プログラミング演習における学習者の行き詰まり同定を目的とした評価指標の妥当性に関する考察

A Consideration of the Validity of Assessment Metrics for Identification of Learners' Stalemate in Programming Exercises

田中 空来[*1], 香山 瑞恵[*2], 新村 正明[*2], 舘 伸幸[*2]
Sora Tanaka[*1], Mizue Kayama[*2], Masaaki Niimura[*2], Nobuyuki Tachi[*2]
[*1] 信州大学大学院 [*2] 信州大学
[*1]Graduate School of Science & Technology, Shinshu University [*2]Shinshu University
Email: {kayama, niimura}@shinshu-u.ac.jp

あらまし: 本研究の目的はプログラミング演習における学習者の行き詰まりを同定することである．そのため，模範解答との類似度と相違度に基づく評価指標を提案する．本稿では，まず，プログラムに対応する抽象構文木（AST）から算出した類似度と相違度の計算方法を示す．次に，これらの指標と一般的なプログラムメトリックスとの関係を整理する．この結果に基づき，提案指標の妥当性を考察する．
キーワード: プログラミング演習，行き詰まり，類似度，相違度，AST

1. はじめに

プログラミング教育では，指導者が学習者に対して課題を与え，学習者はその課題に個別で取り組む演習形式が採用されることが多い(1)．本研究では，このような形式をプログラミング演習と称する．学習者が多数の場合，少数の指導者が，行き詰まっている学習者を特定するのは困難である．そのため，学習者の進捗状況を確認するツール(2-4)などが提案されている．これらの研究では，学習者の編集操作やコマンド実行等の行動，コンパイル時や実行時のエラーなどに基づく評価指標が用いられているが，ソースコードの質的な評価指標は考慮されていない．

これに対して，本研究では，学習者が作成したソースコード（以下，学習者コード）の質的な評価指標の開発を試みてきた (5)．本稿では，模範解答のソースコード（以下，模範コード）に対する類似度と相違度に基づく評価指標を提案する．

2. 提案する評価指標

提案評価指標である類似度は「学習者コードと模範コードが構造的に一致するノード(後述)の割合」，相違度は「学習者コードの中で模範コードに含まれないノードの割合」である．以下，2.1 でこれらの評価指標の設計概要を示し，2.2 で計算方法を示す．

2.1 設計概要

本研究では，抽象構文木（Abstract Syntax Tree，以下, AST）のノード単位でソースコードを分析する．ここでは，GumTree(6,7)をソースコードの解析ツールとして利用する．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，AST 間でノードの一致，挿入，削除，更新，移動を検出するが, ソースコードの構造が破綻していると，AST が生成できない場合がある．本研究では，このうち一致と挿入を使用している．GumTree に与える

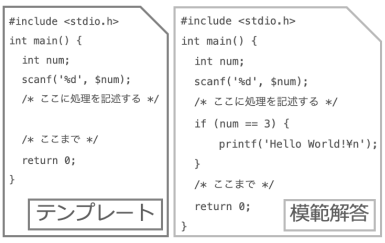

図 1 テンプレート(t)の例　図 2 模範解答(a)の例

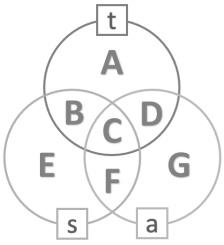

図 3 各コードの関係性

ソースコードは，テンプレート（以下，t），模範解答（a），スナップショット（s）の 3 種とする．t と a は指導者があらかじめ用意しておく（図 1,2）．学習者は t を用いてプログラミングを始める．図 1 の t には学習者が処理を記述する範囲がコメントで示されており，学習者はその中のみ編集を行うことが期待される．図 2 の a は t に処理が追記された模範解答例である．s は編集途中の学習者コードであり，編集が加えられた場合に一定間隔で自動保存される．

2.2 計算方法

図 3 に t，a，s に含まれるノードの関係を示す．A～G を GumTree で解析されたノードの部分集合とした場合, それぞれ, t = {A, B, C, D}, a = {C, D, F, G}, s = {B, C, E, F}となる．このうち E は s のみに存在するノードであり， F は s と a にのみ存在するノードである．t には，基本的に A，B のノードは存在しないが，一部課題で t を編集可能であるため，t の要素に含んでいる．類似度と相違度の計算式は図 3 の要素を用いて次の(1)，(2)のように表される．両指標の値域は，[0, 1]である．

(1) 類似度 $= \frac{\{A,F\}}{\{A,B,F,G\}}$ ，(2) 相違度 $= \frac{\{D,E\}}{\{A,D,E,F\}}$

3. 提案指標の妥当性の検証

一般的なプログラムメトリックスである，Jaccard

表 1　各パラメータとの相関係数

相関係数		Jaccard 係数	類似度	相違度	CC	行数
Jaccard 係数	s_ok		0.96	-0.32	0.63	0.82
	s_ng					
類似度	s_ok	0.96		-0.19	0.75	0.9
	s_ng	0.96				
相違度	s_ok	-0.28	-0.15		0.01	-0.07
	s_ng	-0.39	-0.25			
CC	s_ok	0.64	0.77	0.05		0.59
	s_ng	0.61	0.72	-0.05		
行数	s_ok	0.82	0.9	-0.03	0.63	
	s_ng	0.83	0.89	-0.14	0.53	

係数，循環的複雑度（Cyclomatic Complexity，以下，CC），ソースコードの行数（以下，行数）と提案指標の関係を整理する．

3.1 プログラムメトリックスの概要

Jaccard 係数は 2 つの集合の和集合に対する積集合の割合を計算するものである．a と s の Jaccard 係数は(3)のように表され，値域は，[0, 1]である．また，1 から Jaccard 係数を引いた値を Jaccard 距離と言う．

(3) $J(a,s) = \frac{|a \cap s|}{|a \cup s|} = \frac{\{C,F\}}{\{B,C,D,E,F,G\}}$

CC は，ソースコードの複雑度を測るメトリックスである．行数は，空行やコメントアウトされている箇所を省いて求められる．

3.2 実験条件

2021 年度に実施された A 大学情報系学科 2 年生向けの C 言語プログラミング演習を対象とした．70 分間の期末試験に出題された 5 題のうち 1 課題（以下，解析課題）を用いた．解析課題を解いた 147 名の学習者コード（17,234 ファイル）のうち，AST 生成が成功したコード（5,813）を s とした．なお，これらの s は 2 秒間隔で自動保存されていた．

3.3 結果

Jaccard 係数，類似度，相違度，CC，行数の相関を表 1 に示す．表 1 の薄灰色部分は s の相関係数である．類似度に対する Jaccard 係数の相関係数は 0.96，CC は 0.75，行数は 0.9 となった．一方，相違度に対する相関係数は全ての指標で絶対値で約 0.32 以下となった．黒色部分はコンパイルが成功した s_ok（3,707）と失敗した s_ng（2,106）の相関係数である．これらのいずれの指標でも s_ok と s_ng が同様の値を示した．このことから，相関係数はコンパイルの成否の影響を受けないことがわかった．

3.4 考察

3.3 の結果から，類似度と Jaccard 係数は，同等の指標と考えられる．相違度は他の指標と相関が低い．相違度の性能を確認するために，類似度と行数の散布図を相違度の値で色分けした（図 4）．図 4 では相違度 0.23 未満の●とそれ以外の●の分布が重なっている（●と●は各約 50%）．すなわち，相違度は類似度や行数とは異なる性質を表していると考える．

これらの結果から，類似度と相違度を質的な評価指標，CC と行数を行動的な評価指標とすることで，学習者の行き詰まりを同定できると考える．また，

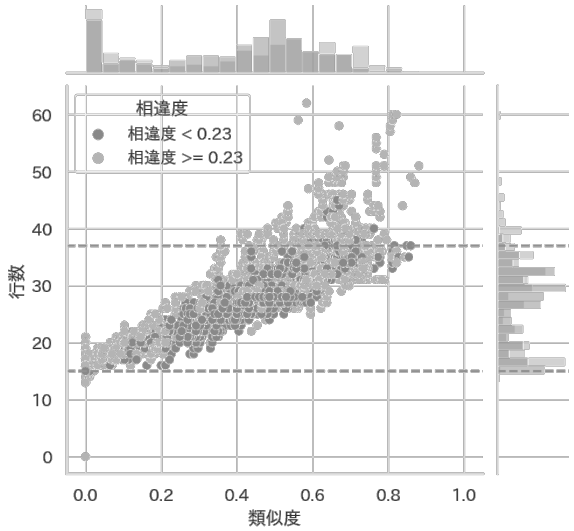

図 4 類似度と行数の散布図における相違度の分布

類似度は t，s，a の 3 種のコードを利用するため，t の要素を除いて計算できる点が Jaccard 係数よりも優れていると考える．

4. おわりに

本稿では，プログラミング演習における学習者の行き詰まり同定を目的とした評価指標の妥当性を考察した．特に，類似度と相違度の有用性を検証した．今後は他の複数課題での検証により，この結果の再現性を確認する．また，評価指標の時系列変化や個人の特徴を整理する．その上で，プログラミング演習支援システムの具体化を図る．

参考文献

(1) 槙原絵里奈，藤原賢二，井垣宏ほか：“初学者向けプログラミング演習のための探索的プログラミング支援環境 Pockets の提案”, 情報処理学会論文誌, Vol.57, No.1，pp.236-247（2016）
(2) 市村哲，梶並知記，平野洋行：“プログラミング演習授業における学習状況把握支援の試み”，情報処理学会論文誌, Vol.54，No.12，pp.2519-2526（2013）
(3) 井垣宏，斎藤俊，井上亮文ほか：“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可視化システム C3PV の提案”，情報処理学会論文誌, Vol.54，No.1，pp.330-339（2013）
(4) 堀口諒人，筒井善規，井垣宏：“プログラミング演習における学生のプログラミング行動推定のための授業環境と実験環境の比較”, 第 7 回実践的 IT 教育シンポジウム（rePiT2020）論文集，pp.114-120（2020）
(5) 秋山直人，新村正明：“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”，教育システム情報学会研究会講演論文集，Vol.34，No.6，pp.51-55（2020）
(6) J.-R.Falleri, F.Morandat, X.Martinez, et al.,“Fine-grained and accurate source differencing”，Proc. of the 29th ACM/IEEE International Conference on Automated Software Engineering，pp.313-324（2014）
(7) GumTreeDiff/gumtree : A neat code differencing tool – GitHub，https://github.com/GumTreeDiff/gumtree（2022 年 5 月 17 日 確認）