情報処理学会論文誌 Vol.63 No.3 742–751 (Mar. 2022)

オンラインジャッジシステムにおける解答履歴を利用した問題の関係性調査

槇原 絵里奈[1,a)] 池田 太郎[2,b)] 小野 景子[1,c)] 新濱 遼大[2,d)]

受付日 2021年5月31日, 採録日 2021年12月3日

概要：オンラインジャッジシステムには多くの問題が収録されており，これらをプログラミング学習におけるアルゴリズムの自学自習支援として活用することで，教員の問題作成コストをへらすことができる．一方，学生が主体的に自身のプログラミング能力に適した問題を選択することは困難であり，最適な問題自動選択手法が望まれている．本論文では，オンラインジャッジシステムにおけるユーザの解答履歴に着目し，解答履歴，解答の正誤を深層学習モデルある Long Short Term Memory（LSTM）により学習し，問題間の関係に基づいた問題推薦が可能な手法を提案する．オンラインジャッジシステムである Codeforces の実データを対象に提案法の性能を検証し，ユーザの解答履歴の高い推定性能を確認した．また，遷移の可視化により，難易度の高い問題へ遷移するための最短な問題経路や，様々な種類の問題へ着手できる核となる問題を明らかにした．

キーワード：プログラミング学習，自学自習支援，オンラインジャッジシステム，機械学習

Problem Characteristics in Online Judge System by Using History of Submitted Source Code

Erina Makihara[1,a)] Taro Ikeda[2,b)] Keiko Ono[1,c)] Ryota Shinhama[2,d)]

Received: May 31, 2021, Accepted: December 3, 2021

Abstract: The online judge system contains a large number of problems. The problems can help educators to reduce the cost of creating problems as a student's homework in programming education. However, it is difficult for a student to select the problem which suitable for improving her/his programming skill. Therefore, the automatic recommendation method of the optimum problem for improving a user's programming skill is required. In this paper, we focused on the logs of submitted history by users in the online judge system. By learning the submitted history and corresponding result using Long Short Term Memory (LSTM) which is one of the deep learning model, we propose the method of automatic problem recommendation based on the relationship on problem characteristics. We experimented by using users' submitted history data of Codeforces, a famous online judge system. Out proposed method demonstrated the high performance of the estimation regarding a user's submitted history. Furthermore, visualizing the transition of submitted history revealed the shortest path to transit more difficult problems, as well as the core problem to transit various types of problems.

Keywords: programming education, self-study support, online judge system, machine learning

1. はじめに

IoT や AI が世間に広く普及したことにより，プログラミング学習者も増加の傾向にあり，初学者を対象とした講義や学習サイトが増加している [1]．大学においても学生が実際にプログラミングを行いながら，特定のプログラミング言語の文法や機能について主体的に学ぶことができるプログラミング演習という科目が開講されている．しかし

[1] 同志社大学理工学部
Faculty of Science and Engineering, Doshisha University, Kyotanabe, Kyoto 610–0394, Japan
[2] 同志社大学理工学研究科
Graduate School of Science and Engineering, Doshisha University, Kyotanabe, Kyoto 610–0394, Japan
a) emakihar@mail.doshisha.ac.jp
b) tikeda@mikilab.doshisha.ac.jp
c) kono@mail.doshisha.ac.jp
d) shinhama.ryota@mikilab.doshisha.ac.jp

ながら，プログラミング教育の中でもアルゴリズムの学習は言語には依存しない一方，単純にエラーが生じないプログラムを作成するだけでなく，実行時間やメモリ使用量など，決められた制約のもと問題解決を図る能力が必要となる [2]．したがって，アルゴリズム教育を促すためには様々な制約を課した問題を与える必要があるが，教員に作問の負担を強いるうえ，ユーザによって学習環境が異なる自学自習において，メモリ使用量などを課した問題を与えることは困難である．

そこで本研究では自学自習におけるアルゴリズム学習の支援を目的に，オンラインジャッジシステム（Online Judge System，以下 OJS）に注目する．OJS とは，オンライン上でソースコードを送信するとコンパイル・テスト・採点が自動で行われ，その結果の閲覧が可能なシステムのことを指す [3]．OJS は国内外問わず複数存在し，定期的にプログラミングコンテストを開催している．また，コンテストで使用した問題はコンテスト終了後に OJS 上で一般公開され，誰もが自由に解くことができる．さらにほとんどの OJS では他の学習者が提出したソースコードを閲覧できるため，問題の解法が分からなかった場合他のユーザの解答例を参照することも可能である．

以上より OJS を利用することで，ユーザは様々な種類の問題に触れることができ，自身のプログラミング能力やアルゴリズムに対する理解度を向上させることが可能であると考える．しかしながら OJS には様々なレベルの問題が存在するため，プログラミング学習を目的にしたユーザが適切な問題を選択できなかった場合，プログラミング学習そのものや OJS を使用することに対するモチベーションが低下する恐れがある．実際，教育機関において単に OJS を使用するだけでは学生のプログラミング能力は向上しないことや [4]，OJS の 1 つである AOJ において，利用者の約 88%が 100 問以下，90%が 50 日以内しか取り組んでいないという報告がある（則行 [5]）．則行はユーザが OJS を継続利用しない原因について，気軽に取り組むユーザが多いためであると推測した．則行が述べるように OJS は個人で取り組むため強制力がない点に加え，OJS サイトや問題によっては英語で出題されるため理解が困難である点，論理的思考力や読解力も必要である点，毎日取り組むだけではレート [6] が上がらない点なども，継続利用者が少ない原因として考えられる．すなわち，ユーザが OJS を継続利用できない原因は複数の要因が考えられ，単純に同難易度の問題を与えるだけでは解決が困難であることが推測される．ユーザの学習意欲を維持しつつ適切な問題を与えるためには，単純に同じ難易度の問題を与えるだけでなく，どのように解き進めたらプログラミング能力の向上につながるか，問題全体の関係性を把握する必要がある．

そこで我々は，OJS における問題間の関係に基づいた問題推薦を提案する．先行研究において，問題文や提出されたソースコードの類似度に基づく問題間の関係について調査した [7]．しかしながら，OJS における問題文は物語調になっていることも多く，またソースコードも多様な書き方で提出できるため，問題文や提出されたソースコードからは問題間の関係を見つけることは困難であった．そこで，問題文や提出されたソースコードに依存しない手法として，本論文ではユーザの解答履歴に着目した．Codeforces では，ユーザの解答履歴から問題番号とその問題に対する正誤判定が得られる．それらの情報を利用することで，同じような問題に取り組んでいる先行学習者の解答履歴より，次に取り組むべき課題を推定でき，これまで困難であった，個人のアルゴリズム理解度やプログラミング能力，問題選択の方針などに最も適した問題推薦が可能になると考える．また，問題への正誤情報より正解答を導く解答順や，次の問題につまずく，すなわち誤解答する確率も導出可能である．

本研究では，プログラミング学習やアルゴリズム学習における OJS の継続利用を促すための第一歩として，学習者の学習履歴から次に取り組む問題の推定とその正誤確率を推定可能なモデルを提案する．これらの情報は時系列データであり，時系列データからのラベル推定が可能である深層学習モデルより頻繁に用いられる LSTM（Long Short Term Memory）の適用を考える．さらに，予測結果を可視化し，問題どうしの関係性をネットワークとして表現することで，OJS の豊富な問題を自学自習に利用できるだけなく，ユーザに対しプログラミング能力の向上につながるような問題提供が可能になると考える．

本論文の構成は次のとおりである．まず 2 章において本研究の対象となる OJS について，概要および関連研究をふまえ，プログラミング教育へ導入するうえでの課題を述べる．次に 3 章では，ユーザの解答履歴を基に LSTM を用いて問題間の関係を推定する提案手法，および結果について述べる．4 章では作成したモデルの評価として，3 章で作成したモデルに解答履歴を与え，問題での遷移モデルを可視化した．結果をふまえ，本提案手法のプログラミング教育に対する有効性に関して述べる．5 章では本提案手法の妥当性の脅威と，教育現場への導入に関し考察を述べる．最後に，6 章でまとめを述べる．

2. オンラインジャッジシステム

2.1 概要

オンラインジャッジシステム（OJS）とは，ユーザから提出されたプログラムをコンパイル・実行し，複数のテストケースや検証機などを用いてプログラムを評価し，ユーザへ結果をフィードバックするシステムを指す [3]．代表例として，国内では AIZU ONLINE JUDGE[*1] や AtCoder[*2]，

[*1] https://onlinejudge.u-aizu.ac.jp/home
[*2] https://atcoder.jp/

国外では TopCoder [*3]や Codeforces [*4]などが存在する．これらサイトでは定期的に競技プログラミングやプログラミングコンテストが行われている．プログラミングコンテストでは複数の参加者が同時刻に同じ問題を解いていき，正解問題数や解答時間などに応じてユーザの順位付けやレーティング [8] が行われる．また，OJS サイトには過去のコンテストの問題や練習問題などが多く収録されており，コンテストに関係なく，ユーザは 24 時間いつでも自由に問題に着手し，ソースコードを提出することが可能である．

本論文では OJS の中でも世界最大規模であり，コンテストなどユーザの活動が活発で API が充実している Codeforces を調査対象とする．

2.2 Codeforces

Codeforces とはロシア発祥のプログラミングコンテストサイトであり，2019 年時点での登録者は 6 万人以上，問題数は 6,000 問以上の世界最大規模の OJS である．問題は番号とアルファベットによってまとめられている．たとえば，問題番号 1 には A から C の 3 問が含まれており，A が最も簡単で B，C と進むにつれ複雑な問題となる．問題によっては J まで番号が振られている場合もあり，コンテスト中でない限り，ユーザは A 問題のみに着手する，特定の問題セットに着手するなど自由に問題を選択できる．

Codeforces における採点方式は，他の OJS 同様，実行時間およびメモリ使用の制限のもと，複数のテストケースが実行されることによる自動採点で行われる．Codeforces の問題は，英語あるいはロシア語による問題文，入力の仕様，出力の仕様，サンプル入力，サンプル出力，問題のヒントで構成される．問題に対するユーザの提出履歴は，問題ごとに，提出 ID，提出時間，提出者 ID，問題 ID，使用言語，正誤判定結果，CPU 使用量，メモリ使用量，コードサイズで構成される．また，多くの OJS において API が提供されており，Codeforces もユーザ情報や問題の情報など，多くのデータが API によって取得可能である[*5]．

Codeforces の他の特徴として，問題に難易度とタグが付与されている．Codeforces では問題の難易度を 800 から 3500 までの 100 刻みの値で表し，問題には 37 種類のタグのどれか 1 つ以上が付与されている．Codeforces の利用者はこれらの情報を活用して問題を選択することが可能である．

2.3 OJS のプログラミング学習への活用事例

OJS はソースコードの採点を自動化し素早くフィードバックできるため，OJS を教育に用いる研究も報告されている [4], [9]．松永らは大学のプログラミング演習の講義において，OJS の中から教員が指定した問題を学生が解答するように指示した [4]．そして，学生の問題に対する正答数や正答するまでに間違えた回数，正答率などで学生のプログラミング能力および，プログラミング能力獲得の過程を分析した．彼らの分析の結果，OJS はソースコードを自動評価するため，学生の人数が増えても教師の負担が増えにくいという利点があるが，単純に OJS を使用するだけではプログラミング能力が向上した学生は一部にとどまることが判明した．

また，多くの OJS には API が備わっており，API を利用したデータの活用も行われている [5], [10], [11], [12]．則行は OJS の解答履歴から項目反応理論を用いて上級者を特定し，上級者のみが解答している問題を特定する手法を提案した [5]．海外においても，Pereira らによる，OJS を利用したプログラミング演習における学生のドロップアウトの予測が行われている [11]．Pereira らはオンラインのプログラミング演習に学習用の OJS を用い，学生のソースコードの記述量や正答率，OJS へのアクセス時間などのデータから，機械学習によりドロップアウトする学生の特徴を調査した．

以上より，OJS をプログラミング教育の一環として導入することで教員の負担を軽減し，学生のプログラミング学習の支援が可能である．一方で，プログラミング学習に対する学生のモチベーションを維持するために，OJS のデータの分析結果を基に，学生へ与える課題は慎重に選択する必要がある．

2.4 OJS をプログラミング学習で用いる際の問題点

国内外問わず，どの OJS においても非常に多くの問題が掲載されている．各問題は難易度順に掲載されているとは限らないため，ユーザは問題文や入出力例から，その問題が自身のプログラミングに対する理解度で解くことができるか判断する必要がある．先述したとおり，Codeforces は問題選択の支援として，各問題に難易度やタグが付与されている．しかし，たとえば難易度が 1000，タグが implementation の問題は 130 問以上あり，これらの情報のみで学習者が次に解く問題を選択することは困難であると考えられる．したがって，Codeforces を含む OJS をプログラミングの自学自習へ導入するためには，ユーザに対し問題選択の支援を行う必要がある．

則行や中川らは，OJS においてユーザの提出したソースコードを基に，次に解くべき問題を計算し提供している [5], [13]．しかしながら，ユーザが与えられた問題を解くことができるかは，そのユーザのプログラミングやアルゴリズムに対する理解度に依存する．ユーザのプログラミングやアルゴリズムに対する理解度は，提出された 1 つのプログラムからは測ることができず，それまでにどの問題へ着手し，成功，あるいは失敗したのか，問題へ取り組ん

[*3] https://www.topcoder.com/
[*4] https://codeforces.com/
[*5] https://codeforces.com/apiHelp

だ過程を見る必要がある．言い換えると，多くのユーザが短い期間で解き進めることができた問題群は特定の理解度において解答可能であり，たとえ同じ難易度でも連続で失敗，あるいは解答が提出されない問題群は異なるプログラミング能力を求める可能性がある．すなわち，ユーザの解答履歴を分析することで問題間の関係を調査でき，最終的にはユーザの理解度に応じた適切な問題を提供することにつながると考えられる．

以上より本研究では，OJS を用いたプログラミングおよびアルゴリズム学習支援を目的に，そのための第 1 段階として，ユーザの解答履歴を基に問題選択のための問題推定可能なモデルの提案と，問題間の関係を調査する．

3. 提案手法

3.1 概要

OJS における問題どうしの関係性を求めるため，本研究ではユーザの解答履歴に着目する．ここで，ユーザの解答履歴は，そのユーザのプログラミング能力により変化することが考えられる．一方，本論文ではユーザの解答履歴から求めた問題間の関係が，プログラミング学習に有効か調査することを目的とするため，対象ユーザのレーティングや Codeforces の経験歴は考慮せずに扱う．対象ユーザのプログラミング能力を反映したモデルに関しては，今回の実験結果を基に今後設計したいと考える．

研究の概要を**図 1** に示す．以下，各 Step について詳細を述べる．

Step0 では先行研究 [7] において，Codeforces より API などを用いて，A 問題の 1 から 100 に着手したユーザ情報や各ユーザの解答履歴，ソースコードなどを取得した．2.2 節で述べたとおり，A 問題は問題セットのうち最も難易度が低い問題である．ここで，A 問題のみに注目した理由は，各問題セットにおいて A 問題に着手したユーザ数が最も多く，後述する機械学習を用いてモデルを構築するためのデータ取得に適していると考えたからである．さらに，本研究ではユーザの解答履歴，すなわち問題の遷移に注目するため，多くのユーザが遷移した問題セットを対象にする必要がある．加えて，連続する番号の問題は，そのユーザの知識に関係なく連続で着手される確率が高いと考えた．以上をふまえ各問題を分析した結果，本研究では**表 1** に示す 11 問を対象とした．

Step1 において，対象データ 11 問のうち，8 問以上 10 問以下で問題に着手したユーザを対象と見なし，事前に取得した Codeforces のデータセットから，ユーザ情報と着手した問題番号および正誤判定に関するデータ抽出を行った．データを収集する際，11 問すべてに着手したユーザを対

表 1 対象の課題
Table 1 Target problems in Codeforces.

問題番号	タイトル	難易度	タグ
A3	Shortest path of the king	1000	greedy, shortest paths
A7	Kalevitch and Chess	1100	brute force, constructive algorithms
A13	Number	1000	implementation, math
A17	Noldbach problem	1000	brute force, math, number theory
A37	Towers	1000	sortings
A43	Football	1000	strings
A53	Autocomplete	1100	implementation
A56	Bar	1000	implementation
A74	Room Leader	1000	implementation
A88	Chord	1200	brute force, implementation
A90	Cableway	1000	greedy, math

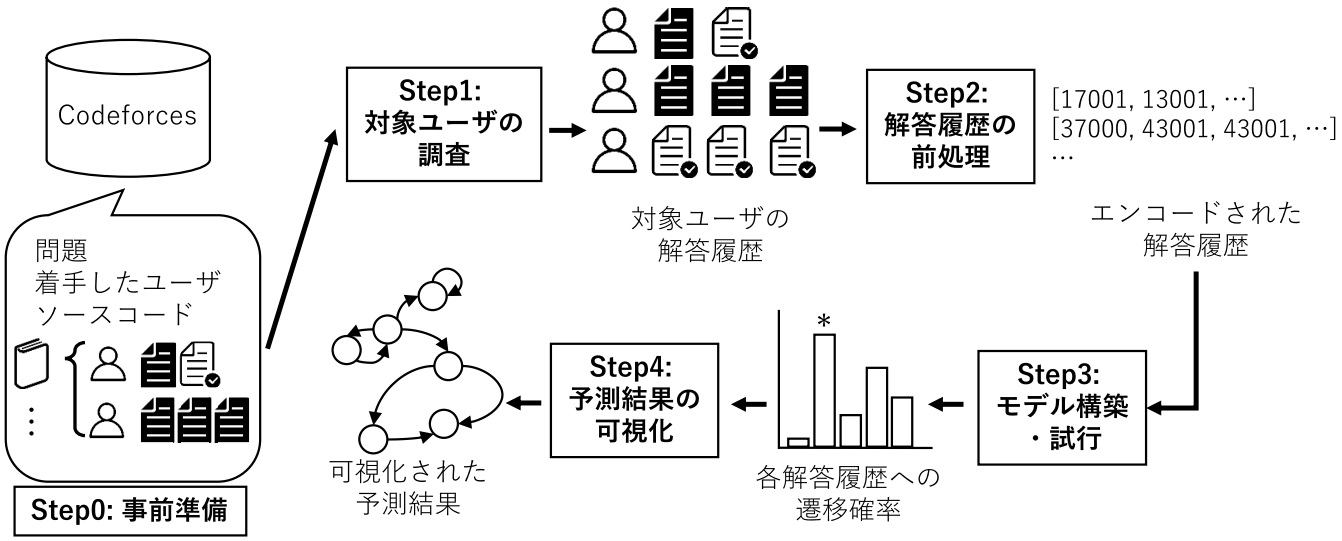


図 1 提案手法の概要
Fig. 1 The overview of our methodology.

象にした場合，プログラミング能力やアルゴリズム理解度に関係なく順に解き進めていくユーザが多かったため，11問すべてに着手したユーザを対象外とした．また，問題間の関係を示すモデルを構築するためには，なるべく多くのユーザが遷移し，かつ多くのデータが必要となるため，10問に着手したユーザのみを対象にせず，8問以上の問題に着手したユーザまでを対象とした．一方，問題間の関係性を調査するためには，より多くの問題に遷移しているユーザや長期的にCodeforcesに取り組んでいるユーザ，データ収集期間中にレートが向上したユーザなどユーザの性質を考慮する必要がある．本論文では提案手法が有効であるかを示すためモデルを生成する必要があったため，不連続かつ様々なタグ，難易度を持つといった問題の性質に着目しデータを収集したが，ユーザの性質を考慮したデータの使用に関しては，実験結果を基にデータの収集を進めていきたいと考える．

入力に使用するデータは，2020年12月1日から2020年12月14日の期間に取得可能であった全ユーザのうち，上述した条件を満たすユーザの解答履歴を対象とした．その結果，対象者は552名となり，1人あたりの問題解答平均数は16.2回，平均正答数9.7回，平均誤答数6.5回となった．

次にStep2において，各ユーザの解答履歴と問題への正誤が推定可能な系列データへのエンコードを行う．具体的には各解答履歴を4桁か5桁の文字列へ変換する．文字列は先頭から，問題番号が最大2桁，問題種別が2桁，正誤を示すフラグが0か1の1桁で構成される．なお，問題種別は今回はA問題のみを扱っているため，すべて00で統一する．具体的には，問題A3に誤答した場合は3000，問題A17に誤答した場合は13001と示す．本調査では表1に示すA3からA90を対象にしたため上記の変換を用いたが，より桁数の多い問題を扱う場合は先頭の問題番号を示す桁数を変更することで対応可能である．

そして，Step3において各履歴を次節に示す流れで学習し，解答履歴の遷移モデルを作成した．最後に，Step3で作成したモデルによって得られる各解答履歴への遷移確率を，Step4において可視化し，プログラミング学習における有効性について検証した．

3.2 生成モデル

前述のとおり，本モデルでは1人の解答履歴を数字の並びとしてとらえる．すなわち，本分析で扱う解答履歴は，着手した問題番号および正誤判定の0か1からなる22種類と，パディング用の0を加えた合計23種類からなる．

まず，訓練用データ生成のため，各履歴のすべての数値を0から22の整数値として扱えるように整形する．次に，計算において扱いやすくするため，各自の履歴を60にパディングし訓練を行う．訓練時は**図2**に示すとおり，1つの訓練シーケンスの長さを5とし，ターゲットとして1つずらしたデータを与える．これらはディープラーニングを用いたオンライン学習サイトにおける学生の正誤判定の予測や，一般的なテキスト生成における出現文字の予測に使われる手法である[14], [15], [16].

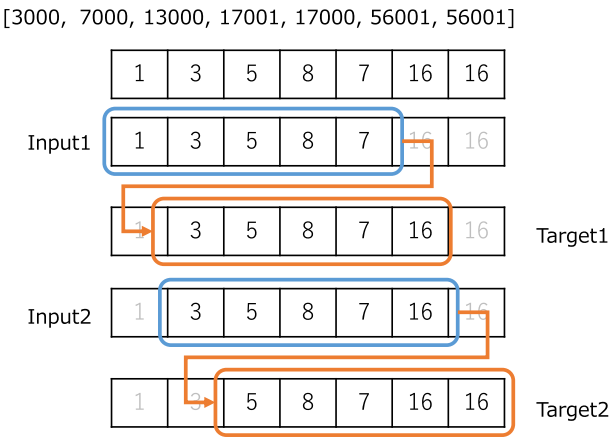


図2 入出力データ
Fig. 2 Input and output data.

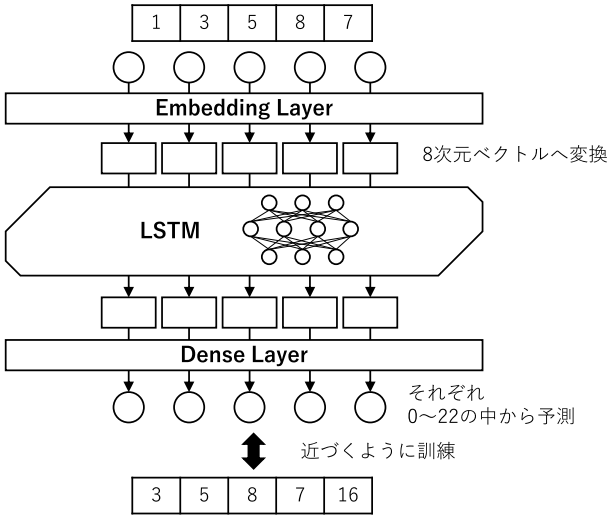


図3 モデルの概要
Fig. 3 The overvoew of model.

作成したモデルの概要を**図3**に示す．モデル生成にあたり，埋め込みベクトルの次元の数は8，隠れ層のユニットの数は512，バッチサイズは32とした．中間層における予測モデルの生成にはLSTM（Long Short Term Memory）を使用した．機械学習における時系列予測の他の有名なモデルとして，GRU（Gated Recurrent Unit）やRNN（Recurrent Neural Network）が存在する．LSTMの特徴として，長期依存するデータの学習が困難であるといったRNNにおける勾配消失問題を解決するモデルであり，一方で計算に多くのリソースを必要とするため，高精度であるもののGRUより低速であるといわれる．また，LSTMは入力に複数の時系列データのベクトルを与えることも可能である．本論文では問題番号と正誤判定を1つの文字列として入力データを作成したが，LSTMを用いることで，問題番号と正誤

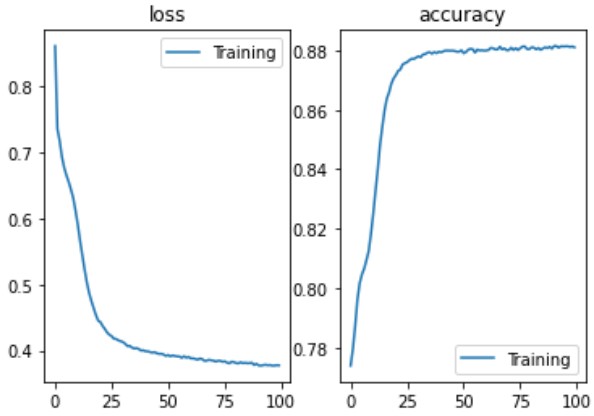


図 4 訓練データの損失関数と正解率の遷移
Fig. 4 Accuracy and Loss of training data.

判定，さらに各問題の難易度や着手した期間など，柔軟な入力データの生成が可能となり，対象のユーザのプログラミング能力や理解度を多面的に反映した問題推薦につながると考える．本論文では 11 問の問題を対象に実験を行ったが，今後は長期的に様々な問題へ取り組んでいるユーザも対象にしたいと考える．以上の LSTM の特徴および本研究の展望を考慮し，本研究では LSTM をモデルとして採用した．

損失関数には交差エントロピー誤差，活性化関数にはシグモイド関数を用い，学習率は 0.01 である．また，出力層にソフトマックス関数を用いた．ソフトマックス関数とはその層から出力されたすべての値に指数関数をかけることで正の値に直し，その和を 1 に正規化するものである．すなわち，ソフトマックス関数を通すことで各出力値，本実験では次に着手する問題とその正誤の確率を求めることができる．なお，本研究ではすべての解答履歴を訓練用データとして用いた．

3.3 結果

3.2 節で述べたモデルを作成し，3.1 節で述べたユーザの解答履歴を用いて訓練を行った．解答履歴は 23 の 5 乗，すなわち約 650 万通り存在しうるが，本実験で扱ったユーザが実際に遷移した，連続する 5 回の解答履歴で一意のものは 6,510 通りであった．本実験では 5 つの連続した履歴，各問題および正誤判定による 23 の特徴量，3 層の単純な LSTM を基にしたモデルを用いたが，対象データの特徴を加味した適切な学習モデルの見極めは今後の課題とする．

図 4 に 100 エポックを使用して訓練した後の損失関数と正解率の遷移を示す．損失関数，正解率ともに収束しており，訓練が正しく行われたことが分かる．訓練終了後，先述した 6,510 通りの連続した 5 つの学習履歴をモデルに与え，次にどの解答履歴へ遷移するかカウントした．同様の作業を 10 回繰り返した後，各履歴への遷移回数を 10 で割り遷移確率を求めた．次に，実際の解答履歴も同様に，連続した 5 つの学習履歴の次にどの解答履歴へ遷移したか確率を求めた．そして，両データともに 10%以上の確率で遷移する解答履歴を遷移先と見なし，実際のデータとの比較を行った．検証した結果を**表 2** および**表 3** に示す．表 3 において各評価項目約 98%となっており，訓練が正しく行われたことを示す．

表 2 予測した解答履歴の精度検証
Table 2 Thread score of prediction result.

		実際の値	
		Positive	Negative
予測値	Positive	6,461	260
	Negative	3,257	139,729

表 3 作成したモデルの精度検証結果
Table 3 The result of verification.

Accuracy	Precision	Recall	F 値
0.976507	0.998143	0.977222	0.987572

4. 評価

可視化による定性的評価

作成したモデルの結果に対する有効性の評価のため，モデルへ解答履歴を与え，予測された結果を可視化した．結果の可視化には折れ線グラフやサンキーダイアグラムなど様々な手法が考えられるが，本評価では各解答履歴をノード，解答履歴間の遷移確率をエッジととらえ，ネットワーク図を作成する．Graphviz [*6] より作成したネットワークを**図 5** に示す．ノード上の数値は問題番号，T は正答，F は誤答を示す．解答履歴を 1 問モデルへ与えた際の予測結果から，各問題への遷移確率と正答誤答の確率をソフトマックス関数により求め，遷移確率が 10%以上の場合，エッジとして抽出した．エッジの長さ，すなわちつながっているノード間の距離が短いほど遷移確率が高いことを示す．ノード間において相互に遷移する場合は双方距離の平均値とした．また，遷移先が多いノードほど，ノードのサイズを大きく表現した．たとえば，37T からは 43T，43F，74T，37F の 4 つのノードへ遷移するが，88F は自己遷移のみである．すなわち，サイズが大きいノードほど，様々な問題へ着手するための中継として扱える可能性が高いことを示す．今回の可視化では一部のノードとエッジが重なっているため，可読性の高い可視化方法に関しては今後の課題とする．

表 1 と比べ，タグが一致する問題でも相互に遷移しない場合がある．問題番号 A3，A90 と A7，A90 は同様の難易度であり，同様のタグ “greedy（貪欲法アルゴリズム）” や “math” を持つが，相互の遷移確率は 10%以下であり，タグが異なる問題への遷移確率の方が高いことが分かる．問

[*6] https://graphviz.org/

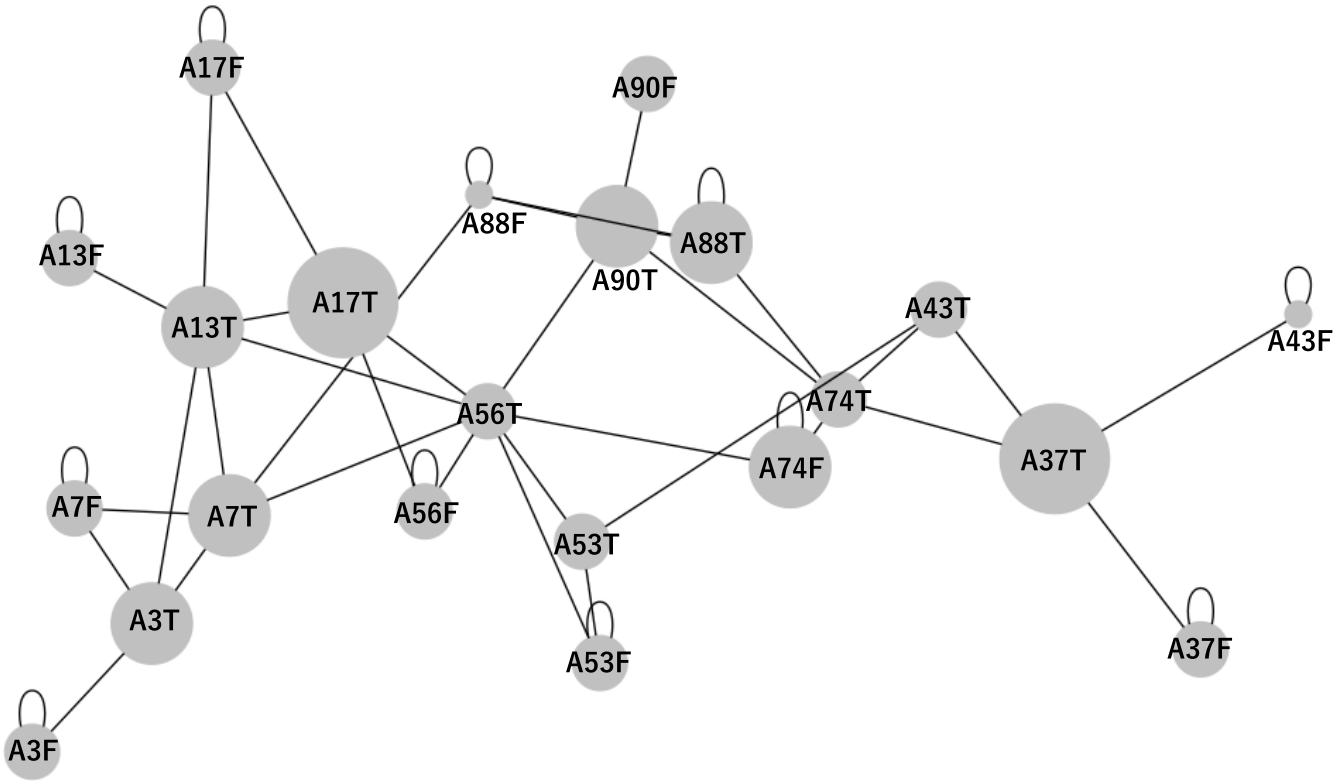


図 5 解答履歴をもとにした問題ネットワーク（T：正答，F：誤答）
Fig. 5 Network of submitted history model (T: True, F: False).

題番号 A13，A56，A74 も同様のタグ “implementation” を持ち，A56 には A13 と A74 からの遷移は存在するが，A13 と A74 間にエッジは生成されなかった．以上より，タグと難易度が同一のものでも，連続して解くことが可能であるとは限らないことが分かる．そこで，本提案手法において解答履歴を用いることで，問題全体を俯瞰でき，タグや難易度に依存しない問題の選定が可能となる．

また，解答履歴の可視化からユーザへの問題の推薦も可能である．例としてユーザが A37 を解くことができ，次に着手する問題を探していた場合を想定する．今回対象とした問題において，A37 より難易度が高い問題は A7 か A53 が存在する．図 5 より，A37 から A53 へ遷移するルートは，A37，A43，A53 の 2 ホップや，A37，A74，A90，A56，A53 の 4 ホップなどが存在することが分かる．すなわち，ユーザが A74 に着手して誤答が続いたとしても，A43 を推薦することで，より短いホップ数で難易度の高い問題へ導くことができる．今回は 11 問の A 問題のみを対象として扱ったが，問題数を増やすことで，徐々に高い難易度の問題に挑戦できる．加えて，様々なタグの問題にも着手できる遷移を選ぶことで，ユーザのプログラミングやアルゴリズムに対する理解度を向上させる問題の推薦が可能になると考える．

以上より，本提案手法はユーザに対し，タグや難易度に依存しない柔軟な問題推薦を可能とする．加えて，ユーザが解くことができなかった問題に対しても，将来的に挑戦できるような問題の推薦も可能であり，ユーザのプログラミングや OJS の継続的な利用に対するモチベーションの維持にもつながると考える．したがって，ユーザの解答履歴を用いたモデルの作成は，OJS を用いた自学自習に有効であると考える．

既存手法との比較

中川らの研究では，OJS においてユーザが提出したソースコードに類似し，かつ提出ソースコードよりプログラムの品質が高い他のユーザの提出したソースコードを提示することで，ユーザが自身のソースコードの品質を漸進的に向上させることを目的としている [13]．また，ユーザへソースコードを提示する際，ユーザの解答履歴に基づいた遷移グラフを構築し，ソースコードの選択を行っている．

上記の問題選択を行うために中川らが提案した重み付き遷移グラフは，前後 2 つの解答を対象に，誤答から正答へ遷移した場合と，正答から正答へ遷移した場合の遷移数の合計より構築される．一方，本研究では 5 件の解答履歴を基に次に着手する問題の予測を行い，解答履歴を基にした遷移予測モデルを構築した．

Codeforces を含む多くの OJS において，ユーザは問題選択の際，問題文だけではなく，タグや難易度，着手したユーザ数や正解率など，複数のデータを参照することができる．実際に本文中の表 1 の問題において，ユーザが A56 に正答した次へ着手する問題の調査を行った．その結果，A56 に正答後の遷移で最も多い問題は A74 であるものの，

A17，A37，A43，A53，A56 のように順調に解き進めてきたユーザは A74 へ着手し，A56 に複数回失敗したユーザは A56 を解きなおす，さらに A43，A90，A13，A17，A56 のように順番に関係なく同難易度の問題に着手してきたと思われるユーザは A3 や A53 に着手するなど，同様の問題に正答した後，次にどの問題を選択するかはユーザによって異なることが分かった．すなわち，ユーザはそれぞれ問題選択の方針を持っており，問題を推薦する際はユーザの問題選択の方針を考慮しつつ，ユーザのプログラミング能力を向上させるような問題を推薦することが望ましいといえる．

遷移数の多い問題を優先的に推薦する場合，ユーザのその問題に着手するまでの解答履歴が失われるため，当該ユーザの問題選択の方針を無視した問題を推薦する恐れがある．一方で，同様の問題に着手した後，それまでのユーザの解答履歴，すなわち各ユーザの問題選択の方針を考慮し，ユーザによって異なる問題の推薦を可能にする点が本研究の新規性であると考える．

また，表 1 に示す問題において，遷移の数を基にした遷移グラフを作成し，遷移数が多い問題へ遷移する場合と比較を行った．遷移率が 5%以上の遷移を正解として扱った結果，すべての解答履歴を対象にすると約 61.7%の解答が，先行研究と同様に正答から正答への遷移および誤答から正答への遷移のみを扱った場合は約 40.0%のみの解答が正解として検出された．すなわち，上述のとおりユーザは様々な問題選択を行っており，また，正答へつながる問題選択はユーザにとって困難であることが分かる．

さらに，本提案と同様に 5 つの解答履歴を基に，次の遷移先を遷移数から予測を行った．2 つ以上の遷移先が存在する 548 件の解答履歴を対象に，正誤関係なく最も遷移数が多い遷移先を正解として扱った結果，Accuracy 0.95，Precision 0.60，F 値 0.76 を得ることができ，いずれも本提案手法の方が高い精度を示す．

以上より，本研究では既存研究で扱っていなかった誤答から誤答への遷移や，正答から誤答への遷移も対象にモデルを作成しており，解答履歴を基にユーザが次に解く問題およびその正誤を予測した結果，単純に遷移数が多い問題を推薦する場合より高い精度を示した．したがって，様々なユーザの多様な解答履歴を考慮した問題を推薦するにあたり，本提案手法は有用であると考える．

5. 考察

5.1 妥当性に関する議論

本論文の目的は，OJS におけるユーザの解答履歴から作成したモデルが，OJS を用いた自学自習に有効であるかを検証することである．4 章より，OJS の問題をユーザに推薦する場合，OJS で使用されるタグや難易度だけでは問題の選定が難しく，ユーザの解答履歴を基に問題間の関係を求め可視化する手法は有効であることが分かった．また，ユーザの解答履歴は他の OJS においても，閲覧，あるいは API などで取得可能である場合が多い．本提案手法ではソースコードを取得する必要もないため，言語に依存しない分析も可能であり，他の OJS においても広く適用可能である．

一方で本研究で構築したモデルにはいくつかの妥当性の問題が存在する．まず，モデルへの入力データに関する内部妥当性が考えられる．先述のとおり，本論文ではまずユーザの解答履歴を基に解答履歴の遷移モデルの作成を目指した．したがって，取り組んだユーザ数が多く，同一ユーザが頻繁に遷移する問題を対象に調査した結果，表 1 の 11 問を対象とした．しかしながら，この対象の問題で，ユーザがほかに着手した問題がどれほど存在するかは考慮していない．たとえば，本調査において問題番号 A3 の次に A7 に着手したユーザでも，A3 の次にすぐに A7 へ着手したユーザと，A4，A5，A6 にも着手してから A7 へ取り組んだユーザもいる．ただし，3.1 節で述べたように，対象の問題を選択する際に，多くのユーザが頻繁に遷移する問題を選んだため，本調査で得られた問題間の関係について一定の妥当性は確保されると考える．一方で，ユーザが様々な道筋で問題を解き進められる問題推薦を行うためにも，今後は対象の問題を増やして調査する必要がある．

同様に，3.1 節で述べたとおり，使用したデータセットに関しても内部妥当性が存在する．今回，データ取得期間中に取得できた条件を満たすすべてのユーザのデータを使用したが，ユーザにプログラミング能力が向上する，あるいは継続的に Codeforces を利用するモチベーションが続くような問題推薦を行うためには，それらに該当するユーザのデータを使用することが望ましい．今回使用したデータには Codeforces を長期的に利用しているユーザもいれば，A 問題の一部のみにしかほぼ着手していないユーザも含まれていた．一方，本データセットには該当の問題の約 8 割に着手したユーザを対象としており，また長期的かつ多くの問題に着手できなかった理由として，適切な問題選択ができなかったことも理由として考えられる．今後は Codeforces におけるユーザの問題投稿数や各レート帯の人数，継続期間などを分析したうえで，ユーザの性質も考慮したデータセットを構築したいと考える．

外部妥当性として，提案手法は既存ユーザの解答履歴を用いることから，既存ユーザが着手していない問題や新規に追加された問題への対応が困難である点があげられる．Codeforces における A 問題は他の問題に比べ多くのユーザが着手する傾向があるため，Codeforces 利用初期段階における問題推薦としては本提案手法は有効であると考える．一方，E 問題以上の着手するユーザが少なく難易度も高い問題や，新規に追加された問題に対応するためには，問題文やタグ，実行時間の制約など，問題固有のデータを

利用した問題間の関係性についても調査を進める必要があると考える．

構成概念妥当性への脅威として，入力データや特徴量の設計があげられる．3.1 節で述べたとおり，本調査では対象としたユーザのプログラミング能力や Codeforces への取り組み頻度，すなわちレートの情報などを入力データへ与えていない．ユーザのプログラミングやアルゴリズムに対する理解度を考慮した問題推薦を行う場合，短期間でレートが上がったユーザや，長期的に Codeforces に取り組んでいるにもかかわらずレートや正答率が伸びないユーザなど，ユーザの特性も考慮しモデルを訓練する必要がある．また，本論文では取得した解答履歴すべてを訓練に用いたため，新規に対象の問題へ着手したユーザの解答履歴を用いた検証も必要である．以上より，着手した問題と正誤判定のほかにどのような特徴量が考えられるか，モデルの適切なパラメータや設計，他のモデルとの比較も行いながら，今後実験を重ね見極めていきたいと考える．

5.2 教育現場への応用

本提案手法により，プログラミング学習の自学自習に OJS を利用することが可能になると考える．今回は対象となった問題全体の関係性を俯瞰できるように，図 5 に示すネットワーク図を作成した．本提案手法では問題に誤答した際の解答履歴の遷移に関しても予測を行っている．したがって，特定の問題を解くことができなかった場合，どのような道筋で解きなおしていくことで，以前解くことができなかった問題に到達するか，学習者に気づきを与えることができると考える．問題に対して，OJS が与える採点結果とは異なる気づきを与えることで，学習者のプログラミングや OJS の利用に対するモチベーション維持につながると考える．

さらに，特定の問題へ遷移するための最適なホップ数だけでなく，様々な問題へ遷移することができるノードも判明した．実際にプログラミング学習の一環として本提案手法を利用する場合，学習者のモチベーションを保つためには少ないホップ数で難易度の高い問題へ着手できることも重要だが，様々な問題へ触れることでプログラミング能力やアルゴリズムに対する理解度の向上にもつながると考える．そのためには，図 5 における A17 や A37 のような，様々な問題へ遷移する問題を推薦する必要がある．どのような問題を優先して推薦すべきかについては，長期的に OJS へ取り組んでいるユーザや高レートのユーザの分析を通して調査していきたいと考える．

6. おわりに

本論文では，プログラミング教育における自学自習の支援として OJS に着目した．OJS から次に解く問題をユーザへ推薦するため，ユーザの解答履歴に基づいた問題間の関係を求めた．11 問の問題に対する 552 名のユーザの解答履歴を LSTM を用いて学習し，問題の遷移確率モデルを作成した．モデルを可視化したところ，遷移確率の高い問題群だけでなく，難易度の高い問題へ着手するための問題の遷移や，様々なアルゴリズムの種類へ着手できる問題などが分かった．これらの情報は，プログラミング教育における自学自習において有用な情報である．

今後の展望として，以下 2 点が考えられる．

適切な入力データの選定

モデルへ与える特徴量の選定や，使用する連続した解答履歴の件数の調整があげられる．また，モデルの予測結果を用いることで，問題を解き進めていった際のレートの変動といったシミュレーションも可能となる．ユーザへ適した問題を提供するために，モデルの精度を高めるための設計に加え，作成したモデルよりユーザへどのような情報を与えるか，被験者実験を通し，適切なシステム UI もあわせて検討していきたいと考える．

被験者実験

本研究のゴールは，ユーザに適した問題を自動推薦することで，OJS を用いた自学自習を継続的させつつ，ユーザ自身のプログラミング能力やアルゴリズム理解度を向上させることである．したがって，次の段階として，構築したモデルを基にユーザへ推薦した問題が適切であったか，あるいは問題推薦の有無によるユーザの問題選択の傾向や継続的に OJS を利用可能かなどについて調査する必要がある．そのためにも，今後はまず Codeforces における長期利用ユーザやレートが向上したユーザの特徴について調査し，データセットに用いるユーザについて検討したのちに，短期的，長期的にユーザに与える影響について調査していきたいと考える．

謝辞 本研究は JSPS 科研費 20K14101，21K12097 の助成を受けたものです．

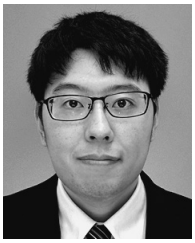
参考文献

[1] 総務省：プログラミング人材育成の在り方に関する調査研究，入手先 ⟨https://www.soumu.go.jp/menu_news/s-news/01ryutsu05_02000068.html⟩.
[2] 國宗永佳，仲林 清：プログラミング導入演習に対して学習支援システムと自己調整学習が与える影響の分析，電子情報通信学会通信ソサイエティマガジン，Vol.13, No.2, pp.100–109 (2019).
[3] 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).
[4] 松永賢次：導入プログラミング教育におけるオンラインジャッジシステムの活用の試み，情報科学研究，Vol.31, pp.25–41 (2011).
[5] 則行祐作：オンラインジャッジ上のプログラミング学習者の成長分析，奈良先端科学技術大学院大学修士論文 (2017).
[6] Codeforces Rating System, available from ⟨https://codeforces.com/blog/entry/102⟩ (accessed 2021-01-10).
[7] 池田太郎，槇原絵里奈，小野景子，米田浩崇：オンラインジャッジシステムにおける問題の関連性調査，Vol.120,

No.231, pp.27–29, 電子情報通信学会 (2020).
[8] Ebtekar, A. and Liu, P.: An Elo-like System for Massive Multiplayer Competitions *CoRR* (2021).
[9] Yera Toledo, R., Caballero Mota, Y. and Martínez, L.: A recommender system for programming online judges using fuzzy information modeling, *Informatics*, Vol.5, No.2, p.17, Multidisciplinary Digital Publishing Institute (2018).
[10] Amirabbas, M., Mojtaba, V.-A., Alireza, K., Ahmad, B.-D. and Bahman, Z.: Code4Bench: A multidimensional benchmark of Codeforces data for different program analysis techniques, *Journal of Computer Languages*, Vol.53, pp.38–52 (2019).
[11] Pereira, F.D., Oliveira, E., Cristea, A., Fernandes, D., Silva, L., Aguiar, G., Alamri, A. and Alshehri, M.: Early dropout prediction for programming courses supported by online judges, *Proc. International Conference on Artificial Intelligence in Education*, pp.67–72 (2019).
[12] Rastogi, I., Kanade, A. and Shevade, S.: Active Learning for Efficient Testing of Student Programs, *International Conference on Artificial Intelligence in Education*, pp.296–300 (2018).
[13] 中川尊雄，藤原　新，畑　秀明，松本健一：プログラミング学習者向けソースコード提示システム TAMBA，ソフトウェアエンジニアリングシンポジウム 2016，Vol.2016, pp.34–41 (2016).
[14] Piech, C., Bassen, J., Huang, J., Ganguli, S., Sahami, M., Guibas, L.J. and Sohl-Dickstein, J.: Deep knowledge tracing, *Advances in Neural Information Processing Systems*, Vol.28, pp.505–513 (2015).
[15] 中川大海，那須野薫，岩澤有祐，上野山勝也，松尾　豊：Deep Knowledge Tracing の拡張による擬似知識タグの生成，人工知能学会論文誌，Vol.33, No.3, pp.1–11 (2018).
[16] Yu, W., Zhu, C., Li, Z., Hu, Z., Wang, Q., Ji, H. and Jiang, M.: A Survey of Knowledge-Enhanced Text Generation, arXiv:2010.04389 (2020).

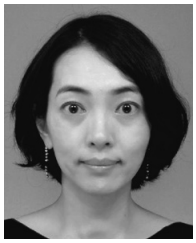
槇原 絵里奈 （正会員）

2013 年大阪工業大学情報科学部情報システム学科卒業．2018 年奈良先端科学技術大学院大学博士後期課程修了．博士（工学）．同年より同志社大学理工学部インテリジェント情報工学科助教．ソフトウェア工学教育，プログラミング教育支援に関する研究に従事．電子情報通信学会，日本ソフトウェア科学会各会員．



池田 太郎

2019 年同志社大学理工学部インテリジェント情報工学科卒業．2021 年同志社大学大学院理工学研究科修士課程修了．在学時，オンラインジャッジシステムにおける問題の関係性調査の研究に従事．



小野 景子 （正会員）

2007 年同志社大学大学院工学研究科博士課程修了．博士（工学）．2009 年同志社大学研究開発推進機構省エネルギー照明システム研究センター特定任用研究員（助教）．2010 年龍谷大学理工学部電子情報学科助教．2014 年龍谷大学理工学部電子情報学科講師．2020 年同志社大学理工学部インテリジェント情報工学科准教授，現在に至る．並列処理，最適設計，進化計算等の研究に従事．IEEE，進化計算学会各会員．

新濱 遼大

2021 年同志社大学卒業．現在，同志社大学理工学研究科博士前期課程在学中．オンラインジャッジシステムにおける機械学習を用いた問題選択支援の研究に従事．