全自動プログラミング演習に向けて

井垣 宏[1,a)]　藤原 賢二[2,b)]　吉田 則裕[3,c)]　槇原 絵里奈[4,d)]

概要：プログラミング教育では，学生一人ひとりの習熟度にあわせた問題の作成や進捗管理，フィードバックが有用であるとされている．一方で，多くの学生を対象とした演習環境において，各学生の状況に合わせた演習を実施するためには，非常に大きなコストがかかる．そこで我々は，プログラミング教育における課題を分類し，学生の詳細なプログラミング活動データを用いた機械学習による解決の可能性を検討する．

1. はじめに

プログラミング教育需要の高まりに伴い，様々な形式でプログラミング演習が行われるようになりつつある．大学等の授業の一環で実施されるもの以外にもオンラインジャッジ [1] と呼ばれる仕組みを利用したオンラインでのプログラミング教育やコンテストなどが実施されている．

大学等の高等教育機関において授業として実施されているプログラミング演習は，通常数 10 名〜100 名程度の学生を少数の教員や TA が指導する．学生は教員から対象の言語やアルゴリズムについての講義や指導を受け，その後指導内容に即した複数の課題を実際に解くことで，言語やアルゴリズムについて習熟していく．プログラミング演習において教員は講義の作成やプログラミング課題の作成，学生の進捗状況にあわせたサポートや提出されたプログラムの評価，フィードバックを行う．

本研究では，初学者等も多く含まれる大学教育におけるプログラミング演習科目を高い品質を保ったまま全自動で行うために必要なデータや分析手法について検討を行う．

2. プログラミング演習

これまで，様々な観点にもとづくプログラミング演習自動化が取り組まれてきた．Wakatani ら [2] はテンプレートを利用した課題の自動生成を行っている．他にも学生のプログラミング行動をソースコード差分 [3] や制御フローグラフ [4] の観点で可視化することによる進捗状況の把握支援が行われている．提出されたソースコードの評価やフィードバックもプログラミング演習における非常に重要な課題と認識されており，間違いの修正や学生の成績付け等様々な自動化が取り組まれている [5]

本研究において我々が対象とするプログラミング演習の前提は以下に述べるとおりである．

- 初学者を含む学生を対象として実施されている．
- 課題の規模は小さく，たかだか数十行程度までを想定している．
- ほとんどの課題が入出力を伴うコンソールアプリケーションの開発を対象としている．
- プログラミング環境は教員が提供し，学生はその環境を利用して課題を解く．

我々はこのようなプログラミング演習を前提として，機械学習を用いた課題の作成，進捗管理，評価及びフィードバックの支援の可能性について検討する．

3. 全自動プログラミング演習に向けて

教員によって提供されるプログラミング環境を利用することで，学生によって最終的に提出されるソースコードだけでなく，課題ごとのソースコード編集履歴も比較的容易に収集できる．また，環境によってはプログラムのコンパイル・実行履歴も収集可能である．そのため，全自動プログラミング演習に向けた最初のステップとして，これらのデータを利用したプログラミング演習支援手法の可能性について検討する．

3.1 ソースコードの自動評価に係る課題

通常，学生によって提出されたソースコードを評価する

[1] 大阪工業大学
1-79-1 Kitayama, Hirakata-shi, Osaka 573-0196, Japan
[2] 豊田工業高等専門学校
2-1 Eisei-cho, Toyota-shi, Aichi 471-8525, Japan
[3] 名古屋大学
Furo-cho, Chikusa-ku, Nagoya-shi, Aichi 464-8601, Japan
[4] 奈良先端科学技術大学院大学
8916-5 Takayama-cho, Ikoma-shi, Nara 630-0192, Japan
[a)] hiroshi.igaki@oit.ac.jp
[b)] fujiwara@toyota-ct.ac.jp
[c)] n-yoshida@nces.is.nagoya-u.ac.jp
[d)] makihara.erina.lx0@is.naist.jp

場合，事前に用意されたテストケースを利用することが多い．特にオンラインジャッジ等では，提出されたソースコードに対して複数のテストケースを走らせることで，どの程度完成しているかを評価する枠組みが確立している．

一方で，我々が想定する初学者向けのプログラミング演習では，テストケースのみでソースコードの自動評価を行うことには以下のような問題がある場合がある．

A1: テストケースによる判定結果のみでは提出したソースコードのどこに問題があるか学生が理解できない場合がある

A2: A1 について，テストケースそのものも学生に提示すると，学生が与えられたテストケースのみが通るプログラムを作成してしまうことがある

A3: 特定の文法要素を利用して欲しいといった入出力以外の要素にもとづく評価ができない

A4: 入出力結果が正解例と完全に一致しないと正解と判定されない．そのため，ちょっとした出力の表記ゆれや改行・スペースの違いだけで不正解と判定される．

A1 について，“テストケース 10 個のうち 7 個が通りました”，といった結果だけを学生が与えられたとき，特に初学者は自分の作成したプログラムのどこに間違いがあるかを理解できないことがある．ここでテストケースの詳細を学生に提示してしまうと A2 の問題が発生する．A3 については，例えば for 文を指導したにも関わらず，while 文しか使っていない場合が相当する．A4 では，オンラインジャッジ等のすべてのテストケースが正常にとおることが要件である場合は問題にならない．しかしながら，初学者を対象とする場合は，ちょっとした表記ゆれは許容し，本質的な文法やアルゴリズムの間違いのみに集中して欲しいといったケースも存在する．実際に第一著者の大学で実施しているプログラミング演習では正規表現を利用したチェックツールを教員が実装しており，学生の提出したプログラムと正答の間にスペースや改行，ちょっとした表記ゆれといった違いが合った場合でも正解と判定するようにしている（非常に多くの学生が不正解となってしまうため）．結果として A4 の状況を改善できてはいるが，そのために非常に大きなコストをかけてチェックプログラムを開発しているのが現状である．

3.2 ソースコード自動評価支援の検討

A1〜A4 をふまえると，我々が意図するソースコードの自動評価は下記のような要件を満たしていることが望ましい．

R1: 入出力だけでなく，ソースコード中の記述も評価対象に含まれる

R2: ちょっとした表記ゆれ・改行・スペース等の違いは許容され，本質的な違いのみを間違いと判定する

R3: テストケースや入出力チェックプログラムの開発にかかるコストができるかぎり小さい

R1〜R3 にもとづき，以下に示す自動評価手法の実現を目指す．

S1: 学生の提出したソースコードとテストケースにもとづく入出力値の組み合わせを分類する

S2: S1 で分類した結果にもとづき，代表値を手作業で評価し，ラベル付を行う

S3: S2 の結果に対して機械学習手法を適用し，他のソースコードに適用する

S1 では，ソースコードを一定の規準で正規化し，自然言語処理技術でも利用される手法を用いた分類を行う．S2 では，分類によって発生した一定以上の規模のクラスタを対象に，典型的なソースコード及び入出力値の組み合わせを選別し，そのソースコードが正解か不正解か等の分類を行う．S3 では，S1，S2 による分類結果及びラベル付けの結果を用いて学習を行い，他のソースコードにも適用する．この手法では，教員は課題とその課題に対するテストケースを用意すれば，あとは実際の学生数よりも少ない数のソースコードを評価するだけで，全ての提出されたソースコードの評価が可能となる．

4. おわりに

初学者を含む多くの学生に指導を行うプログラミング演習を前提として，現状の把握と収集可能なデータ，ソースコード自動評価支援手法の検討を行った．今後は検討した支援手法を実現し，精度や教員に求められるコストのより具体的・実証的な分析を進めていきたい．

謝辞 本研究は科研費（17K00500）の助成を受けたものである．

参考文献

[1] 渡部有隆：オンラインジャッジの開発と運用 -Aizu Online Judge-，情報処理，Vol. 56, No. 10, pp. 998–1005 (2015).

[2] Wakatani, A. and Maeda, T.: Automatic generation of programming exercises for learning programming language, *Proc. 2015 IEEE/ACIS 14th International Conference on Computer and Information Science (ICIS)*, IEEE, pp. 461–465 (2015).

[3] Makihara, E., Igaki, H., Yoshida, N., Fujiwara, K. and Iida, H.: Detecting exploratory programming behaviors for introductory programming exercises, *Proc. 2016 IEEE 24th International Conference on Program Comprehension (ICPC)*, IEEE, pp. 1–4 (2016).

[4] Hovemeyer, D., Hellas, A., Petersen, A. and Spacco, J.: Control-Flow-Only Abstract Syntax Trees for Analyzing Students' Programming Progress, *Proceedings of the 2016 ACM Conference on International Computing Education Research*, ACM, pp. 63–72 (2016).

[5] Keuning, H., Jeuring, J. and Heeren, B.: Towards a systematic review of automated feedback generation for programming exercises, *Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education*, ACM, pp. 41–46 (2016).