機械学習による命令スケジューリング規則の改良

THE REFINEMENT OF INSTRUCTION SEQUENCING RULES
USING MACHINE LEARNING METHOD

鈴木智史[1)]，梅谷征雄[2)]
Satoshi Suzuki and Yukio Umetani

1)静岡大学大学院　情報学研究科　情報学専攻（〒432-8011　静岡県浜松市中区城北3-5-1，gs07031@s.inf.shizuoka.ac.jp）
2)静岡大学　情報学部　情報科学科（〒432-8011　静岡県浜松市中区城北3-5-1，umetani@inf.shizuoka.ac.jp）

In the genetic instruction scheduler that applied the genetic algorithm to the instruction scheduling for pipeline processor, there was a problem that it requires high calculation cost though it achieves an excellent schedule. By applying the scheduling rules obtained by the genetic scheduler to the instruction scheduling, calculation cost was decreased. However, the effect of optimization was not better than that of the genetic instruction scheduler.We improved the scheduling rules for better optimization by defining new attributes which constitute scheduling rules.
Key Words; *scheduling, genetic optimization, rule optimization*

１．はじめに

近年，プロセッサのパイプライン制御方式が複雑化しておりプロセッサの性能を十分に引き出せていない可能性がある．複雑化するプロセッサに対応するため，遺伝的アルゴリズムを命令スケジュールに応用した遺伝的命令スケジューラ[1)]が提案された．静的な評価関数を用いる従来のコンパイル手法とは異なり，対象としたプログラムの実行時間のみに依存したスケジューリングをするため，そのプロセッサの最適化方法が分からなくても最適化できるという特長を持つ．この遺伝的命令スケジューラを Pentium プロセッサに適用した結果，実行時間の観点からは十分な効果が得られた[2)]．しかし，計算コストがかかりすぎるという問題があった．

遺伝的命令スケジューラによって最適化された命令列には何らかの規則性があると考えられる．計算コストの問題を解決するためにそういった規則を使用してスケジューリングを行うスケジューラが実装され，ルール学習型命令スケジューラ[3)]と名付けられた．ルール学習型命令スケジューラの小規模なプログラムに対しての効果の検証が行った結果，計算コストは大幅に削減できたが最適化の効果は遺伝的命令スケジューラには及ばなかった．

本研究では，ルール学習型命令スケジューラの最適化の効果を高めるため，スケジューリング規則の改良を行った．

２．遺伝的命令スケジューラ

遺伝的アルゴリズムとは動物の進化を模倣した学習的アルゴリズムであり，それを命令スケジュールに応用したものが遺伝的命令スケジューラである．与えられた命令列の並び順を入れ替えて，実行時間の性能を評価し，性能のよい命令列を残す．これを繰り返すことによって最適な命令列を生成する．遺伝的命令スケジューラのフローチャートを図 1 に示す．

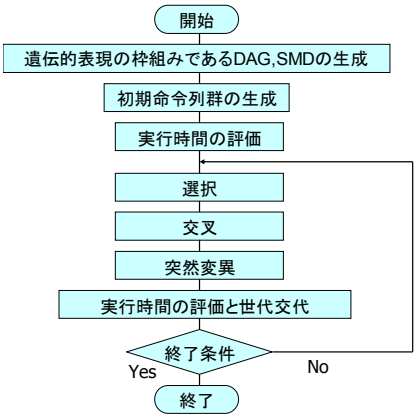

図１　遺伝的命令スケジューラ

３．ルール学習型命令スケジューラ

ルール学習型命令スケジューラとは，遺伝的命令スケジューラの結果から得られた規則を使用してスケジューリングを行うスケジューラである．ルール学習型命令スケジューラのフローチャートを図 2 に示す．

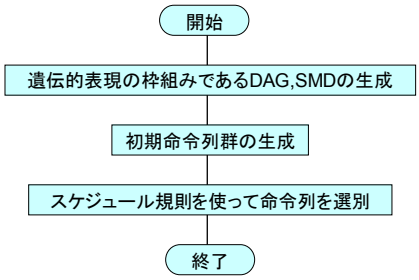

図２　ルール学習型命令スケジューラ

４．スケジュール規則

ルール学習型命令スケジューラにおいて，最適命令列を選別するのに使用するのがスケジュール規則である．

４．１　属性値

アセンブラコード中の以下の項目がプログラムの実行時間に影響を与えることが先行研究[4]で分かっている．

・平均依存距離
・オペランドの依存
・ストア命令の連続性
・浮動小数点命令のあとに整数命令
・浮動小数点命令のあとの整数命令の連続性
・ロード命令の連続性
・浮動小数点命令の連続性
・浮動小数点命令の平均連続性
・連続ストア命令間のアドレスの距離

これらの項目を属性と呼び，属性を数値化したものを属性値と呼ぶ．

４．２　スケジュール規則の抽出

遺伝的命令スケジューラでは命令列の並び替え，性能の評価が繰り返し行われるため，大量の並びの異なる命令列が生成される．それぞれの命令列の属性値を計算し，実行時間とともにデータセットと呼ばれる形式にまとめる．このデータセットをデータマイニングツール WEKA[5] に入力し，スケジュール規則の抽出を行う．

４．３　スケジュール規則の例

表 3 にスケジュール規則の一例を示す． 規則抽出の対象にしたプログラムは，LFK ベンチマーク[6]と呼ばれる FORTRAN で書かれた 24 個のプログラムを C に書き直したものである．

表１　カーネルプログラムのスケジュール規則例

rule	平均依存距離	オペランドの依存	ロードの連続	floatの平均
1	<=1.65 > 1.42	<=0.40		>1.14
2		>0.49		<=1.63 >1.5
3		>0.46	<=0.5	<=0.88
4		>0.46		
5	<=2.40	>0.46		<=0.85 >0.5
6		>0.46	>0.43	<=0.33
7	<=1.85	>0.45	>0.1 <=0.18	<=1.66
8	>3.11			
9	>2.21	<=0.5 >0.45		>1.33 <=1.85
10	<=2.03	<=0.37	>0.1 <=0.36	<=1.25
11		<=0.22 >0.18	<=0.20	
12			<=0.46 >0.3	

例えばある命令列において，平均依存距離が 1.42 より大きく 1.65 以下かつオペランドの依存が 0.40 以下かつ float の平均が 1.14 より大きければ，その命令列はルール 1 を満たすということになる．

５．新しい属性の追加

ルール学習型命令スケジューラの改良のため，新しい属性について検討した．性能の良い命令列と悪い命令列を比較し，性能の良いものの命令の並びにはどういった傾向があるかを調べ，それを新しい属性として追加した．命令列の比較に使用したプログラムは LFK ベンチマークである．

５．１　fld 命令の連続性

LFK ベンチマークの一つ，kernel19 に関して性能の異なる命令列の比較を行ったところ，fld 命令の連続性が性能に大きく影響を与えている可能性が高いことがわかった．図 3 に命令列比較の一例を示す．

早い命令列(実行時間:725ms)

fld	qword ptr [esp+4]
fld	qword ptr [ebx+8*eax]
fstp	qword ptr [esp+4]
fstp	qword ptr [ebx+8*esi]

遅い命令列(実行時間:1022ms)

fld	qword ptr [esp+4]
fstp	qword ptr [ebx+8*esi]
fld	qword ptr [ebx+8*eax]
fstp	qword ptr [esp+4]

図３　命令列の比較（kernel19）

fld の連続性についての属性を作成し，fld の連続性と命令列の実行時間との関係を調べた．図 4 のグラフから，fld の連続性が kernel19 の命令列の性能に大きな影響を与えていることがわかる．

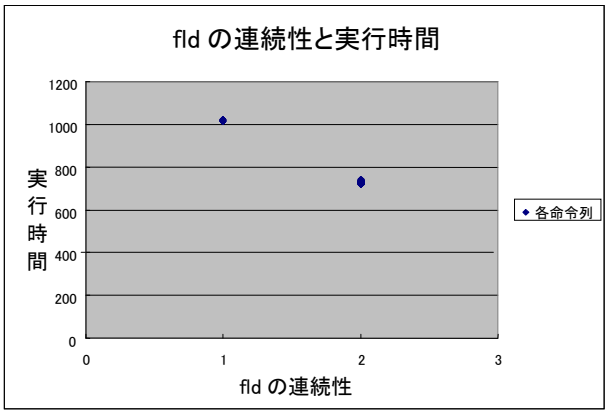

図４　fld の連続性と実行時間

５．２　fmul 命令の連続性

表２，３から分かるように kernel20 では性能の良い命令列には fxch 命令の挿入数が多い．

表２　fxch 挿入数（良）

命令列番号	fxch挿入数
79	4
49	2
80	4
50	3
55	2

表３　fxch 挿入数（悪）

命令列番号	fxch挿入数
68	0
23	0
65	2
92	0
74	1

このことから kernel20 ではフロート命令の並びが性能に影響を与えていると考えられる．フロート命令の並びについて焦点を絞って命令列の比較を行ったところ，性能の良い命令列では fmul 命令が連続する傾向が見られた．そこで fmul の連続性に関する属性を作成し，実行時間との関係を調べた．結果を図５に示す．

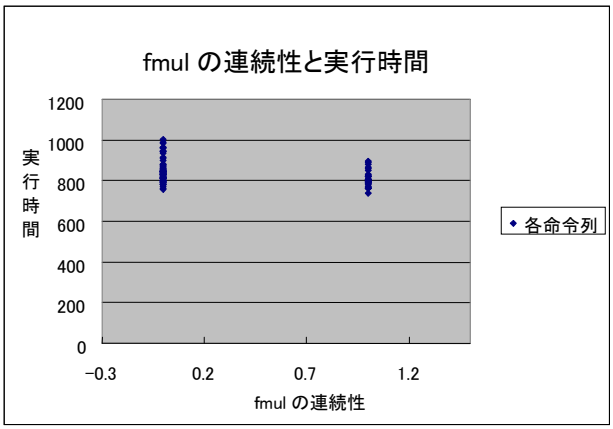

図５　fmul の連続性と実行時間

５．３　inc 命令の位置

命令列を比較した結果，kernel1 において inc 命令の位置が性能に大きく影響を与えていると考えられる．inc 命令の位置に関する属性を作成し，実行時間との関係を調べた．結果を図６に示す．

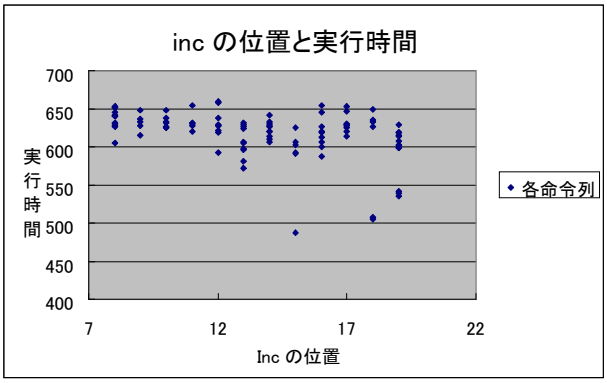

図６　inc の位置と実行時間

５．４　add 命令の位置

kernel24 に関して命令列の比較を行った結果，add 命令の位置が性能に影響を与えている可能性が考えられる．add 命令の位置に関する属性を作成し，実行時間との関係を調べた．結果を図７に示す．

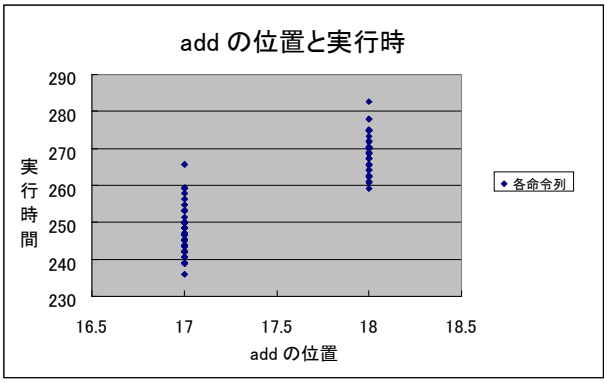

図７　add の位置と実行時間

６．結果

新しい属性を使用して，スケジュール規則の抽出を行った．新しく得られたスケジュール規則をルール学習型命令スケジューラに組み込み，最適化の効果を調べた．

６．１　新規則の抽出

既存の１０個の属性に新たな属性４個を加えて規則の抽出を行った．これを規則Ｂとする．規則Ｂの一部を表４に示す．

表４　規則Ｂ

rule	平均依存距離	オペランドの依存	ロードの連続
1	1.44 < <= 1.84		
2	1.44 < <= 1.84	<= 0.205	<= 0.16
3	1.44 < <= 1.84	0.13 < <= 0.205	
4	<= 1.84	0.087 < <= 0.343	0.196 < <= 0.353
5	<= 1.84	<= 0.344	0.196 < <= 0.4
rule	floatの連続	incの位置	addの位置
1	<=1	<=26	
2	> 1	<= 25	
3	> 1	> 25	
4	<= 3	> 26	<= 54
5		> 26	54 < <= 82

新たな属性４個のみを使用して規則の抽出を行った．これを規則Ｃとする．規則Ｃの一部を表５に示す．

表５　規則Ｃ

rule	fmulの連続性	incの位置	addの位置	fldの連続性
1		22 < <= 26	22 < <= 23	
2		30 < <= 36	23 < <= 27	<= 1
3		<= 36	27 < <= 54	<= 1
4		<= 31	51 < <= 54	> 1
5		36 < <= 38	50 < <= 54	

また，既存の１０個の属性を使用して得られた規則を規則Ａとする．規則Ａは表１に示したものである．

６．２　実行結果

規則Ａ，Ｂ，Ｃをそれぞれルール学習型命令スケジューラに組み込み，最適化の効果を調べた．ルール学習型命令スケジューラは初期命令列群の生成の段階で１０個の命令列を生成する．これを１０回繰り返し，１００個の命令列を生成した．それらを規則にマッチしたものしなかったものに分け，それぞれの命令列の実行時間の平均値をとった．スケジューリングの対象とするプログラムは LFK ベンチマークから９つと，SPEC ベンチマークから３つである．LFK ベンチマークへの結果は表６から１４に，SPEC ベンチマークへの結果は表１５から１７に示す．　色が塗られている規則は，「規則にマッチ」の値が最も良かったものである．

表６　kernel1 の結果

規則	規則にマッチ	マッチせず	マッチ数
A	638.1	617.3	2/100
B	615.2	618.6	26/100
C	624.7	615.9	20/100

表７　kernel2 の結果

規則	規則にマッチ	マッチせず	マッチ数
A	306.2	305.5	13/100
B	307.7	304.7	30/100
C	317.9	300.3	30/100

表８　kernel3 の結果

規則	規則にマッチ	マッチせず	マッチ数
A	4877.8	4870.1	31/100
B	4865	4875.9	31/100
C	4873.9	4870.1	62/100

表９　kernel4 の結果

規則	規則にマッチ	マッチせず	マッチ数
A	674.8	—	100/100
B	673.8	675.2	33/100
C	674.4	674.8	21/100

表１０　kernel12 の結果

規則	規則にマッチ	マッチせず	マッチ数
A	130.9	133.6	91/100
B	129.7	132.7	52/100
C	132.9	130	40/100

表１１　kernel15 の結果

規則	規則にマッチ	マッチせず	マッチ数
A	807	—	100/100
B	807.1	807	23/100
C	808.3	806.7	21/100

表１２　kernel19 の結果

規則	規則にマッチ	マッチせず	マッチ数
A	776	787.7	6/100
B	790.3	785.4	33/100
C	729.5	816.6	34/100

表１３　kernel20 の結果

規則	規則にマッチ	マッチせず	マッチ数
A	835.3	837.7	72/100
B	830.1	840.1	41/100
C	—	836	0/100

表１４　kernel24 の結果

規則	規則にマッチ	マッチせず	マッチ数
A	258.3	—	100/100
B	255	259.8	31/100
C	—	258.3	0/100

表１５　bzip の結果

規則	規則にマッチ	マッチせず	マッチ数
A	8100.7	8098.2	90/100
B	8040.6	8099	1/100
C	—	8098.5	0/100

表１６　twolf(1)の結果

規則	規則にマッチ	マッチせず	マッチ数
A	425.4	423.1	73/100
B	—	424.8	0/100
C	424.8	—	100/100

表１７　twolf(2)の結果

規則	規則にマッチ	マッチせず	マッチ数
A	432.1	—	100/100
B	432.1	—	100/100
C	422.4	434.6	21/100

７．まとめ

９つの LFK ベンチマークのうち，６つで規則Ｂが最良の結果を示した．既存の属性に新たな属性を加えることでスケジュール規則の改良が成功した．

また kernel19 に関しては，規則Ｃが一番良い結果を出している．これは属性の一つである fld の連続性が大きく影響しているためと考えられる．kernel19 に対しては fld の連続性は非常に有効だが，他のカーネルプログラムには効果がない．これは他のプログラムでは fld の連続性が一定の値をとり，変化することがないためである．

LFK ベンチマークには効果のあった新属性だが，SPEC ベンチマークに対しては，規則のマッチ数が極端に多かったり，マッチしなかったりと規則による選別がうまく行われないものが多かった．これは今回使用した規則が LFK ベンチマークに特化したものだったためと考えられる．

今後の課題として，SPEC ベンチマークの命令比較による新しい属性の検討が挙げられる．SPEC ベンチマークに関しては，LFK ベンチマークを対象にしたときよりも遺伝的命令スケジューラの最適化の効果が薄いことが分かっている．ルール学習型スケジューラを改良する前に，遺伝的命令スケジューラの改良を検討する必要があるかもしれない．

参考文献

1) 伊東勇，梅谷征雄：遺伝的アルゴリズムを用いる命令スケジューリング方式とその効果，情報処理学会論文誌 Vol.41 No.4,pp.1073-1085,2000

2) 長倉裕叔，梅谷征雄：ＰＣアーキテクチャにおける遺伝的命令スケジューリングの適用実験，情報処理学会研究報告 Vol.2000,No.57,pp.31-36,2000

3) 4) 川上肇，梅谷征雄：自動学習による高性能プロセッサ向け命令スケジュール規則の抽出とその実装，情報処理学会研究報告，2006-ARC-166,2006

5) Weka Machine Learning Project, http://www.sc.waikato.ac.nz/ml

6) McMahon，F.，The Livermore FORTRAN Kernels:A computer test of the numerical performance range, Technical Report，Lawrence Livermore National Laboratory,1986.