C プログラムの診断と重み付け採点を行う学習支援システム

大和雄一郎† 吉永泰甫† 竹内章††

本研究の目的は，学習者がプログラミング演習で作成した C プログラムを自動的に診断し，採点することである．診断では，プログラムの処理をノード，データの参照関係をデータフローとしたグラフにプログラムを変換する．そして正解と学習者のグラフを比較し類似度を計算する．採点では診断結果を利用してデータの参照関係をもとに点数計算を行う．さらに今回，アルゴリズムを意味的なまとまりとし，それらの重要度に応じた重み付きの採点を行えるようにした．最後に，システムの採点結果の妥当性を調べるため，教師との採点結果を比較したところ相関係数が 0.7～0.8 であった．

Learning Support System for Diagnosis and Weighted Grade of C Programs

Yuichiro Yamato†, Taisuke Yoshinaga† and Akira Takeuchi††

This paper proposes a method of automatically diagnosing and weighted grading learner's C programs. In the diagnosing step, programs are converted into graphs where nodes represent processes and links represent reference relations of data in the programs. Then, learner's graph is compared with correct graphs corresponding to correct programs, and scored with degree of similarity to find out the most similar one. In the grading step, similarity between the learner's graph and the most similar correct graph worked out, and then the grade of the program is calculated in accordance with the similarity and weights assigned to components of algorithm. We evaluated our method by comparing the grades and teacher's scores. The result shows the correlation coefficient is 0.7 – 0.8.

1. はじめに

プログラミングの初心者が講義で出題されるプログラム課題に取り組む場合，その課題に沿ったプログラムを初めから作成できることは少ない．そのため試行錯誤を繰り返して間違い箇所を見つけたり,他者からアドバイスを受けたりして課題をこなす．このとき，教師は学習者のプログラムを見て，そのプログラムがどのような動きをするのか判別し助言を行うが，時間的な制約や人員的な制約から，教師が全ての学習者のプログラムを見ることは難しい．そのため，学習者プログラムを自動で診断し，フィードバックを与えるシステムが多く研究されている．

プログラムの診断方法は記述そのものを診断する静的な診断と，テストデータを与えその出力内容から診断する動的な診断があり，プログラミングの学習支援では両方が用いられている．静的な診断で学習支援を行う研究[1]では，正解プログラムのソースコードを準備し，自動でソースコードを穴埋め式の問題とし，学習者が解を入力するシステムである．これにより，プログラム中の間違い箇所の把握が行え，採点が比較的容易になる一方で，通常のプログラミング演習に比べて学習者は自由にプログラムを作成できない．動的な診断で学習支援を行う研究[2]では，学習者プログラムのコンパイル後，いくつかのテストデータを与え，その出力から正誤判定を行っている．

動的な診断では，比較的容易にプログラムを診断できる反面，主に正誤判定しか行えない場合が多く，部分ごとの評価ができず，間違い箇所の指摘を行うこともできない．一方，静的な診断では，学習者の演習中に作成するプログラムが多様であるために診断が容易ではないが，記述されたプログラムを診断するため，部分ごとに採点することができ，間違い箇所の指摘を行うことが出来る．そこで我々は教師の採点の支援と,間違い箇所指摘による学習者の支援を行うために静的な診断を行うことにした．

本研究ではプログラム中の意味的なまとまりごとに重要度に従った重みをつけ，採点することで，教師の採点支援と学習者のプログラム修正支援を行う．我々はこれまでに，データの参照関係に着目した診断システム[3]の研究を行っている．システムが返す採点結果が教師の行う採点と近いものであれば教師の採点支援を行えるが，先行研究では正解と学習者のプログラムの機械的なマッチングしかしておらず，プログラム構造の意味的なまとまりを認識し重みを付けて採点することができていない．これは先行研究が,重み付けを行うための意味的なまとまりを認識できてないためである．よって本研究では，学習者プログラムに対して，プログラムの構造ごとに認識し重みを付け，その重みに従った採点をできるようにすることを目的とする．

† 九州工業大学大学院情報工学府
Graduate School of Computer Science and Systems Engineering, Kyushu Institute of Technology
†† 九州工業大学大学院情報工学研究院
Faculty of Computer Science and Systems Engineering, Kyushu Institute of Technology

2. プログラム診断と採点の概要

本研究の対象は，プログラムの概念や文法などの学習を終え，アルゴリズムについて学ぶプログラミング演習であり，システムは与えられた課題に対して学習者が作成したプログラムを入力として受け取り，その評価結果を出力する．アルゴリズムは意味的なまとまりを持って構成されているため，この意味的なまとまりごとに評価することでアルゴリズムが正しく構築できているか確認する．

診断と採点の概要を図 1 に示す．評価対象の学習者プログラムを受け取ると，演習の正解プログラムと比較診断を行い，その診断結果を用いて採点する．採点は，学習項目の重要度に従って教師が正解プログラムの意味的なまとまりごとに与えた重みを参照し，その重みに従って行う．最後の採点結果出力では，プログラム全体の点数だけでなく，意味的なまとまりやその構成要素ごとの部分点も出力する．

出力される点数により教師には採点を，また学習者には間違い箇所の把握を支援できると考えている．

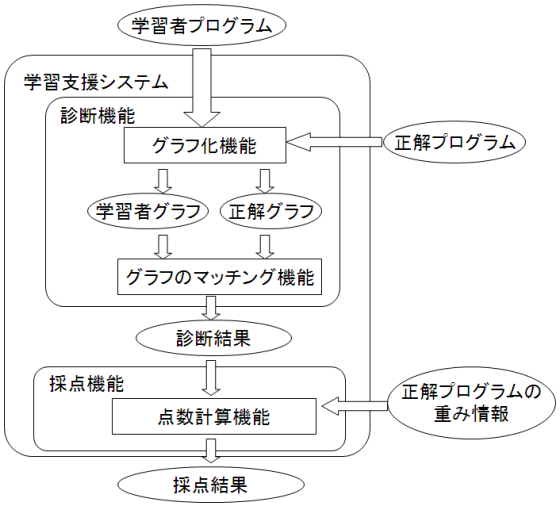


図 1：診断と採点処理の概要

3. 診断機能

学習者プログラムを静的に診断する多くの研究[4][5][6]では，制御構造を中心に正解プログラムとの対応を取っている．これに対して，我々は正解と学習者のプログラムを，データフローを表現するグラフに変換し，対応を取っている．このようにすることで，正解と制御構造が異なる学習者プログラムであっても，データの依存関係が類似している部分を検出でき，学習者の意図をくみ取りながら正解へと導く支援ができると考えている．

3.1 グラフ化とマッチング

プログラムをグラフ化したものは，プログラム中の処理を表すノードと，ノード間のデータ参照関係を表すデータフローで構成される．プログラム中の処理は代入，条件分岐，ループ，ブレイク，リターン，関数定義，関数呼出の 7 種類がある．図 2 のプログラムをグラフに変換した例を図 3 に示す．

正解と学習者のプログラムをグラフへ変換し，それぞれのノードと変数の対応づけを行う．ノードは処理の種類やノード内で用いる変数のデータフローなどから加減点し類似度を計算した後，最も高いものと 1 対 1 で対応づける．変数は型やデータフローなどから加減点し類似度を計算したあと，1 対 1 もしくは，1 対多で対応づける．

```
int ListScan(list_t *ls){
    list_t *p;
    for(p=ls;p!=null;p=p->next){
        if(p->value == 3)
            return 1;
    }
    return 0;
}
```

図 2：グラフ化のプログラム例

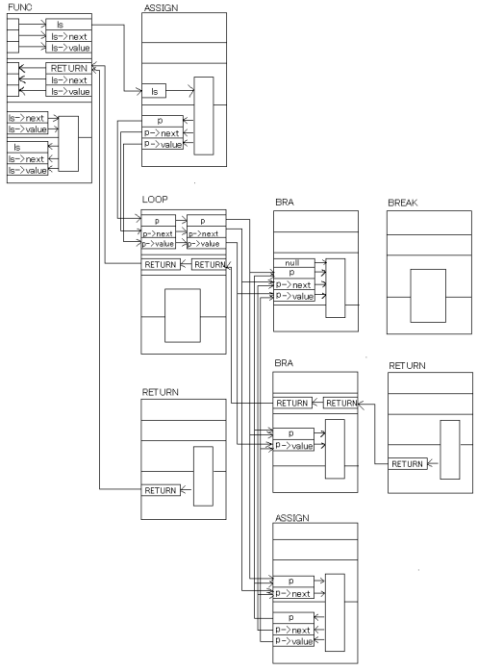


図 3：グラフ変換例

3.2 グラフの標準化

プログラムをグラフに変換する際には，記述の違いが問題とならないようにグラフの標準化を行っている．記述の違いとは for 文と while 文のように処理が同じでも異なる記述がされることであり，標準化を行わない場合は両方の正解を用意しなければならない．そこで標準化を行えば正解を用意する数が減り，教師の負担を減らすことができる．

本研究では，先行研究に加え作業用変数の対応を行うことでグラフの標準化機能を改良した．一般的に作業用変数とはソートアルゴリズムなどで使われる一時的に値を格納するために必要となる変数である．しかし本研究では作業用変数を，一時的に値を格納しておく本来なくてもよい変数と定義し，グラフ中の作業用変数の削除，グラフの修正を行った．図 4 は作業用変数のグラフ標準化例である．

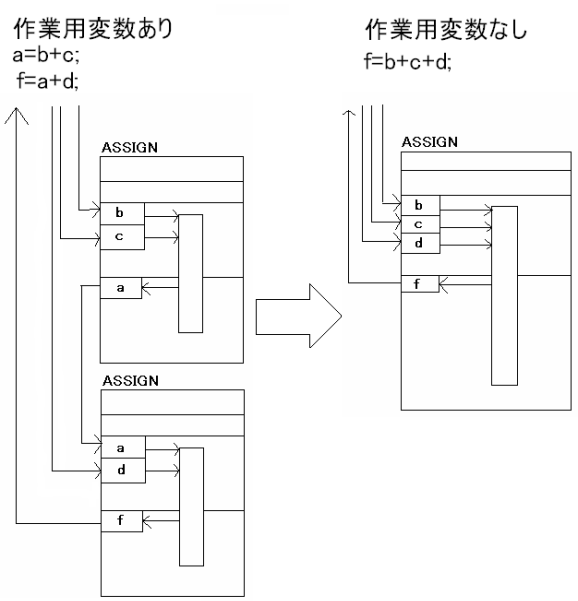


図 4：作業用変数のグラフ標準化

4. 採点機能

採点機能は，教師の採点および学習者の間違い箇所把握の支援を目的としている．教師は採点時にアルゴリズムの重要度に応じて点数を与えている．3.1 節で述べた類似度は，学習者プログラムと正解プログラムとの一致の程度を表してはいるが，プログラムの部分ごとの重要度を考慮していない．そこで，アルゴリズムを構成する意味的なまとまりのある部分ごとに重要度を与えて採点することにした．

本研究では意味的なまとまりを扱うため，以下二つのブロックを定義した．

・基本ブロック
　プログラムの一文をより基本的な処理に分割したもの．グラフのノードに対応し，採点の基本単位となる．

・意味ブロック
　プログラムの処理が複数集まってひとつの意味を構成するもの．意味的なまとまりがこのブロックに対応し，基本ブロックや他の意味ブロックから構成される．

プログラムをブロックにより木構造で表すと,葉が基本ブロックとなる.(図5参照)
事前に，教師には正解プログラム中の基本ブロックを用いて意味ブロックの設定を行い,意味ブロックを構成する各ブロックに重み付けを行ってもらう.(図5参照)
システムでは診断結果を用いて，基本ブロックの採点を行う．その後，基本ブロックの点数と重みを利用して意味ブロックの採点を行う．

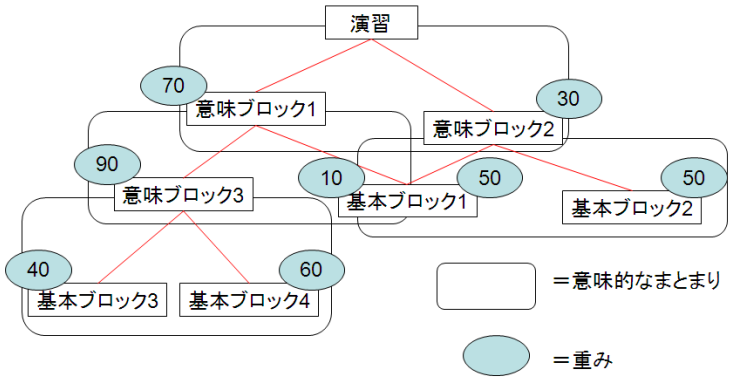


図 5：プログラムの木構造の例

4.1 基本ブロックの採点

学習者プログラム中の基本ブロックのもつデータ参照関係の正しさに応じて加点する．データ参照関係の正しさとは，入出力の変数とデータの入出力先の基本ブロックが正解と対応しているものを正解とし，どちらか一方でも対応していない場合は間違いとする．採点を行う条件を次に示す．

4.1.1 正解と対応している場合

採点する基本ブロックが正解と対応している場合，データフローごとにどの基本ブロックからデータを受け取りどの基本ブロックへデータを渡しているのかを正解との対応関係から診断し，正しいときに加点する．図 6 は採点例として基本ブロック「c=a+y」の採点を行っている．図 6 では対応している変数をわかりやすくするため同じ変数名を用いているが、本来違う変数名であっても問題はない．基本ブロックの対応は配置している場所で表している．採点したい基本ブロックは，入力で y を受け取っており、正解と対応していないデータを受け取っている.その結果，3 つあるデータフローのうちデータ参照関係が正しいものが 2 つなので 66.66 点となる．

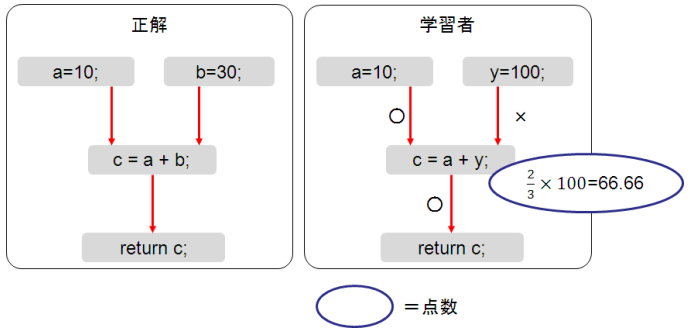


図 6：基本ブロックが正解と対応する場合の採点例

4.1.2 正解と対応しない場合

採点する基本ブロックが正解と対応しない場合，この余分なブロックは正解には必要がない処理のため採点対象としない．もしこの基本ブロックが正解に必要なデータ参照関係に影響を及ぼしていれば，他の基本ブロックのデータフローが正解と異なり，その基本ブロックの点数が下がることで，余分なブロックの影響があらわれる．図 7 では正解と対応しない基本ブロックが他の基本ブロックのデータ参照関係に影響を及ぼしている例である．図 6 と同様に基本ブロックの配置と変数名により対応関係を表している．基本ブロック「b=a+30」の入力変数は対応しているが，受け取り先が正解と対応していないため，参照関係が正しくない．その結果，2 つあるデータフローのうち正しいものが 1 つなので採点結果が 50 点となっている．

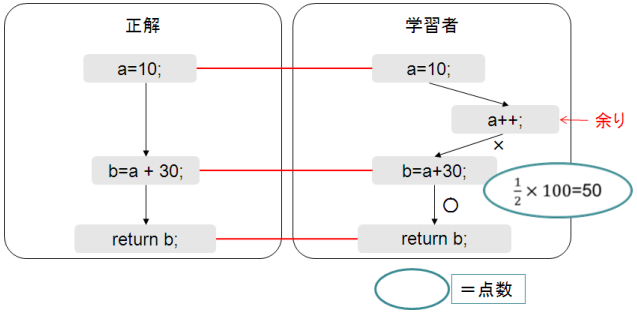
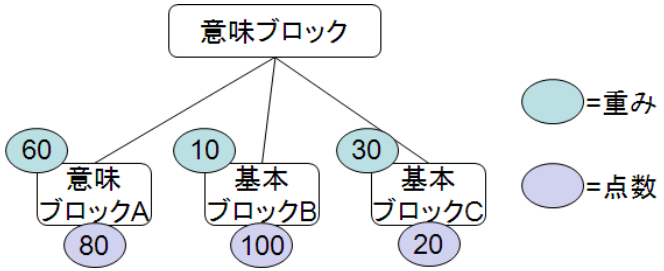


図 7：基本ブロックが正解と対応しない場合の採点例

4.2 意味ブロックの採点

意味ブロックを構成する基本ブロックの点数を重みに換算して採点を行う．図 8 は採点例である．



$$80 \times \frac{60}{100} + 100 \times \frac{10}{100} + 20 \times \frac{30}{100} = 64$$

図 8：意味ブロックの採点例

5. 評価実験

5.1 目的と評価方法

システムの採点結果の適切性を評価するために，教師の採点結果との比較実験を行った．本評価実験のプログラムには，2010 年度に本学の知能情報工学科で実施された講義「データ構造とアルゴリズム」の筆記試験の結果を用いた．問題は，リングバッファを用いたキューの操作関数の作成であり，4 つの関数を作成するもので各 4 点満点である．課題内容を表 1 に示す．

表 1：試験で作成された関数

関数名	配点	内容
step	4	キュー配列の添え字を一つ進める
enqueue	4	キューにデータを追加する
dequeue	4	キューからデータを取得する
findQueue	4	ある値がキューに入っているか確かめる

評価は教師の採点とシステムの採点を比較し相関係数を求めた．4 点満点では散布図から相関を見ることが難しいため，4 つの関数の合計点での相関係数も求めた．

5.2 評価結果

比較結果を図 9，10 に示す．教師とシステムの採点が一致している点をわかりやすくするために直線を引いている．また，■の点は 5 件以上だったものを表し，●の点は 4 件以下だったものを表している．それぞれの相関係数を表に示す．表 2 内の（）は教師とシステムともに満点と判断したプログラムを除いた値である．

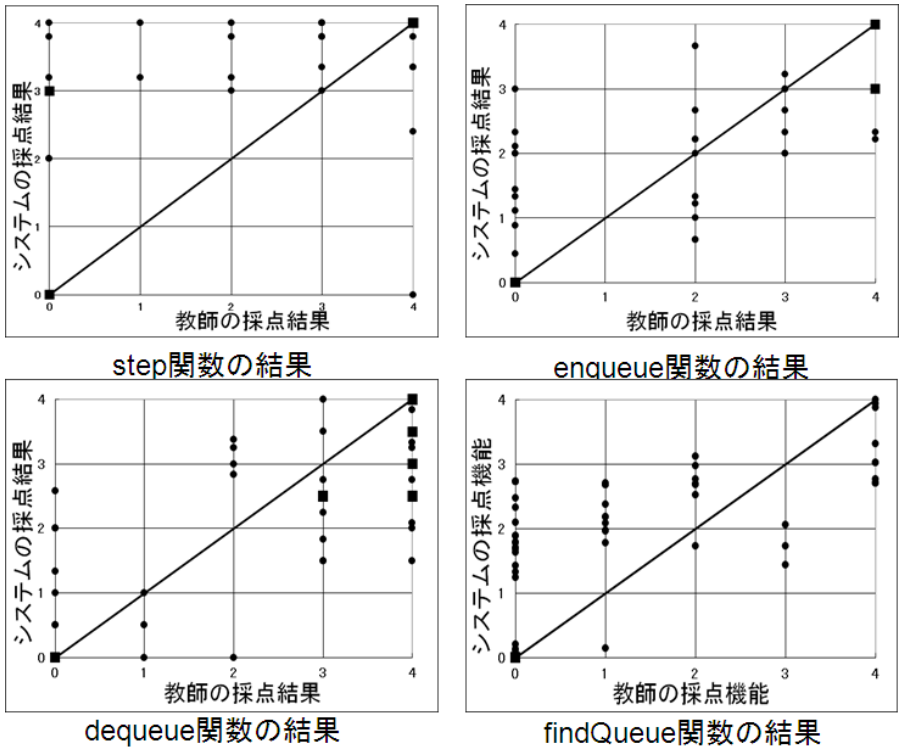


図 9：各関数の実験結果

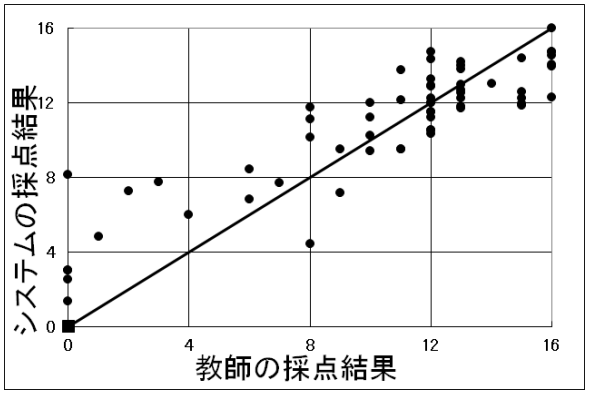


図 10：合計点の実験結果

表 2：実験結果の相関係数

関数名	対象数	相関係数
step	99（38）	0.719(0.465)
enqueue	95（64）	0.848(0.801)
dequeue	98（80）	0.849(0.831)
findQueue	87（86）	0.699(0.681)
合計点	70（69）	0.923(0.921)

```
int step(int idx) {
   return (idx + 1) % QUEUESIZE;
}
```

※QUEUESIZE：キューのサイズ

図 11：step 関数の正解例

6. 考察

表に示したように，相関係数はほぼ 0.7～0.9 となり，全体的に良い一致を示している．しかし，step 関数の満点プログラムを除いた相関係数は少し低くなり，また個々のプログラムでは教師とシステムの採点結果が離れているものがいくつかあった．その理由としては次のものが挙げられる．

(1) 構文間違いの修正
本システムでは構文間違いのあるプログラムの診断を行うことができないため，学習者プログラムを入力するときに“;”忘れや for 文内の記述順の間違いなどケアレスミスについては修正を行っている．そのため，教師が構文間違いとして減点している場合，システムでは減点されていないためシステムの点数が高くなる傾向がある．

(2) 必要不可欠なブロックによる重み
作成した採点機能では，ブロックが正しければこの重み点数を与えることを行っている. しかし, 教師はブロックが正しくても特に大きな点数を与えないが，間違っていた場合演習自体 0 点とするような，記述されていなければならない必要不可欠なブロックのチェックを行っている．このチェックを行っていないシステムの点数高くなる傾向がある．

(3) プログラムの記述量
step 関数では正解に必要な記述が少なく（図 11 参照）データフローの点数割合が演習の点数にたいして大きくなっている．また，記述量が少ない場合，教師は四則演算などの細かな処理から点数をつけているが，本システムではデータの参照関係のみを用いて採点しているため差が生じていると考えられる．

7. おわりに

本研究では, C プログラムを診断し重みに従った採点を行うシステムを実現した．データの参照関係からプログラムを診断し，プログラム中の意味的なまとまりごとの重みに従った採点を行った．採点の適切性を評価するために，システムと教師の採点結果を比較し相関係数を求めたところ，おおよそ一致した．また，今後の課題は次を考えている．

(1) 必要不可欠なブロックによる重みの診断
必要不可欠なブロックの有無のチェックを行い，記述がなければ点数を大きく下げるようにする．

(2) 重み付け UI の作成
現在重みの設定は，開発者が行わなければならないため，教師が容易に重みをつけることができるような UI を作成する．

(3) 出力結果 UI の作成
現在演習点数と意味的なまとまりの点数，それを構成するプログラム処理ごとの点数を出力している．どのような出力を行うことで，学習者への支援が行えるのか考え UI を作成する．

以上の課題を作成し，実際の講義で利用してもらうことでシステムの有用性を評価していきたい．

謝辞 評価実験にご協力いただいた九州工業大学大学院情報工学研究院所属，
中村貞吾准教授に深く感謝いたします

参考文献

1) 内田保雄:"初級プログラミング学習のための自動作問システム", 情報処理学会　研究報告, pp.109-113，2007
2) 田上恒大，阿部公輝："比較的大きなプログラミング課題のための自動採点システム"，情報処理学会　研究報告，pp.135-140，2006
3) 中村友治，竹内章："データ参照関係に基づく階層グラフによる学習者プログラムの診断", JSiSE2008 第 33 回全国大会講演論文集，pp.228-229，2008.
4) 権田克己，酒井三四郎： "ソースコードに含まれる基本的プログラミング技法の抽出"，信学技報 KBSE94-34，pp.17-24，1994.
5) Haruki UENO："A Generalized Knowledge-Based Approach to Comprehend Pascal and C Programs"，IEICE TRANS. INF. & SYST. VOL.E83-D NO.4，pp.591-598，2000
6) 鈴木浩之，小西達裕，伊藤幸宏："抽象的データ構造を含むアルゴリズム表現に基づくプログラム評価支援システムの構築"，教育システム情報学会誌　24（3），pp.167-186, 2007.