Vol.2025-CE-178 No.4
2025/2/15

プログラミング学習における無操作状態に着目した支援タイミング決定手法の提案

野原 広大[1,a)]　石田 繁巳[2,b)]　稲村 浩[2]

概要：プログラミング初学者は，漠然と問題を捉えて理解が進まない「初歩のつまづき」に陥りやすい．本研究では，初学者に対して学習支援の提供が可能なバーチャル TA システムの実現を目指している．学習者のキーボードおよびマウスの操作が行われていない「無操作状態」に着目し，支援対象および支援タイミングを決定するシステムを設計した．提案手法では，学習者の無操作状態から特徴量を抽出し，指数分布に基づいて無操作頻度パラメータ λ を算出する．λ を用いて学習者を分類し，支援が必要な対象を特定する．さらに，最大無操作時間を検出し，これをもとに支援タイミングを S 秒後に決定する手法を提案した．評価実験では，被験者 25 名の行動データを用いて，支援対象決定精度と支援タイミング誤差を検証した．その結果，支援対象決定の F1 スコアは 0.67，支援タイミング誤差の最小値は 564.27 秒であり，提案手法の有効性を示した．

キーワード：能動的支援，　バーチャル TA，　行き詰まり状態

1. はじめに

様々な分野で IT (Information Technology) 技術や AI (Artificial Intelligence: 人工知能) 技術の活用が進み，これらを活用・開発するために必須となるプログラミング教育の重要性は飛躍的に高まっている．初等教育からプログラミング教育を施す IT 人材育成も重要視されており，文部科学省は 2020 年度，2021 年度からそれぞれ小学校，中学校でのプログラミング教育を必修化し，2022 年度からは高等学校で情報 I を必修化した [1]．

プログラミング教育では，特に初学者に向けた支援が重要である．多くの場合において，プログラミング教育は学びながら実際にプログラムを記述する演習形式がとられている．プログラミング初学者はプログラミングに関する知識が乏しい状態で問題文からどのようなプログラムを作成するかを考える必要があり，学習の障壁が高い．このため，「何らかのエラーが出た」「コンパイルはできたけどなぜか動かない」など，漠然と問題を捉えて理解が進まない「初歩のつまづき」状態に陥りやすい．プログラミング初学者に対しては，演習中に質問できる程度まで学習が進むように多くの支援が必要となる．

プログラミング教育の重要性の高まりに比べて教育者の数が不足していることから，これまでにも学習支援に関する様々な研究が報告されている．例えば，アイトラッカや生体センサなどの特別な機材を用いて把握した学習者の状況に基づいてフィードバックを行う手法が報告されている．特別な機材の導入が困難であることも多いため，学習者の提出物を分析してフィードバックを行う手法，連続したコンパイルやコンパイルと実行の繰り返しなどの特定の行動に基づいて行き詰まりを検出する手法なども提案されている．

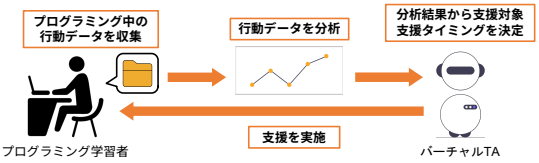

図 1　バーチャル TA システムの概要

しかしながら，これらの手法はプログラミング初学者に対して行う必要のある「初歩のつまづき」に対応できない．「初歩のつまづき」は初学者が課題を提出する前に発生することが考えられるため，提出物に依存する手法での支援は困難である．初学者はコンパイルを行う前に行き詰まる場合も多く，提出前の「コンパイル」という行動での支援も難しい．

筆者らは，プログラミング行動のみに基づいて，「初歩のつまづき」状態から理解を促すための支援を行うバーチャル TA (Teaching Assistant) システムの実現を目指している．**図 1** に，バーチャル TA システムの概要を示す．本システムでは，普段のプログラミング環境に存在する機材

[1] 公立はこだて未来大学大学院 システム情報科学研究科
[2] 公立はこだて未来大学 システム情報科学部
[a)] g2123047@fun.ac.jp
[b)] ish@fun.ac.jp

から取得できる行動データを取得・分析して学習支援を実現する．キーボード操作という行動データからは学習者のコーディング状況や検索状況を確認することができ，マウス操作からはブラウザでの操作やタブの移動が確認できる．これらの操作が行われていない状況は，学習者が考えている状況であることも推定できる．このように推定した学習者の状況に基づき，考え方に対する助言や，ソースコード上の間違いの指摘などの支援を行う．

バーチャル TA システムの実現に向けては，誰に，いつ支援を行うかを決定することが重要となる．「初歩のつまづき」は自力での解決に膨大な時間を要するため，「初歩のつまづき」状態に陥いる，あるいは陥りやすい学習者を特定する必要がある．一方で，「初歩のつまづき」状態からすぐに支援をしてしまうと，学習者自身が何も学習せず，学習指導したとしてもその内容が定着しない．このため，「初歩のつまづき」から学習者がある程度自身で解決を図った後に支援を行うことが重要である．

本稿では，プログラミング学習者がキーボードやマウスを操作していない「無操作状態」に着目して支援対象・タイミングを決定する手法を示す．学習者が「初歩のつまづき」状態に陥ると，エラー解決の手がかりを考える際や外部から情報を収集する際，収集した情報をもとにエラー解決を試みたが解決せずにその理由について考える，エラーの確認や資料を参照する，などの行動により操作を止める時間が生じる．このような「無操作状態」は，学習者が行き詰まっている状況を反映する指標として利用可能である．そこで，一定時間操作のない「無操作状態」の発生を記録し，その発生頻度・発生間隔・無操作状態の長さに基づいて支援対象及び支援タイミングを決定する．

被験者 25 名がプログラミング問題を解いている間に収集した操作データを用いて，提案する支援対象・タイミング決定手法の有効性を検証した．その結果，成績によってプログラミング中の無操作状態の回数や発生頻度に有意な差があることを示し，無操作回数によって支援対象を特定可能であることを示した．無操作状態の頻度をもとに学習者を分類した結果，頻度の低い学習者には最大無操作時間に有意な傾向があり，最大無操作をもとにした支援タイミング決定の可能性を示した．

本稿の構成は以下の通りである．2. では，関連研究として学習支援に関する既存手法を示す．3. で提案する支援対象・タイミング決定手法を示し，4. でその評価を示す．5. で評価結果に関して議論し，最後に 6. でまとめとする．

2. 関連研究

2.1 提出物や収集したデータを用いた研究

プログラミング学習支援に関する研究では，学習者が提出したソースコードなどのデータを収集し，分析することで学習者の状況を推定する手法が報告されている．

長らは，学習履歴を収集・抽出することでフィードバックを返す “proGrep” を提案している [2] が，学習履歴を収集するためにこれまでに起こったエラーなどの事例を収集する必要があり，それらの収集には時間を要する．田口らは，学習者の理解状況や学習意欲を分析し，協調フィルタリングにより出題課題を変更する手法を提案している [3] が，同一の単元で複数の問題を用意する必要があり，問題を作成する教員にとっては負担が大きいことに加え，理解状況を確認するために問題を 1 回解く必要がある．Gupta らは，コードの文法ミスを修正するフレームワークを開発し，エラーを減らす手法を提案している [4] が，提出されたコードしか修正できないことに加え，コンパイルエラーしか対応していないため，コードをどう書くかは学習者自身が考える必要がある．

このように提出物を分析する手法では，分析やデータの収集，準備に多くの時間が必要となることに加え，提出物がない状態では支援を行うことができない．スナップショットを用いて学習者の行き詰まり箇所を特定する手法 [5] も報告されているが，スナップショットを分析した後に支援を行うため，先ほどの研究と同様に，学習者が支援を必要としたタイミングからラグが発生してしまう．

2.2 特定の行動を用いた研究

2.1 節で示した手法に対して，特定の行動を用いた支援手法は，学習中に取得できるデータから学習者の状況を分析し，その場で支援の実施やフィードバックを返すことが可能である．

槇原らは，修正・コンパイル・実行を繰り返す行動に着目することで行き詰まった学習者を検出する手法を報告している [6]．宇野らは，保存回数・コンパイル回数などの 6 つのパラメータから学習者の状況を推定し，学習成果をリアルタイムに通知する手法を報告している [7]．これらの手法ではコンパイルをもとに学習者の行き詰まりを推定するため，学習者がコンパイルする前に行き詰まった場合は支援が行えないことが問題として挙げられる．

浦上らは，ソースコードの編集履歴から行き詰まりを検出する手法を提案している [8] が，ソースコードを複数回収集する必要があり，コンパイルに着目した手法と同様にソースコードを収集した時点で行き詰まっていた場合には支援が行えないことが問題として挙げられる．

2.3 センシングによる学習者状況推定に関する研究

センシング技術やセンサなどの特別な機材を活用することにより学習者の状況推定を行う研究が報告されている．

学習者をセンシングする手法として，倉田らは，感情センサなどの生体センサやアイトラッカを用いてプログラミング中の困惑状況を推定する手法を報告している [9]．江木らは，加速度センサを用いて学習者の筆記行為を検知する手法を提案している [10]．Ishimaru らは，アイトラッキングによって多岐選択問題を解いているときの学習者の自信度を推定し，復習すべき問題をフィードバックする “CoALA” を提案している [11]．これらの手法は，学習者

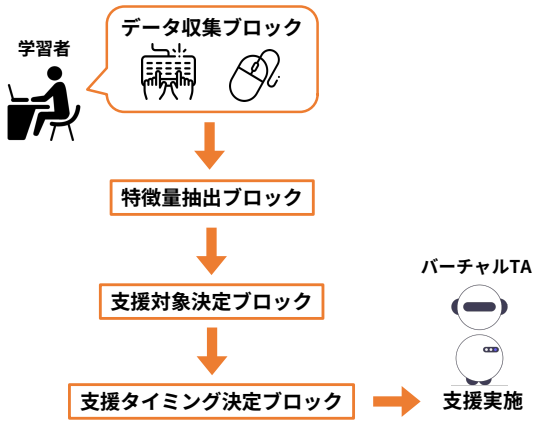

図 2 支援対象・タイミング決定手法の概要

の動作などから状況推定を行うことで早期に行き詰まりを推定することが可能であるが，アイトラッカのような特別な機材を利用しているため，教育現場やオンライン学習での活用は困難である．

3. 支援対象・タイミング決定手法

3.1 アプローチ

支援対象・タイミング決定の基本アプローチは，プログラミング学習者がキーボードやマウスを操作していない「無操作状態」の発生状況が，プログラミング学習者の状況に依存することを利用することである．学習者がプログラミング中に行き詰まった場合，キーボードやマウスを操作せず考える「無操作状態」が発生することが考えられる．無操作状態の発生頻度や時間の長さは，学習者の経験や知識により変化すると考えられる．そのため，無操作状態に着目することで学習者の「初歩のつまづき」を早期に見つけ，支援が可能となる．

3.2 概要

図 2 に，支援対象・タイミング決定手法の概要を示す．本手法は，4 つのブロックから構成されている．データ収集ブロックでは学習者がキーボード・マウスを操作した際のキー押下データ，マウス移動データ，それぞれを操作した際の時刻データを収集する．収集した行動データは特徴量抽出ブロックに入力され，収集したデータから特徴量を抽出する．支援対象決定ブロックでは，抽出した特徴量をもとに初学者などの優先して支援を行う対象を決定する．支援対象の決定に必要な量の行動データが集まった時点で学習者の分類を行う．支援タイミング決定ブロックでは，分類結果から決定した支援対象の行動データを収集し，支援タイミングを決定する．本手法では，支援対象の「最大無操作時間」に基づいて支援タイミングを決定するアプローチを用いる．

以降は各ブロックの詳細について述べる．

3.3 データ収集ブロック

データ収集ブロックでは，学習者がプログラミング中に操作を行うキーボードおよびマウスの操作情報を対象データとして収集する．収集する操作データは，以下の形式のベクトルデータ p_i として記録される：

$$p_i = [t_i, \alpha_i, x_i, y_i]$$

ここで，各パラメータは次のように定義される：

- t_i：操作が発生した時刻
- α_i：操作の種類（“key” または “mouse”）
- x_i：キーボード操作の場合は押下されたキー，マウス操作の場合はカーソルの x 座標
- y_i：マウス操作の場合はカーソルの y 座標，キーボード操作の場合は常に null

収集された行動データをもとに無操作状態データセットを生成し，特徴量抽出ブロックにて使用する．

3.4 特徴量抽出ブロック

特徴量抽出ブロックでは， 3.3 節で収集した操作データを基に無操作状態データセット $\mathcal{D}_{\mathrm{NOS}}$ を生成し，学習者を分類および支援タイミングを決定するための特徴量を抽出する．本研究において最終的に必要な特徴量は以下の 2 つである．

- 無操作時間 τ：無操作状態の継続時間
- ラムダ λ：無操作時間 τ が従う指数分布のパラメータであり，無操作状態の発生頻度を表す

これらの特徴量を計算するプロセスは以下の通りである．

(1) 収集した操作データから無操作状態を検出し，無操作状態データセット $\mathcal{D}_{\mathrm{NOS}}$ を生成する
(2) 無操作状態データセット $\mathcal{D}_{\mathrm{NOS}}$ を基に，無操作時間 τ を抽出する
(3) 無操作時間 τ を用いて指数分布モデルのパラメータ λ を算出する

以下，各プロセスの詳細について説明する．

はじめに，無操作状態の検出とデータセットの生成を行う．無操作状態の定義は「5 秒以上キーボード・マウスによる操作がない状態」とする．具体的には， 3.3 節で収集した行動データに含まれる収集した行動データ p_i の t_i を使用する．その値が 5 秒を超えた場合，当該行動データを無操作状態データセット $\mathcal{D}_{\mathrm{NOS}}$ に格納する．無操作状態データセット $\mathcal{D}_{\mathrm{NOS}}$ は，操作データ p_i の間隔 $t_{i+1} - t_i$ が 5 秒以上を満たす場合に，その開始時刻 t_i と継続時間 $t_{i+1} - t_i$ を格納して構成される．

$$\mathcal{D}_{\mathrm{NOS}} = \{[t_i, \tau_i] \mid \tau_i = t_{i+1} - t_i > 5\}$$
$$(i = 1, \ldots, N-1)$$

ここで，N は無操作状態の回数を表す．$\mathcal{D}_{\mathrm{NOS}}$ から継続時間 τ_i のみを取り出した集合を T とする．

$$T = \{\tau_i \mid [t_i, \tau_i] \in \mathcal{D}_{\mathrm{NOS}}\} \quad (1)$$

最後に，無操作時間 $\tau_i \in T$ を用いて指数分布モデルのパラメータ λ を算出する．算出する手順は以下のとおりである．

(1) 無操作時間 $\tau_i \in T$ のヒストグラムを生成し，各ビンの中心値を計算する．

(2) ヒストグラムと指数分布モデルの確率密度関数 (PDF) の誤差の 2 乗和を最小化するよう，次の目的関数を設定する．

$$E(\lambda) = \sum_{\tau_i \in T} \left[\mathrm{Hist}(\tau_i) - f(\tau_i; \lambda)\right]^2 \qquad (2)$$

ここで，$\mathrm{Hist}(\tau_i)$ はヒストグラムの密度値を，$f(\tau_i; \lambda)$ は PDF の値を示す．

(3) 最小二乗法を用いて λ を最適化する．

指数分布モデルの PDF は次式で与えられる．

$$f(\tau; \lambda) = \lambda e^{-\lambda(\tau - \tau_{\mathrm{shift}})}, \quad \tau \geq \tau_{\mathrm{shift}} \qquad (3)$$

ここで，式 (3) における各パラメータは以下のとおりである．

- $f(\tau; \lambda)$：無操作状態の継続時間 τ が従う確率密度
- λ：無操作状態の発生頻度を表すパラメータ
- τ：無操作状態の継続時間
- τ_{shift}：PDF のシフト量であり，無操作状態の定義時間に対応（本稿では 5 秒）

算出された λ は，次章の支援対象決定ブロックおよび支援タイミング決定ブロックにおいて利用される．

3.5 支援対象決定ブロック

支援対象決定ブロックでは， 3.4 節で算出された指数分布のパラメータ λ を用いて，支援対象を決定する．具体的には，λ の閾値を設定し，閾値より小さい学習者を支援が必要な学習者として決定する．初学者の傾向として「知識や経験が不足しているため，外部情報を検索し，試行錯誤を繰り返す」ことが考えられる．これにより，短い無操作時間 τ_i の無操作状態が多くなり，初学者の λ 値を低くする要因となることが考えられる．一方，経験豊富な学習者は，知識や経験に基づいて問題解決を試みるため，比較的長い無操作時間 τ_i の無操作状態が多くなる傾向があり，結果として λ の値が高くなると考えられる．本稿では，支援対象を決定するための支援対象閾値 λ_{th} は 0.5 に設定した．

3.6 支援タイミング決定ブロック

支援タイミング決定ブロックでは， 3.5 節で決定した支援対象に着目して支援タイミングを決定する．本手法では，支援対象の「最大無操作時間」に基づいて支援タイミングを決定するアプローチを用いる．

本手法の意図は，最大無操作時間が学習者の「長考」を表していると考える点にある．長考しているということは，学習者が課題の解決方法を模索している状態であり，支援が必要であると判断できる．支援対象となる初学者の傾向として最大無操作時間は比較的短くなり，長考の時間には一定の傾向や上限があることが予想される．これらの傾向から，本手法では以下の手順で支援タイミングを決定する．

(1) 3.5 節で決定した支援対象の λ を用いて最大無操作時間の閾値 $\tau_{\max}$ を決定する

(2) 無操作時間 τ_i が閾値 $\tau_{\max}$ を超えた場合，その時間を最大無操作時間として記録する

(3) 最大無操作時間の検出をもとに，S 秒後を支援タイミングとして決定する

支援タイミングを最大無操作の S 秒後とする理由は，学習者に考える時間を与えるためである．学習者が行き詰まりの状態にある際にただちに支援を行うのではなくある程度の時間を設けることで，学習者が自ら解決策を考える機会を確保する．特に，初学者の中には「困ったらすぐに質問する」傾向にある学習者も存在する．学習プロセスでは，自身の知識やスキルを身につけるためにしっかりと考えて試行錯誤することが重要である．これにより，学習者が単に支援を受けるだけでなく，自律的な学習を促進することが期待できる．

$\tau_{\max}$ 及び S 秒の決定方法については，評価において議論する．この手法により，学習者が長考する状況を適切に捉え，効率的かつ効果的な支援タイミングを提供できると考えられる．

図 3 実験環境

4. 評価

3. で示した支援対象者及びタイミング決定手法の有効性を検証するため，被験者が実際にプログラミング問題を解いている間に収集した行動データを収集し，支援対象精度及び支援タイミング誤差を評価した．

4.1 評価環境

図 3 に実験環境を示す．ソースコードの編集，コンパイ

表 1 実験内容

条件	基礎問題		高難易度
	1 問目	2 問目	1 問目
難易度	普通	難しい	難しい
制限時間	60 分		
生成系 AI の利用	利用不可		
ブラウザの利用	利用可能		

表 2 被験者の解答問題内訳

	応用問題	基礎問題
下級生 (1〜3 年生)	0	15
上級生 (4 年生以上)	8	2

ル，実行はマイクロソフト社の Visual Studio Code (VScode) ソースコードエディタを使用した．VScode は Dell 社製「Latitude 3520」ノート PC で実行した．プログラミング中の被験者からキーボードおよびマウスの行動データを収集するために，データ収集用のコードを実装した．実装したコードは，キーボードが入力したキーの種類，マウスが移動した x，y 座標，キーボード・マウスそれぞれが操作された時間を記録するためのコードである．データ収集用のコードは Python を用いて実装した．学習中の被験者の検索履歴や，特定のタブでの操作時間を調査するために ActivityWatch を使用した．

被験者は 10 代から 20 代の大学生 25 名である．全ての被験者が，学部 1 年次に C 言語の習得を目的とした講義を履修済み，もしくは履修中の学生である．

表 1 に実験内容を示す．各被験者にはプログラミング問題を提示し，ノート PC には 3 キーキーボードを接続し，被験者には支援が欲しいときに押下するように指示した．基礎問題は難易度が「普通」と「難しい」の 2 問を用意し，応用の問題は難易度が「難しい」の 1 問のみとした．各課題の制限時間は 60 分とし，生成系 AI の利用を禁止した．これは，AI を使用せずに被験者自身の行動を記録することが目的である．一方で，ブラウザの利用は可能とした．実験を実施するにあたって，被験者には実行環境や実行，コンパイル等のコマンドの説明を行い，C 言語や実験環境に対する基礎的な理解を促した上で実験を実施した．

表 2 に，被験者の解答問題内訳を示す．被験者 25 名のうち，学部 1 年生から学部 3 年生の被験者 15 名は全員が基礎問題を解いた．学部 4 年生から修士学生合計 10 名のうち，2 名のみ基礎の難易度の低い問題を解き，それ以外は応用の難易度の高い問題を解いた．

実験終了後，被験者にはアンケートを実施した．アンケートでは，取り組んだ問題に対する難易度，授業外でのプログラミング経験の有無，実験に対する感想やフィードバックについての回答を指示し，集計を行った．問題の難易度は 5 段階で回答を指示し，1 が簡単，3 がまあまあ難しい，5 が全くわからない，とした．

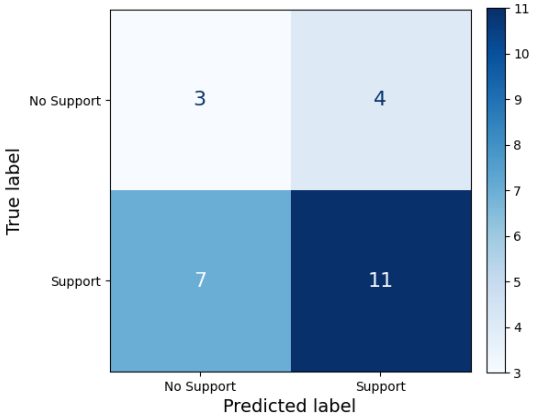

図 4 支援対象決定精度の混同行列

4.2 支援対象決定精度

3.5 節で示した支援対象者決定手法の性能を検証するため，支援対象者決定精度を評価した．被験者アンケートで「問題は難しかった」または「授業外でのプログラミング経験はない」と回答した被験者を支援対象者の真値として，支援対象の決定結果から，True Positive (TP)，False Negative (FN)，False Positive (FP) の人数を評価した．そして，TP，FN，FP から F1 スコアを計算して支援対象者推定精度とした．

図 4 に支援対象者決定精度の混同行列，**表 3** に評価結果を示す．本評価では，支援対象閾値 λ_{th} を 0.5 とした結果である．図より，TP が多いことから実際に支援が必要な被験者を検出していることがわかる．ただし，FN が多いことから支援が必要であるが支援が不要と判断していた被験者が 7 名存在したことがわかる．表より，決定精度は 0.67 であった．

これらの評価結果より，λ を用いた支援対象決定精度にはある程度の有効性があると言える．

4.3 支援タイミング誤差

3.6 節で示した支援タイミング決定手法の性能を検証するため，最大無操作時間と λ の関係から最大無操作時間の閾値について議論し，決定した支援タイミングと実際の支援タイミングとの誤差を検証した．

3.6 節で示したパラメータである S を変更しながら，決定した支援タイミングから実際の支援タイミングまでの誤差の平均を調査した．**図 5** に支援タイミングの平均誤差の評価結果を示す．表より，誤差が最小の時の S は 1224.33 秒，誤差の最小値は 564.27 秒であった．図より，グラフの外形から誤差の最小値は一点となっており，最小を示す値が適切な支援タイミングであることを示した．

表 3 支援対象決定精度の評価結果

	評価結果
適合率 (Precision)	0.73
再現率 (Recall)	0.61
F1 スコア	0.67

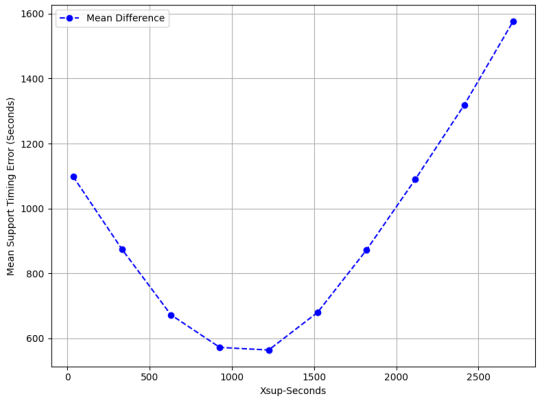

図 5 支援タイミングの平均誤差

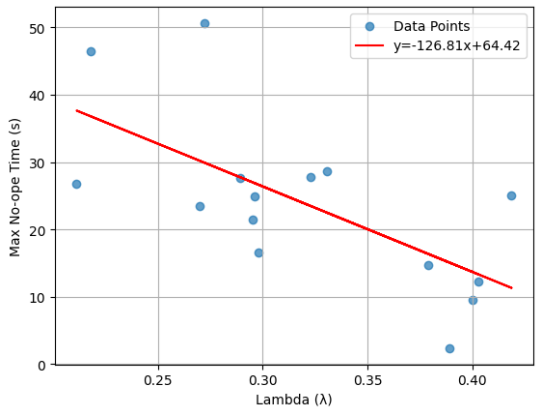

図 6 ラムダが 0.5 未満の被験者のラムダと最大無操作時間の関係図

4.2 節で決定した λ が 0.5 未満の被験者（低ラムダ群）を支援対象とした場合の λ と最大無操作時間の関係を調べ，最大無操作時間に一定の傾向があるか検証した．**図 6** に低ラムダ群の λ と 3 キーキーボードのキー押下による支援タイミングと最大無操作時間の関係を示す．λ と最大無操作時間には中程度の相関関係が見られた．低ラムダ群 15 名全ての被験者は，最大無操作時間が 50 秒以内であった．このことより，低ラムダ群の被験者には最大無操作の上限に傾向があり，λ と関係があることを示した．この結果，λ をもとに最大無操作時間閾値 $\tau_{\max}$ を決定することが可能であり，最大無操作をもとにした支援タイミング決定の実現可能性を示した．

5. 議論

本研究では，プログラミング学習者の無操作状態に着目し，支援対象と支援タイミングを決定する手法を提案した．その結果，支援対象決定精度は F1 スコア 0.67 であり，支援タイミング誤差が最小時の S は 1224.33 秒，誤差の最小値は 564.27 秒であった．提案手法は，学習者のソースコードの参照や，センサなど特別な機材の利用を行わず，学習者のキーボードおよびマウスによる行動データのみで，支援対象と支援タイミングの決定が可能であることを示した．最大無操作時間の発生を「長考」として学習者の支援タイミング決定が可能であることを示した．

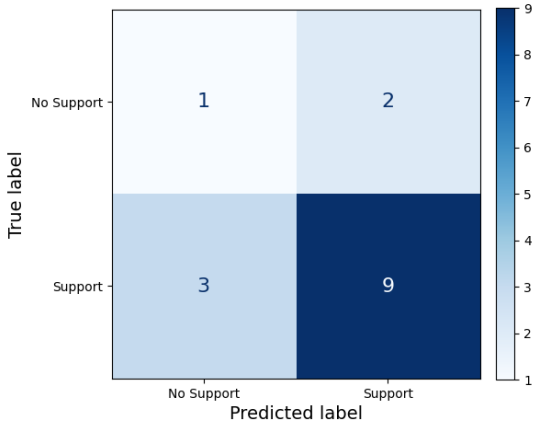

図 7 低学年被験者における支援対象決定精度の混同行列

以降は，支援対象決定精度やパラメータ λ について考察を行い，提案手法の限界について述べる．

5.1 支援対象決定精度についての考察

図 7 に低学年被験者における支援対象決定精度の混同行列，**表 4** に評価結果を示す．本研究は，「初歩のつまづき」を発生させやすい初学者を支援対象としている．そのため，比較的初学者が多いと思われる低学年に着目した精度を算出した． 4.2 節において示した，全被験者をもとにした結果より 11 ポイントの精度向上を確認した．このことから，低学年向けの支援において，本手法は更なる有効性を示した．

支援対象決定の評価には，被験者によるアンケート結果を用いた．本論文では，「問題は難しかった」または「授業外でのプログラミング経験はない」被験者を支援対象の真値とした．一方，「問題は難しかった」と回答した被験者を真値として評価を行った場合，FN に該当する被験者が 8 名となった．そのうち，5 名については以下の理由により FN であったと推測される．

- 2 名は難易度の低い問題を解きたいと申告した 4 年生であり，比較的難易度が低いた感じたため
- 1 名は C 言語の文法を忘れたからできなかった，覚えていれば簡単にできた，と回答したため
- 2 名は C 言語の講義を履修中の 1 年生であり，習ったばかりで自信があったため

これらの理由により，5 名はアンケート上で「問題は難しかった」と回答することがなかったと考えられる．本論文の評価は，主観的なアンケートの回答結果を用いたことによる精度の低下が考えられる．支援対象の真値として，客観的な指標を用意する必要がある．

表 4 低学年被験者における支援対象決定精度の評価結果

	評価結果
適合率 (Precision)	0.82
再現率 (Recall)	0.75
F1 スコア	0.78

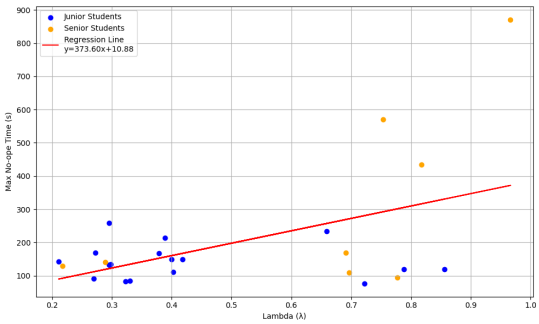

図 8 最大無操作時間と λ の関係

5.2 パラメータ λ による支援対象の分類についての考察

図 8 に，被験者 25 名の λ と最大無操作時間の関係を示す．図より，λ が 0.5 未満と 0.5 以上で被験者が分かれていることが確認できる．さらに，高ラムダ群は最大無操作時間にばらつきが見られるのに対して， 4.3 節で述べたように低ラムダ群は最大無操作時間に一定の傾向があり，ばらつきが少ないことが確認できる．

図 8 の点は，被験者が解いた問題によって色を分けている．このうち，低ラムダ群に属する 2 名の被験者は応用の問題を解いている．この 2 名は，問題の変更には至らなかったものの，事前にプログラミングが得意でないという申告を受けていた．一方，図より高ラムダ群に応用の問題を解いた被験者が多いことが確認でき，これは高ラムダ群には上級生が多い傾向にあることを示している，これらのことから，λ によって支援対象を分類することは，ある程度の妥当性があると考えられる．

5.3 提案手法の限界

本研究は， 3.1 節で述べたとおり，プログラミング学習者のキーボードおよびマウスを操作していない「無操作状態」に着目した手法を提案している．そのため，学習者が「初歩のつまづき」に陥っている際に，「同一のキーを押下し続ける」「マウスを動かし続ける」といった恒常的に操作を続ける場合は支援が困難である．このような場合については，学習者の操作に一定の傾向が存在するか確認することで支援が可能になると考えている．例えば，連続して同一のキーを押下し続けている場合や，同一の軌跡によるマウス操作が行われている場合などを検知することで，無操作状態でない学習者に対しても支援が可能である．

本手法によって支援が可能となる学習者は，初学者やプログラミングを苦手とする学習者である．そのため，経験者やプログラミングを得意とする学習者に対して支援を行うことは困難である．このような場合は，文献 [7] や文献 [6] で報告されている既存手法を用いることで支援が可能だと考えている．これらの手法は， 2.2 節で述べたとおり，プログラミング中のコンパイルや特定の行動の繰り返しを用いて支援の実施を判断している．初学者と異なり，経験者であればコンパイルなどの特定の行動を起こす可能性が高く，特定の行動をもとにした支援が可能である．

6. おわりに

本論文では，初学者が陥りやすい「初歩のつまづき」に対して支援を行うために，プログラミング学習者のキーボード・マウスを操作していない無操作状態に着目することで，優先して支援を実施する支援対象と，適切な支援タイミングの 2 つを決定する手法を提案した．提案手法を評価するために，被験者 25 名のプログラミング中の無操作状態を収集し，支援対象決定精度と支援タイミング誤差を評価した．その結果，支援対象決定精度は 0.67 であり，支援タイミング誤差は 564.27 秒，支援タイミングは最大無操作から 1224.33 秒後であった．今後の課題として，支援対象決定の精度向上を目的とした指標の検討や，恒常的に操作を続ける学習者に対する同一の操作による検知手法の検討が挙げられる．

参考文献

[1] 文部科学省：高等学校学習指導要領 情報科関係資料，https://www.mext.go.jp/a_menu/shotou/zyouhou/detail/mext_01831.html.
[2] 長 慎也，筧 捷彦：proGrep －プログラミング学習履歴検索システム，情報処理学会研究報告コンピュータと教育（CE）， Vol. 2005, No. 15(2004-CE-078), pp. 29–36 (2005).
[3] 田口 浩，糸賀裕弥，毛利公一，山本哲男，島川博光：個々の学習者の理解状況と学習意欲に合わせたプログラミング教育支援，情報処理学会論文誌， Vol. 48, No. 2, pp. 958–968 (2007).
[4] Gupta, R., Kanade, A. and Shevade, S.: Deep Reinforcement Learning for Syntactic Error Repair in Student Programs, *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33, No. 01, pp. 930–937 (2019).
[5] 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， Vol. 35, No. 1, pp. 1_3–1_13 (2018).
[6] 槇原絵里奈，井垣 宏，吉田則裕，藤原賢二，飯田 元：プログラミング演習における探索的プログラミング行動の自動検出手法の提案，コンピュータ ソフトウェア， Vol. 35, No. 1, pp. 1_110–1_116 (2018).
[7] 宇野 健，畝川みなみ：プログラミング演習のための学習状況のリアルタイム フィードバックシステムの開発，県立広島大学経営情報学部論集， Vol. 7, pp. 163–169 (2015).
[8] 浦上 理，長島 和平，並木 美太郎，兼宗 進，長 慎也：プログラミング学習者のつまずきの自動検出，研究報告コンピュータと教育（CE）， Vol. 2020-CE-154, No. 4, pp. 1–8 (2020).
[9] 倉田寛大，辻 愛里，藤波香織：マルチモーダルデータを利用したプログラミング作業中の困惑状態推定，マルチメディア，分散，協調とモバイルシンポジウム 2024 論文集， Vol. 2024, pp. 657–663 (2024).
[10] 江木啓訓，尾澤重知：学習者センシングのための筆記行為の検知手法と評価，日本教育工学会論文誌， Vol. 36, No. Suppl., pp. 181–184 (2012).
[11] Ishimaru, S., Maruichi, T., Dengel, A. and Kise, K.: Confidence-Aware Learning Assistant (2021).