. Since the software is automatically tuned, the execution time will be shorten. We have experimented on multi-strings search problem with AASM on different architectures; Sparc-Station 2, Cray X-MP/216 and FACOM M 1800/30. This results shows that AASM is able to minimize the average execution time.

1 はじめに

ソートや文字列検索などのように一つの問題に対して複数のアルゴリズムが存在し、データの特徴によって最適アルゴリズムがことなる場合がしばしばある。最適なアルゴリズムの選択は、理論的計算量の考察だけでは難しく、データの特徴と最適なアルゴリズムとの関係を解析する必要がある。もし、各データに対して最適なアルゴリズムが選択できれば、処理時間は短縮されるであろう。図 1 では、データのサイズが小さい場合にはアルゴリズム A 、中くらいの場合には B、大きい場合には C が最適である。従ってこの場合データのサイズに応じてそれぞれのアルゴリズムを選択できれば、どんなデータに対しても効率的な処理が可能となる。本論文では、これをデータアダプテーションと呼ぶ(図 1)。

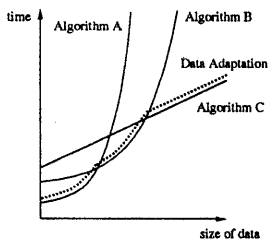

図 1: データへの適応

図 1 では、データの大きさと言う一つの基準で最適アルゴリズムが決められているが、一般的には複数の基準によって最適なアルゴリズムが決まると考えられる。例えば行列計算ではその大きさや密であるか疎であるかにより最適なアルゴリズムが異なる。データアダプテーションを実現するためには、このような複数のデータの性質によってアルゴリズムが選ばれなければならない。この選択を行なうためにはどのアルゴリズムを選択すべきかの選択基準「アルゴリズム選択基準 (Algorithm Selection Criterion)」を作る必要がある。

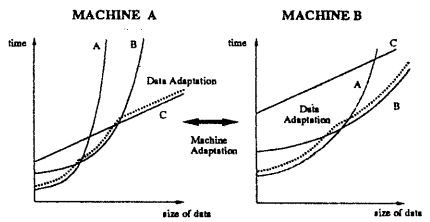

図 2: 計算機への適応

また、データの性質ばかりではなく実行する計算機のアーキテクチャによっても最適アルゴリズムは異なる。図 2 は二つの計算機で最適なアルゴリズムが異なることを示している。本論文では計算機アーキテクチャに適合して効率化を目指すことをマシンアダプテーションと呼ぶ。

我々が提案した AASM (Adaptive Algorithm Selection Method)[5] は、データアダプテーションとマシンアダプテーションを実現するための手法である。AASM では、用意されたアルゴリズムの性能測定を事前に行ない、データに対するアルゴリズム選択基準をニューラルネットによって学習する。学習したアルゴリズム選択基準をもとに、実行時にデータの性質に合わせて最適アルゴリズムを推論し、実行することで効率化を目指す。

AASM と同様に実行時にプログラムを変えて最適化を目指す研究は過去にプログラム変換 [1]、部分計算 [1]、リフレクション [2][3] などでなされてきた。しかし、これらはいずれもプログラムの変換手法であり、実際に最適化を行なうためにはそれ自身の処理時間の見積もりや最適化の判断基準が欠けていた。AASM はこれらの欠点を克服して、実用になる手法を目指す。

2 AASM

AASM はデータアダプテーションとマシンアダプテーションを同時に実現し、平均実行時間の最小化を行なう (図 3)。平均実行時間の最小化とは、あるデータで問題を解いたときに、その実行時間が一つのアルゴリズムのみで実行したときより AASM で実行した法がトータルで小さくなることである。AASM は、必ずオーバーヘッド時間 (AASM 自体の処理時間) を伴うが、最適なアルゴリズムを選択することで得る時間で相殺できるように構成する。

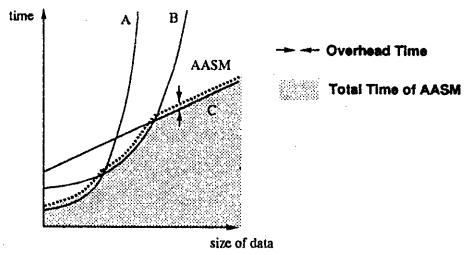

図 3: 最適化する時間

2.1 実現法

AASM はライブラリをインターフェースとしている。ライブラリには同一機能を実現するアルゴリズムが複数登録される。AASM は、ライブラリが呼ばれると起動し、引数データの特徴に合わせて登録してあるアルゴリズムのうちから最適なアルゴリズムを選び、実行する。実行の様子を図 4 に表す。

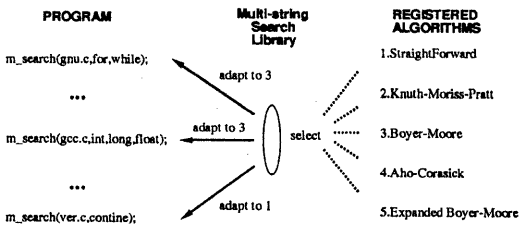

図 4: ライブラリ呼び出しによる AASM の実現

AASM は引数データから特徴量を抽出し、そのデータに最適なアルゴリズムを選択する。最適アルゴリズムの選択をデータ自体ではなく、データの特徴量に変換して行なうのは判別処理を簡約化し、AASM の計算時間を少なくするためである。

2.2 抽出する特徴

AASM でデータから抽出する特徴はアルゴリズムの実行時間に影響を及ぼすものでなければならない。実行時間に影響を及ぼさないものは、AASM の処理のオーバーヘッドになるだけである。

また、データから抽出する処理に時間がかかる特徴は採用できない。なぜなら、抽出することに時間がかかっては、オーバーヘッド時間が増加し、最適アルゴリズムを選択することで獲得した時間で相殺することができなくなるためである。性能測定データはこれらの特徴を網羅し、正しい学習を行なう手助けをしなければならない。

2.3 学習

性能測定から得られたデータに対してどのアルゴリズムが最適か判断するアルゴリズム選択基準をニューラルネット(エラーバックプロパゲーションアルゴリズム付き多層パーセプトロン)[4] を使って学習する。ニューラルネットを使用する理由は、学習の柔軟性が高いためと学習時に誤り率を変えることでアルゴリズム選択基準を微調節できるためである。ネットワークの構成は、高速化のために中間層を一層とする。このニューラルネットではデータから抽出された特徴の数の入力ユニットを用意し、登録されたアルゴリズムの数の出力ユニットを用意する (図 5)。学習パターンには、データと特徴を入力ユニットに与え、そのときに最適なアルゴリズムに対応する出力ユニットに 1、それ以外の出力ユニットは 0 とする。

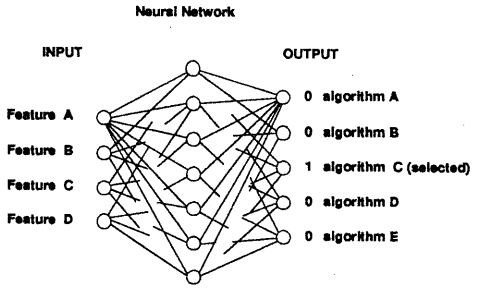

図 5: ニューラルネットの構成

3 多種計算機上での AASM 実験

異なる計算機上で AASM の性能がどのように変わるかを調べるために、複数文字列検索問題を対象に実験を行なった。複数文字列検索問題では、任意のファイル内の複数の文字列の出現個数を調べる。今回の実験ではマルチユーザ環境によって影響を及ぼされるディスクやネットワークの処理時間を含まぬようにメモリ上にファイルのデータを展開して検索を行なった。計算機は CRAY X-MP/216、FACOM M 1800/30、Sparc Station 2 を使った。また、ニューラルネットの学習効果を調べるために学習誤り率を変化させた実験も行なった。

3.1 実験条件

3.1.1 アルゴリズム

複数文字列検索のアルゴリズムは次に挙げる五つを利用した [6]。ここで示すアルゴリズムの計算量は検索文字列の長さを m 、ファイルの大きさを n として示している。

- Straightforwd アルゴリズム (SF)

  単一文字列検索アルゴリズム。文字列前方照合法で計算が失敗した場合でも検索位置を逐次に進める。単一文字列検索で最悪の場合、計算量は O(mn) である。

- Knuth-Morris-Pratt アルゴリズム (KMP)

  単一文字列検索アルゴリズム。文字列前方照合法で照合が途中で失敗した場合に検索位置を最も進められるアルゴリズム。前処理として検索位置をどれだけ進められるかの表を検索文字列から作成する必要がある。単一文字列検索の場合、計算量は O(n) である。

- Boyer-Moore アルゴリズム (BM)

  単一文字列検索アルゴリズム。文字列後方照合法で照合が途中で失敗した場合に検索位置を最も進められるアルゴリズム。前処理として検索位置をどれだけ進められるかの表を検索文字列から作成する必要がある。単一文字列検索の場合、計算量は O(n) である。

- Aho-Corasick アルゴリズム (AC)

  複数文字列検索アルゴリズム。検索文字列のパターンから有限オートマトンを作り、検索ファイルをこの有限オートマトンで走査する。

- 拡張 Boyer-Moore アルゴリズム (EBM)

  複数文字列検索アルゴリズム。Boyer-Moore の拡張版で複数の文字列後方照合法で照合が途中で失敗した場合に検索位置を最も進められるアルゴリズム。前処理として Boyer-Moore で作成した表に複数文字列で重複する部分を重ね合わせて作成する。

これらに対してアルゴリズム選択基準を作るためにデータから抽出する特徴は、ファイルの大きさ、検索文字列の数、個々の文字列長の平均、個々の文字列長の分散とした。これらはそれぞれアルゴリズムの実行時間に影響を及ぼすものである [6]。例を示すと次のようになる。

例: m_search(c-lang.c, int, continue, case)

ファイルサイズ	1803
文字列の数	3
文字列長の平均	5.00
文字列長の分散	2.65

3.1.2 性能測定

性能測定は、GNU の gcc のソースファイルに対する C の予約語 (int,continue,case など) の文字検索とした。ファイルは、0 ～ 10,000 バイト間で一様になる 20 個を選んだ。文字列は 1 ～ 10 個の組合せ 50 通りを試した。測定は 10000 パターンについて行なった。

3.1.3 ニューラルネットワーク

ニューラルネットワークの構成は、特徴が 4 種類のため入力ユニットを 4、登録したアルゴリズムが 5 種類のため出力ユニットを 4 とした。中間層のユニットの数は 7 とした。

入力する特徴は、それぞれの値の振幅 (最大値 − 最小値) がかなり異なるので、これらの値を正規化 (平均 0、分散 1) して入力した。これはニューラルネットが収束しなくなることを防ぐためである。

学習の効果については、学習誤り率を 0.1 と 0.05 まで収束させた二種類のアルゴリズム選択基準について比較した。

3.1.4 実行

AASM の効果を調べるデータとして、latex のソースファイルと latex のコマンド (begin,bibitem,chapter など) の文字列を使った。ここでも性能測定データと同様に 20 個のファイルと 50 通りの文字列の組合わせた 10000 パターンで測定を行なった。

AASM との性能比較のためにこれらのデータに対する各アルゴリズム単体の実行時間の測定もおこなった。

実行時間の測定には UNIX の標準ライブラリにある times 関数を使用した。times 関数の測定単位時間は計算機の OS によって異なり、次のとおりであった。

Machine	OS	測定単位
CRAY X/MP	UNICOS	1 / 117700000
FACOM 1800/30	UXP	1 / 100
SS2	SUNOS	1 / 60

SUNOS や UXP では、測定単位時間が短いので一回の実行では最適アルゴリズムの判断ができない。このため繰り返し回数は、どの測定単位時間でも計れるように 1000 回とした。

表 1: 各アルゴリズム単体と AASM のトータル時間 (sec): 最適な処理には ◇ でマークした。括弧の中の数値は選択されたアルゴリズム自体の処理のトータル時間。

	Machine	ALGORITHM					AASM	BEST
		SF	KMP	BM	AC	EBM		
Learning error rate 0.1	CRAY	10715.1	10946.2	3396.1	8991.2	◇ 2609.6	2610.5 (2534.1)	2445.3
	FACOM	3431.8	3561.7	1088.4	3835.7	1096.5	◇ 1071.5 (987.7)	970.8
	SS2	20943.0	21434.9	5750.9	18216.4	◇ 5262.2	5366.5 (4967.4)	4811.9
Learning error rate 0.05	CRAY	10713.5	10945.1	3395.4	8995.1	2608.7	◇ 2547.5 (2471.2)	2444.4
	FACOM	3431.6	3561.8	1088.3	3836.0	1096.9	◇ 1080.4 (996.7)	970.9
	SS2	20939.2	21427.6	5747.6	18208.6	◇ 5281.0	5373.7 (4977.2)	4810.1

表 2: 各アルゴリズム単体とと AASM のトータル時間比: 括弧の中の数値は選択されたアルゴリズム自体の処理のトータル時間。

	Machine	ALGORITHM					AASM
		SF	KMP	BM	AC	EBM	
Learning error rate 0.1	CRAY	4.382	4.476	1.389	3.677	1.067	1.068 (1.036)
	FACOM	3.535	3.669	1.121	3.951	1.129	1.104 (1.017)
	SS2	4.352	4.455	1.195	3.786	1.098	1.115 (1.032)
Learning error rate 0.05	CRAY	4.383	4.478	1.389	3.680	1.067	1.042 (1.011)
	FACOM	3.534	3.669	1.121	3.951	1.130	1.113 (1.027)
	SS2	4.353	4.455	1.195	3.785	1.098	1.117 (1.035)

表 4: AASM で選択されたアルゴリズムのランク

CRAY

Learning error rate 0.1

SELECT	1st	2nd	3rd	4th	5th	TOTAL
SF	0	0	0	0	0	0
KMP	0	0	0	0	0	0
BM	228	29	0	0	0	257
AC	0	0	0	0	0	0
EBM	602	130	1	10	0	743
TOTAL	830	159	1	10	0	1000

Correct Seclection:0.830

Learning error rate 0.05

SELECT	1st	2nd	3rd	4th	5th	TOTAL
SF	0	0	0	0	0	0
KMP	0	0	0	0	0	0
BM	324	27	0	0	0	351
AC	0	0	0	0	0	0
EBM	604	45	0	0	0	649
TOTAL	928	72	0	0	0	1000

Correct Seclection:0.928

FACOM

Learning error rate 0.1

SELECT	1st	2nd	3rd	4th	5th	TOTAL
SF	32	55	0	0	0	87
KMP	0	0	0	0	0	0
BM	570	83	0	0	0	653
AC	0	0	0	0	0	0
EBM	237	23	0	0	0	260
TOTAL	839	161	0	0	0	1000

Correct Seclection:0.839

Learning error rate 0.05

SELECT	1st	2nd	3rd	4th	5th	TOTAL
SF	25	75	0	0	0	100
KMP	0	0	0	0	0	0
BM	565	115	0	0	0	680
AC	0	0	0	0	0	0
EBM	205	15	0	0	0	220
TOTAL	795	205	0	0	0	1000

Correct Seclection:0.795

SS2

Learning error rate 0.1

SELECT	1st	2nd	3rd	4th	5th	TOTAL
SF	7	79	0	0	0	86
KMP	0	0	0	0	0	0
BM	384	55	0	0	0	439
AC	0	0	0	0	0	0
EBM	398	77	0	0	0	475
TOTAL	789	211	0	0	0	1000

Correct Seclection:0.789

Learning error rate 0.05

SELECT	1st	2nd	3rd	4th	5th	TOTAL
SF	10	87	0	0	0	97
KMP	0	0	0	0	0	0
BM	385	67	0	0	0	452
AC	0	0	0	0	0	0
EBM	385	66	0	0	0	451
TOTAL	780	220	0	0	0	1000

Correct Seclection:0.780

3.2 実験結果

複数文字列検索問題を AASM で解いた場合とアルゴリズム単体で解いた場合について比較した。

3.2.1 トータル時間の最小化

表 1 に計算機、学習誤り率の違いごとに各アルゴリズム単体及び AASM のトータル時間を示す。本来 AASM 以外は学習誤り率を変えても実行時間に影響を及ぼさないが、この表の値は測定の誤差のため若干の違いがある。この表から学習誤り率が 0.1 のときには AASM が最適になる場合が一回、0.05 のときには二回あることがわかる。これはニューラルネットの学習性能を高めることによって AASM の性能が良くなる例である。また、AASM が選択したアルゴリズムの実行時間を見るとすべての場合で最適となる。このことからアルゴリズム単体より計算機ごとにデータに対し最適なアルゴリズムを選択し、実行することで効率化が行なわれていることがわかる。

さらに表 1 のデータをもとに表 2 では、各実行時間をその条件での最適アルゴリズムの実行時間を 1 としたときの値に直した。この表より、AASM は最適アルゴリズムの実行時間の 1.12 倍以下であることがわかる。アルゴリズム単体で実行した場合はこの値が計算機によって越えることがある。

計算機間で比較するとアルゴリズム単体の実行時間の差が一定でないことがわかる。例えば、表 2 から学習誤り率 0.1 の場合の CRAY と FACOM を比較すると SF では 4.382 と 3.535 であるのに対し、EBM では 1.067 と 1.129 である。これは計算機によってアルゴリズムごとの実行時間比が異なることを示している。AASM では、このような状況に対して最適なアルゴリズムを選択している。

学習誤り率を比較すると CRAY と SS2 では学習誤り率を下げることによって実行時間が短くなっているが、FACOM では、逆に実行時間が長くなってしまった。FACOM の場合、選択されたアルゴリズムの実行時間が増加している。これは学習用データに対して過学習が起こっていることを意味する。この解析については、3.2.4 章で行なう。

3.2.2 オーバーヘッド時間

次に各計算間の AASM にかかるオーバーヘッド時間を調べた。表 3 に AASM 一回の処理にかかるオーバーヘッド時間を示す。この表では、学習誤り率が 0.1 のときの結果を示したが、学習誤り率が 0.05 になっても全体に対する割合が若干変わるだけである。

表 3: オーバーヘッド時間 (学習誤り率 0.1)

計算機	オーバーヘッド時間 (μ sec)	全体の割合 (%)
CRAY	75	3.1
UTS	84	8.6
SS2	399	8.3

この表より、CRAY と UTS では絶対的な処理時間は SS2 より高速であるが、全体の処理に対する割合では UTS は SS2 と同じであることがわかる。これはアーキテクチャを反映しているためである。AASM のオーバーヘッドは主にニューラルネットの処理時間であり、算術演算が高速である CRAY ではその特徴を反映してオーバーヘッド時間の割合が小さくなっている。

3.2.3 最適アルゴリズムとの実行時間差

次に最適アルゴリズムとの実行時間差について調べる。最適アルゴリズムとの実行時間差が 0 であることはそのアルゴリズムが最適であることを意味する。図 6 に学習誤り率 0.1 で FACOM で実行したときの結果をグラフにして示した。ページの都合で全計算機について載せることはできないが、傾向としては同じ結果が得られている。

図 6 より、AASM はアルゴリズム単体で実行した場合より最適アルゴリズムとの実行時間差が短い。これは一回の試行で AASM がたとえ選択ミスをしてもその影響は押されられていることを示している。また図 6 では、AASM のこの時間にオーバーヘッド時間を加えた時間場合でも、最悪で 0.5 ms であり、どのアルゴリズム単体より短かい。

3.2.4 学習の効率

AASM で使用したニューラルネットによる学習の効果を表 4 にまとめて示す。この表より正解率を比較すると、学習誤り率を 0.1 から 0.05 に上げると、正解率は CRAY と SS2 では上がり、UTS では落ちる。このことが 3.2.1 章で述べたトータル時間に影響を与えていると思われる。正解率が直接トータル時間に影響を与えると明言できない理由は、選択ミスしたときの最適アルゴリズムとの実行時間差がこの表からわからないためである。

選択される各アルゴリズムも計算機によって異なることが、表 4 から推測できる。特に CRAY 以外

の計算機ではStraightforwd アルゴリズムが選択されるのに CRAY では、Straightforwd アルゴリズムが選択されることがない。

学習誤り率0.1 の CRAY 以外のニューラルネットの選択ミスは常に二番目によいものとなっている。この表からは選択ミスしたときのロス時間の程度が示されないため、順位によって実行時間にどの程度の影響を与えるかわからないが、選択ミスを起こした場合は常に二番になっているため、選択ミスとしては最良であったことがわかる。

4 考察

今回の実験では、対象とした問題が CPU 性能だけに影響されるため計算機の特徴は固定とし、動的に変わる計算機 (OS) の性質までは捕らえなかった。しかし、AASM が動的に変わる計算機環境の特徴なども学習すれば、そうした状況にも適応できると思われる。たとえば、ネットワークの混み具合やメモリのスワップ頻度などが影響し、最適なアルゴリズムが変わるような場合にも有効であろう。こうした動的に変わる計算機環境についての検討は今後の課題である。

5 結論

本論文では、データや計算機に最適なアルゴリズムを動的に選び最適化を行なう AASM の計算機アーキテクチャへの適応を調べた。計算機には CRAY X-MP、FACOM M 1800/30、Sparc Station 2 を用いた。最適化する対象としては複数文字列検索問題を選び、Straightforwd 、Knuth-Morris-Pratt、Boyer-Moore、Aho-Crasick、拡張 Boyer-Moore アルゴリズムを用いて実験を行なった。この実験で AASM がそれぞれの計算機の特徴を捕らえ、計算機に最適なアルゴリズムを選択し、実行することで実行時間の短縮ができることが示された。

AASM は複数文字列検索に限らず、汎用な手法なのでどのような問題にも実装可能である。とくにアルゴリズムがデータの特徴によって変化する問題のように静的に最適なアルゴリズムを特定できなかったり、そのコストが大きいものについては AASM のような動的最適化は有効であると考える。また、アーキテクチャが複雑になる並列計算機にはこのような最適化手法が必要になると予想する。

謝辞

本研究の一部は RWC 計画の一環として「超並列システムアーキテクチャに関する研究」で行なわれたものである。関係各位に感謝いたします。特に貴重な意見を頂いた分散システム研究室の塚本亨治室長と浜崎陽一主任研究官に深謝したします。

参考文献

[1] 古川, 溝口 編: プログラム変換, 共立出版, (1987)

[2] P.Maes, D.Nardi: *Meta-Level Architectures and Reflection*, NORTH-HOLLAND (1988)

[3] A.Yonezawa, B.C.Smith ed.: *Proc. New Models for Software Architecture '92 Refelction and Meta-Level Architecture* (1992)

[4] D.E.Rumelhart, G.E.Hinton, R.J.Wiliams : *Learning internal representations by error propagation, in* Parallel Distributed Processing *Volume 1*, MIT Press (1986)

[5] 須崎, 栗田, 田沼, 平野: 適応的アルゴリズム選択法による動的最適化, 第34回プログラミングシンポジュウム報告集, pp.177-184 (1993)

[6] 尹, 高木, 牛島: 5種類のパターンマッチ手法をC言語の関数で実現する, 日経バイト, 1987-7, pp.175-191 (1987)

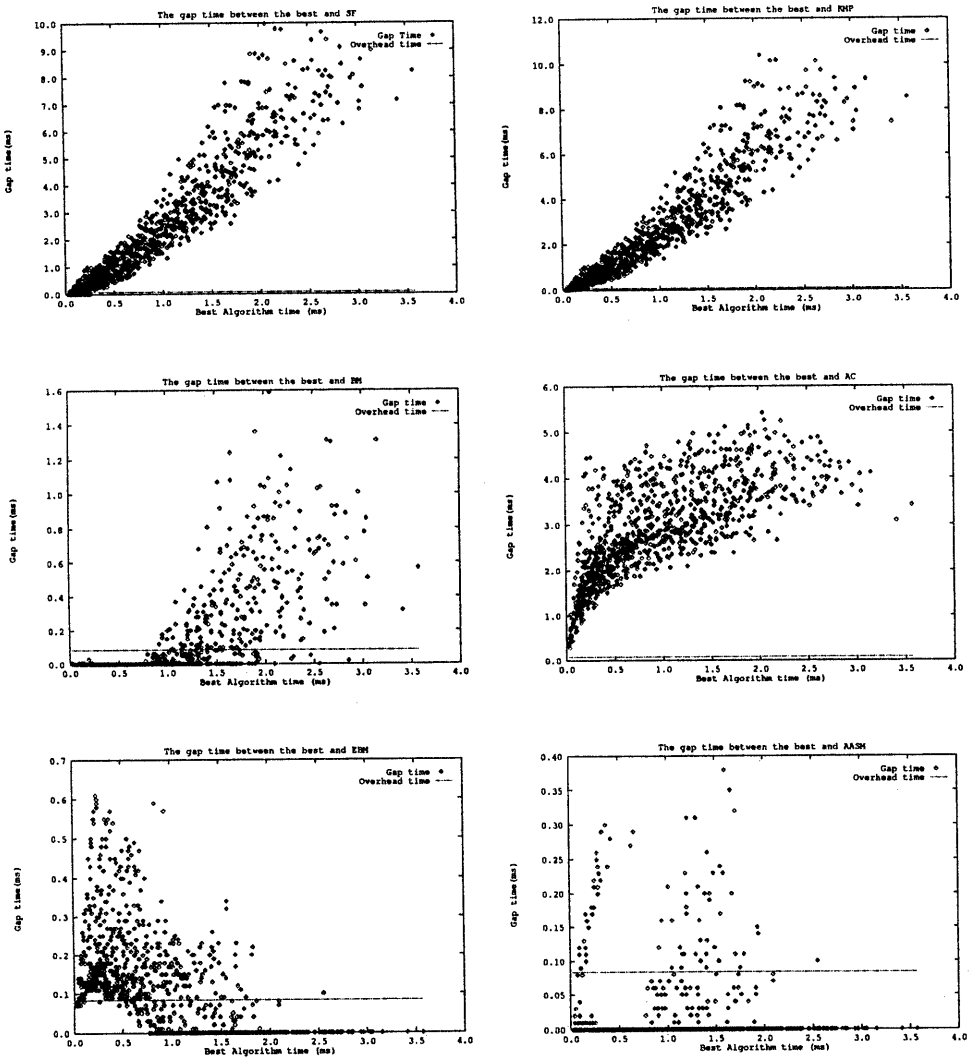

図 6: 最適アルゴリズムとの実行時間差: FACOM 1800/30 学習誤り率 0.1。横軸は最適アルゴリズムの実行時間。立軸は同じ問題を解いたときのそれぞれの実行時間差。