JSiSE Research Report
vol.34,no.6(2020-3)

プログラミング課題における進捗状況可視化手法の提案

秋山 直斗[*1]，新村 正明[*2]
*1 信州大学 工学部情報工学科
*2 信州大学大学院工学系研究科情報工学専攻

Proposal of a Visualization Method of Progress in Programming Task

Naoto Akiyama[*1], Masaaki Niimura[*2]
*1 Faculty of Engineering, Department of Information Engineering, Shinshu University
*2 Shinshu University Graduate School of Engineering Department of Information Engineering

概要：プログラミング教育を目的とした講義では，教授者が受講生に対して課題を与え，受講生はその課題に個別で取り組むという演習方式が採用されている．しかし，大学などにおける多人数講義の場合，少数の教授者が，多数の受講生の中から行き詰まっている受講生を全て特定するのは困難であるため，教授者の学習指導への支援が必要である．教授者が受講生の具体的な行き詰まり内容を把握しやすくする手法を提案した先行研究はあるが，教授者の学習指導が演習後になってしまうという問題点がある．本研究ではリアルタイムで教授者の学習指導を支援するための手法を提案する．

キーワード：プログラミング教育，リアルタイム，AST，GumTree，類似度，可視化

1. はじめに

プログラミング教育を目的とした講義では，教授者が受講生に対して課題を与え，受講生はその課題に個別で取り組むという演習方式が採用されている．そのため，受講生の理解度によって課題の進捗状況に差が生じる．また，大学などで行われる多人数講義の場合，TA（Teaching Assistant）を含めた少数の教授者が，多数の受講生の中から行き詰まっている受講生を全て特定するのは困難であるため，教授者の学習指導への支援が必要である．

このような教授者の学習指導を支援するものとして，先行研究[(1)]では，受講生が編集しているある時点のソースコードを用いて行き詰まり箇所を特定することで，教授者が受講生の具体的な行き詰まり内容を把握しやすくする手法を提案した．しかし，この先行研究においては教授者の学習指導がリアルタイム（演習中）になされないという問題点がある．

本研究では，同じように受講生が編集しているある時点のソースコードを用いるが，リアルタイムで教授者の学習指導を支援するための手法を提案する．

2. 先行研究

2.1 先行研究の概要

藤原ら[(1)]の研究では，井垣ら[(2)]が提案したコーディング過程可視化システム「C3PV」の機能にあるスナップショットを用いることで，プログラミング演習時に受講生がいつ，どの部分で行き詰まっていたのかを特定する手法を提案した．図 1 はこの先行研究において，受講生の行き詰まり箇所を特定する前段階の概要図である．

まず，受講生が取り組んでいるプログラミング課題に対して，編集中のソースコードを①のスナップショットとして記録する．このスナップショットとは，ソースコードのある時点の状態を指し，これは 1 分間隔もしくコンパイルや実行などの操作で記録する．

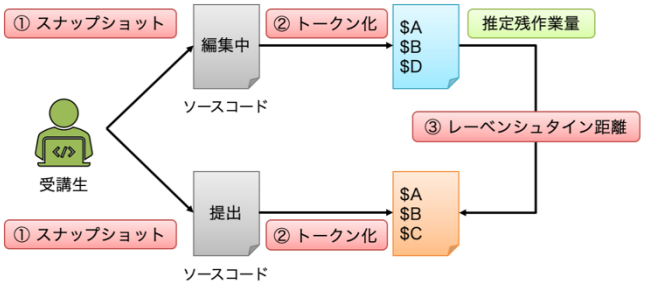


図 1 先行研究の概要図

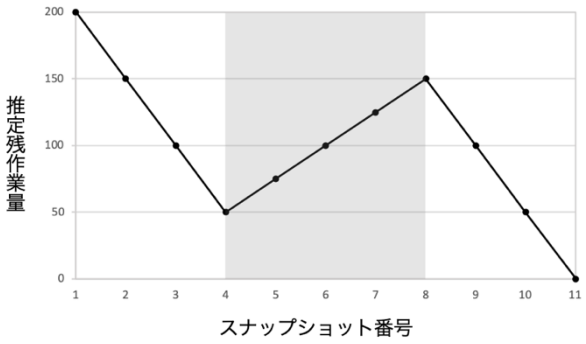


図 2 推定残作業量の推移グラフ

次に，このスナップショットとして記録されたソースコードに対して，②のトークン化を行う．トークン化とは，ソースコードを構成する文字列を一単位の文字列へ変換することを指す．そして，受講生が提出したソースコードを最後のスナップショットとして，同様にトークン化を行い，ある時点での編集中のソースコード（スナップショット）が提出されたソースコードと同一になるために，どれだけの作業量が必要であったのかを表す「推定残作業量」を③のレーベンシュタイン距離によって算出する．

この「推定残作業量」を縦軸に，スナップショットの番号を横軸にとった図 2 のような推移グラフを作成する．なお，横軸におけるスナップショットの番号は記録された時系列を表す．

このグラフの傾きが負の場合は編集中のソースコードが提出されたソースコードに近づいていることを意味しており，逆に傾きが正の場合は遠ざかっていることを表す．先行研究の提案手法では，この傾きの基準値を設定し，その基準値以上の状態が一定回数以上続いている区間を「行き詰まり区間」と定義した．

図 2 の場合は灰色部分（スナップショットの 4～8 番）がそれにあたる．そして，この区間のソースコードにおける編集部分の変化を調べることで，受講生の行き詰まり箇所を特定した．

3. 本研究の目的と提案手法

3.1 本研究の目的

先行研究の場合，受講生が提出したソースコードで

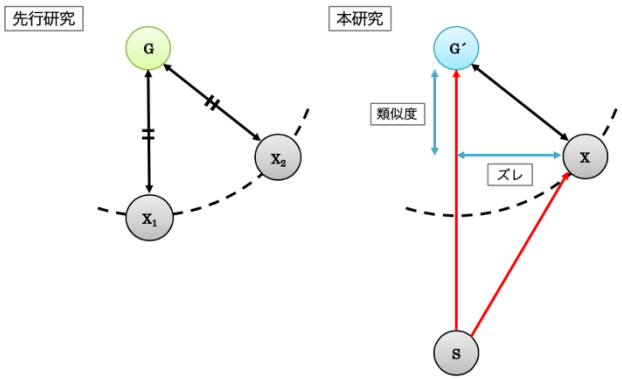


図 3 本研究の提案手法

比較をするため，リアルタイムに行き詰まっている受講生に対しての学習指導がなされない．

そこで，本研究では，次の 3 点をリアルタイムで実現し，教授者の学習指導を支援するための手法を提案する．

① 課題の進捗状況の可視化
② 行き詰まっている受講生の特定
③ これから行き詰まりそうな受講生の早期発見

3.2 提案手法の概要

先行研究では，図 3 のように提出されたソースコード G という演習後に設定されるゴールがあり，スナップショット X との「推定残作業量」を算出していた．しかし，スナップショット X_1 と X_2 の「推定残作業量」に変化がない場合，この 2 つは G を中心とした円周上のどの位置にいるのかが分からないため，受講生の解答の方向性が見えてこない．

そこで本研究では，スタート S とゴール G´をあらかじめ設定し，S～G´間の距離と S～X 間の距離を調べ

ることで，スナップショット X のゴール G′に対する「類似度」と「ズレ」をリアルタイムに算出する方法を提案する．この「類似度」と「ズレ」により，スナップショット X の位置が決まるため，リアルタイムで動く X を見たときに，受講生の解答がどのような方向性で進んでいるのかを知ることが可能になり，誤った方向に進もうとしている受講生を早期に発見できると考えられる．

3.3 提案手法の実現例

先行研究で用いられたレーベンシュタイン距離ではプログラムの構造を捉えられないため，ソースコードの分析には適していない．よって，本研究では抽象構文木（Abstract Syntax Tree，以降 AST とする）のノード単位でソースコードを分析する．

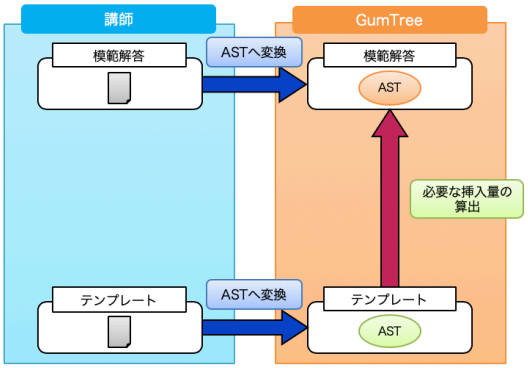
また，本研究では C 言語を対象とするため，これらを踏まえて，GumTree[3][4]をソースコードの解析ツールとして選定した．GumTree は 2 つのソースコードを入力として与えることでそれぞれの AST を生成し，この AST 間でノードの一致，挿入，削除，更新，移動の個数を検出可能である．

この GumTree を用いた提案手法の概要を図 4 に示す．まず，教授者側では 1 つのテンプレートと模範解答のソースコードをそれぞれ用意する．テンプレートとは，図 5 のようにプログラムの形がある程度決まっているものを指す．教授者は受講生に対して，学習して欲しいポイントを灰色部分のようにコメントとして記述し，受講生はこの部分をスタート地点として編集を開始する．模範解答とは，赤字で示されたような正解例を指す．

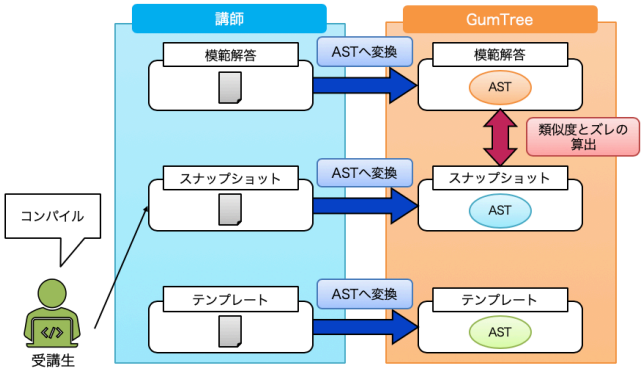
このテンプレートと模範解答のソースコードを GumTree によってそれぞれ AST へと変換し，図 4 の（ｉ）のようにテンプレートが模範解答と同一になるために必要な挿入量を算出する．

次に，（ii）のように受講生がコンパイルをしたタイミングをスナップショットとして記録し，同様に GumTree によって AST へと変換する．

そして，AST 上でスナップショットの模範解答に対する「類似度」と「ズレ」を算出することで，ある時点のスナップショットが模範解答に対してどの位置にいるのかを特定する．



（ｉ）必要な挿入量の算出



（ii）類似度とズレの算出

図 4 提案手法の概要

```
#include <stdio.h>

int main() {
    int num ;

    scanf("%d", &num) ;

    /* ここに処理を記述する */
    /* ここまで */

    return 0 ;
}
```

テンプレート

```
#include <stdio.h>

int main() {
    int num ;

    scanf("%d", &num) ;

    /* ここに処理を記述する */
    if(num == 3) {
        printf("Hello World!¥n") ;
    }
    /* ここまで */

    return 0 ;
}
```

模範解答

図 5 教授者が用意するもの

3.4 類似度とズレの算出方法

図 6 はテンプレート，模範解答，スナップショットをそれぞれ AST 間のノード数で表したものである．

3.3 の GumTree の説明で，AST 間のノード数検出には一致，挿入，削除，更新，移動があると述べたが，更新と移動に関しては一致に含まれる．よって，図 6 の算出方法では一致，挿入，削除の 3 種類を用いる．

（イ）では，テンプレートから模範解答を見たときに，例として，一致するノード数 m が 130 個，テンプレートが模範解答と同一になるために必要な挿入量 i_g を 145 個であるとする．

（ロ）では，あるスナップショットが記録され，テンプレートからこのスナップショットを見たときに，挿入されたノード数 i_s が 125 個であるとする．

（ハ）では，スナップショットから模範解答を見たときに，この挿入されたノード数 i_s の 125 個のうち，模範解答と一致したノード数 m_s（= m_g）が 100 個あったとすると，模範解答とは違ったノード数 d_s が 25 個あると分かる．

よって，「類似度」は模範解答と新たに一致したノード数 m_s の 100 個を，「ズレ」は模範解答とは一致しなかったノード数 d_s の 25 個を，最初に必要な挿入量 i_g であった 145 個で割った値として算出する．

また，ここまでの算出式は(1)，(2)のようになる．

(1) 類似度 = $\dfrac{m_s}{i_g}$

(2) ズレ = $\dfrac{d_s}{i_g}$

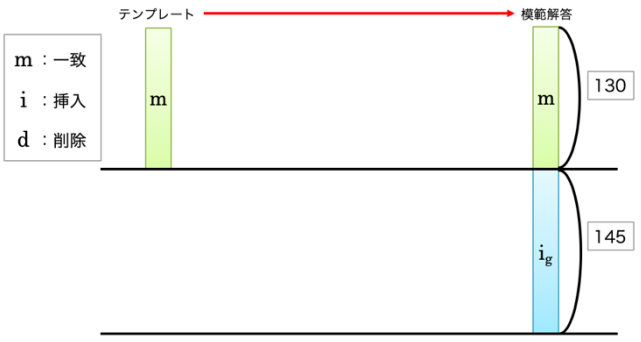
4. 評価

4.1 提案手法を用いた散布図表示

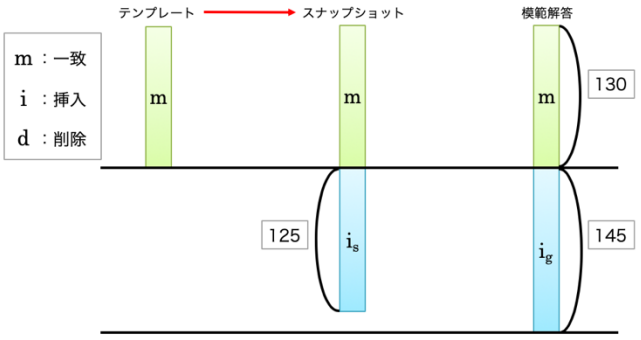
本提案が受講生のプログラミング過程を正しく表現しているかを確認するために，ある受講生のスナップショット群に対して類似度とズレの算出を行った．この結果を散布図として表現したもの図 7 に示す．

縦軸の類似度は[0，1]の区間で正規化されており，1 に近いほど模範解答に近づいていることを表す．また，横軸のズレは値が大きいほど，模範解答にはない部分があることを表す．また，プロットの番号はスナップショットの時系列を表す．

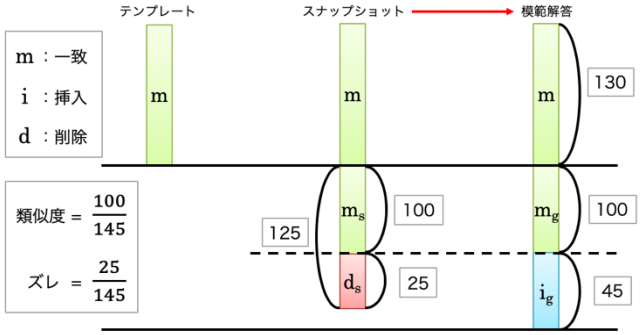
実際のスナップショットの内容と，散布図のプロットの対応を調べたところ，赤枠①の 1～13 番はほぼズレがなく，ソースコードも順調に進んでいることが確認できた．また，赤枠②の 14～33 番では 1～13 番に比べてズレがあり，ソースコードにおいても模範解答にはない処理が散見された．赤枠③における 36 番が最後のスナップショットであり，散布図でズレは見



（イ）必要な挿入量の算出



（ロ）挿入されたノード数



（ハ） 類似度とズレの算出

図 6 類似度とズレの算出法

られるものの実際のソースコードでは正しい実行結果を出力していた．

以上から，受講生のプログラミング過程が，散布図におけるプロット順として可視化できていることが確認できた．

4.2 複数の正解がある場合の比較

次に，複数の正解が考えられる課題に対して，ある受講生のスナップショット群で類似度とズレの算出を

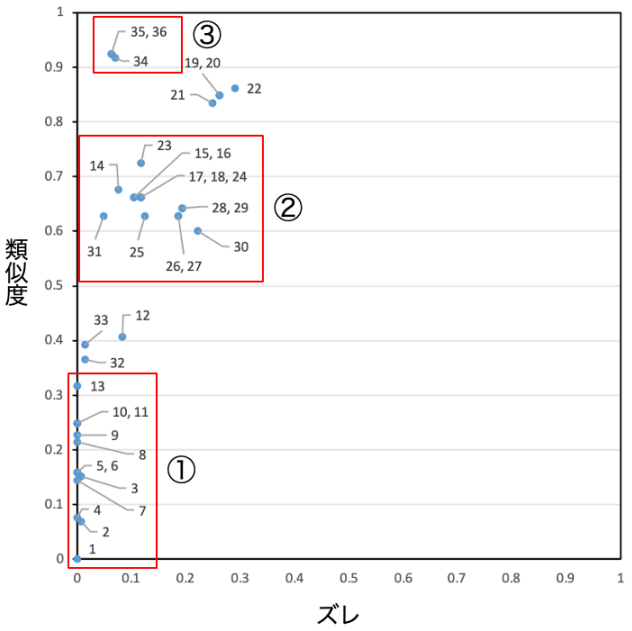


図 7 ある受講生における散布図

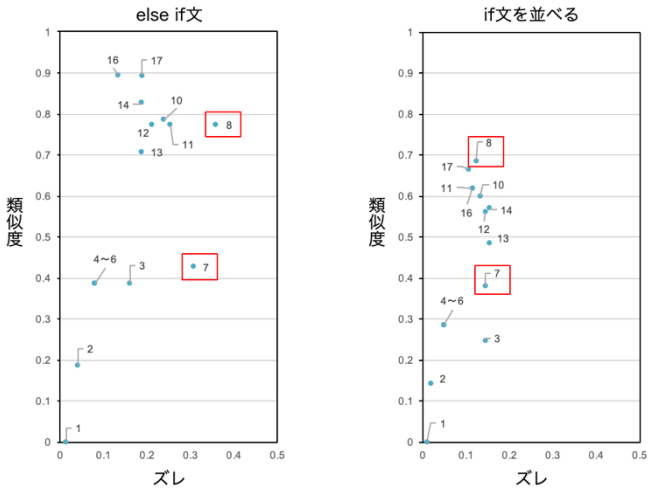


図 8 別解との比較

行った．その結果を図 8 に示す．なお，プロットの 9 番と 15 番に関しては GumTree による計算ができていなかったため，散布図からは省いている．

この課題における解答は，else if 文を使って解く場合と単純に if 文を並べて解く場合の 2 つの解法がある．

実際のスナップショットの内容と，散布図のプロットの対応を調べたところ，赤枠で囲った 7，8 番では if 文を並べて解く方向で進んでおり，散布図を比較すると，else if 文よりも if 文を並べる方がズレは小さくなっている．そして，プロットの 10 番以降では else if 文を使う方向で解いており，else if 文側の散布図ではズレが小さくなっていく動きが見られる．

このことから，プログラミング過程における解法の方向性も可視化できることが確認できた．

5. おわりに

本研究では，リアルタイムで教授者の学習指導を支援するための手法を提案した．

今回は類似度とズレの算出法の検討から課題の進捗状況の可視化を表現できたが，そこから行き詰まっている受講生の特定やこれから行き詰まりそうな受講生の早期発見の段階までには至らなかった．本研究における行き詰まりの見解としては，図 7 の赤枠②のように同じ場所を行き来する場合である．また，図 8 における else if 文側のプロット 7，8 番のような模範解答とは違う処理をしている場合に，類似度の値としては大きくなっているが，同時にズレの値も大きくなっていくような動きがこれから行き詰まる受講生の早期発見につながるのではないかと考えられる．

参 考 文 献

(1) 藤原賢二，上村恭平，井垣宏，吉田則裕，伏田享平，玉田春昭，楠本真二，飯田元：“スナップショットを用いたプログラミング演習における行き詰まり箇所の特定”，コンピュータソフトウェア 35（1），pp.3-13，（2018）

(2) 井垣宏，斎藤俊，井上亮文，中村亮太，楠本真二：“プログラミング演習における進捗状況把握のためのコーディング過程可視化システム C3PV の提案”，情報処理学会論文誌 54（1），pp.330-339，（2013）

(3) GumTreeDiff/gumtree : A neat code differencing tool. https://github.com/GumTreeDiff/gumtree.（2019 年 7 月 5 日確認）

(4) GumTreeDiff/cgum : The C parser for Gumtree. https://github.com/GumTreeDiff/cgum.（2019 年 7 月 5 日確認）