博士論文

# 思考プロセスを検出できる対話型アプリケーションの提案

公立はこだて未来大学システム情報科学研究科

山口 琢

2024年9月5日

Dissertation

# Interactive Applications for Exploring Thinking Processes

Graduate School of Future University Hakodate

Taku Yamaguchi

September 5, 2024

**要旨**

本研究は新しいアプリケーション・ソフトウェアのあり方を提案する。現在、想定しているのはオフィス・アプリケーションや、企業の情報系システムや、教育・人材育成における教材アプリケーションなどである。それらのアプリケーションは、それらを使って文章を書いたり何か問題解決するときの思考プロセスを検出・測定・分析・評価するのに役立つはずである。従来のアプリケーション研究・開発は、主に文章といった出力に注目し、プロセスについては使い勝手の観点で問題にしてきた。それに対して、本研究は出力を作り出すプロセスに注目する。

このようなアプリケーションを使う業務や教育は、アプリケーションが扱うコンテンツ、アプリケーションが備えるべき性質、アプリケーションが記録する測定データ、測定データの分析方法、分析結果に基づく評価や対応を含む。また、コンピューターシステムとしての安定性・性能、およびアクセシビリティも重要である。アプリケーションを教育における演習教材とすれば、コンテンツは問題、コンテンツを開発する作業は作問、測定データはアプリケーションの操作ログ、操作ログの分析結果に基づく評価・対応は学習・指導となる。本研究を構成するこれらの内容は互いに依存し合っている。「任意のアプリケーションでログを出力して分析すれば知りたいことが分かる」といったものではない。提案するような手法で分析するためには必要なデータがあり、それを記録できるようなアプリケーションである必要がある。

まず、最初の1章で、本研究のモチベーションとして、オフィス・アプリケーションの評判を情報の一貫性の観点から振り返る。また、本研究を重要性をもたらす教育分野のニーズを述べる: 教育では、答えのない課題に試行錯誤で取り組める人の育成を重要視し始めている。

2章では関連する研究や取り組みの動向を述べる。本研究のヒントとなった、医薬品業界における strategic review や、学習分析の動向である。

3章では提案するアプリケーションや測定データが備えるべき性質、および、そのようなアプリケーションの例として、本研究で開発したアプリケーションを論じる。

アプリケーションが備えるべき性質で重要なものは、ユーザーの各操作をユーザーの認識や方略を表す「用語」と関連付けられることである。ユーザーのメンタルモデルや UI (User Interface)のオブジェクトをそのように設計する。また、障害のある人も一緒に使えるようなアクセシビリティを備える必要がある。このためには、Web アプリケーションとして実装するのが有利である。

測定データでは、前記の UI オブジェクトをライフサイクル全体にわたって追跡できる必要がある。機密性の高いコンテンツを扱えるために、コンテンツそのものを保存しない測定が望ましい。このことは、分析手法にも当てはまる。

このような性質を満たすアプリケーションとして、ワークシート作文の Topic Writer、文章の断片を取捨選択・並べ替えて作文するジグソー・テキスト、プログラム・コードの断片を取捨選択・並べ替えるプログラミングのジグソー・コード、15パズルなどを開発した。ジグソー・テキストやジグソー・コードではゲームになぞらえて、断片をピース、ユーザーをプレイヤー、問題をパズル問題、パズル問題を解くことをプレイと呼んでいる。これらは、実際の授業などで使わ

れている。本研究は、思考プロセスを顕在化するようなアプリケーションが備える性質や、プロセスを明らかにする方法を提案する。ジグソー・コードなどは、その例である。提案を踏まえて、新しいアプリケーションが開発されたり、既存のアプリケーションがエンハンスされたりすることを期待する。

4章から分析手法を論じる。測定データ(操作ログ)の分析方法は、それによって思考プロセスを検出するものであり、本研究の中核をなす。本研究では新たな分析手法を提案するだけでなく、既存の手法についても批判的に検討する。

4章では時間的な共起分析(temporal co-occurrence analysis)を論じる。これは、前記のUIオブジェクトが前後して操作対象となる頻度を分析するものである。アウトプットのテキストにおいて語が近くに出現する頻度を分析する、従来の共起分析にならった分析手法である。この分析によって、ユーザー(プレイヤー)にとって互いに関連するピースのペアを見いだせた。

5章では時間的な共起分析の応用例を論じる。時間的な共起分析を、不要なピースを含んだ問題(パズル問題)を解くジグソー・コードに適用すると、作問者が意図して紛れ込ませた間違いピースが、正しいピースとの間で取捨選択されたかどうかを検出できる。このような取捨選択は、試行錯誤の一部であり、本研究の目的にかなう。

6章では解の状態遷移を分析する。ジグソー・コードのようなアプリケーションでは、解答の状態をピースの並びで表現することで、解答が作り上げられる様子を状態遷移として表現できる。この状態遷移において閉路を検出し、その閉路の起点が最終的な解答の部分列であるならば、その閉路は意図的な試行錯誤とみなせること示した。アプリケーションがアンドゥ機能を持っていて、それを使ったならば意図的な試行錯誤とみなせるだろう。アンドゥを使った場合、元の状態遷移を逆にたどることになる。本研究は、逆にたどらない遷移についても意図的な試行錯誤とみなせることを示した。

7章ではアプリケーション操作の時間間隔を分析する。ジグソー・テキストやジグソー・コードのピース操作の時間間隔にはプレイヤーの熟達度が現れる。ピースをソフトウェア開発のスクラムにおけるプロダクト・バックログ・アイテムとして、作る順番を決めるといいうパズル問題を解くとき、プロダクト・オーナー経験者の方が未経験者よりも、操作の時間間隔の分散が小さく平均値も小さいことが分かった。ピース操作の時間間隔にはプレイヤーの熟達度が現れるが、この熟達とはジグソー・テキストというアプリケーションの熟達ではなく、パズル問題の内容、この場合はプロダクト・オーナーの経験である。ただし、熟達度は試行錯誤の内容やプレイ戦略そのものではなく、本研究の狙いからは若干離れる。

8章ではプロセスの分析に正解との距離を導入することを論じる。正解との間の何らかの距離を適用して、解くプロセスの時間経過に伴う距離の変化を分析し、距離が大きくなるときにはプレイヤーが迷っていると判定する研究がある。このような判定が妥当かどうか、本研究では検証した。クイックソートなどのアルゴリズムでコンピューターに数を整列させ、途中の数の並び順を記録して、レーベンシュタイン距離など様々な距離の変化を分析した。その結果、整列のプロセスで、必ずしも距離が単調非増加せず、距離が広がることがあるし、正解に達したにもかかわらず再び誤った順序に変わることすらあることが分かった。整列アルゴリズムは、同じルールの

判定と並べ替えを繰り返しているだけであり、途中で悩んでいるとみなすのは妥当ではない。ただし、検証した範囲では、単調非増加する距離も存在した。

9章では提案手法のコンピューター・システムとしての品質・性能を論じる。以上のようなアプリケーションや分析手法が、実験室ではなく実際の運用で、コンピューター・システムとして成立するかどうかは検証が必要である。従来、教育工学研究では、このような検証の報告が少ない。本研究では、サーバーへ送信される測定データの欠損を、提案手法による分析結果への影響の観点から評価して、問題ないレベルであることを確認した。また、本研究で開発したシステムは、200人規模など、いくつかの学校の授業で実際に使われてきた。これら事例をあげることで、提案アプリケーションや分析手法が実際に使えるものであることを示す。

10章では本研究をアクセシビリティの観点から論じる。障害者差別解消法の改正など、我が国は障害のある人もない人も共に暮らせる社会を目指している。新しい技術を開発し導入を目指す研究は、障害がある人でも、それによって利益を得られるよう努力する必要があるだろう。本研究では、ジグソー・テキストにスクリーン・リーダーで操作できるUIを用意した。これによって、画面を見なくても並べ替え問題を解ける。このUIは前記のアプリケーションの条件を満たすものである。予備実験を実施し、このUIで解くプロセスを分析したところ、スクリーンリーダーで解く方が目で見て読んで解くよりも、最初の操作にかける時間が長い傾向が見られた。UIによって解くプロセスが異なることは、異なる思考が起きていることを示唆し、ひいては同じ問題を解いているといえるかどうかの疑問に達する。今後、さらなる研究が必要である。

最後に、残された課題・展望など続く研究への端緒や、アプリケーションを使った事例などをあげる。

以上から、本研究が提案するようなアプリケーションは、特に試行錯誤を含む思考プロセスを検出・測定するのに十分なデータを出力し、それらデータから思考プロセスを分析する手法が存在することを示せた。このような測定は実際の授業で使うのに十分な信頼性を備えている。このようなアプリケーションが多く考案され、実際に使われるようになれば、試行錯誤を含む思考プロセスの是非を評価するのに十分な実態が明らかになるだろう。

キーワード：情報の一貫性、取捨選択・並べ替え型パズル、時間的な共起分析、学習分析

# はじめに

本研究は新しいアプリケーション・ソフトウェアのあり方を提案する。現在、想定しているのはオフィス・アプリケーションや、企業の情報系システムや、教育・人材育成における教材アプリケーションなどである。それらのアプリケーションは、それらを使って文章を書いたり何か問題解決するときの思考プロセスを検出するのに役立つはずである。従来のアプリケーション研究・開発は、主に文章といった出力に注目し、プロセスについては使い勝手の観点で問題にしてきた。それに対して、本研究は出力を作り出すプロセスに注目する(**図1**)。

このようなアプリケーションを使う業務や教育は、アプリケーションが扱うコンテンツ、アプリケーションが備えるべき性質、アプリケーションが記録する測定データ、測定データの分析方法、分析結果に基づく評価や対応を含む。アプリケーションを教育における演習教材とすれば、コンテンツは問題、コンテンツを開発する作業は作問、測定データはアプリケーションの操作ログ、操作ログの分析結果に基づく評価・対応は学習・指導となる(**図2**)。本研究を構成するこれらの内容は互いに依存し合っている。「任意のアプリケーションでログを出力して分析すれば知りたいことが分かる」といったものではない。提案するような手法で分析するためには必要なデータがあり、それを記録できるようなアプリケーションである必要がある。

本稿は3部構成とする。1部は、背景および関連動向を論じた後に、2部の前提となるアプリケーションおよび測定データについて論じる。2部は、本研究の中核となる分析手法を論じる。3部では、本研究の非機能要件ともいうべき性能やアクセシビリティ、また今後の研究への導入といった考察を行う。以下、各部を構成する章を外観する。

## 1部

1章で、本研究のモチベーションとして、オフィス・アプリケーションの評判を情報の一貫性の観点から振り返る。また、本研究を重要性をもたらす教育分野のニーズを述べる: 教育では、答えのない課題に試行錯誤で取り組める人の育成を重要視し始めている。

2章では関連する研究や取り組みの動向を述べる。本研究のヒントとなった、医薬品業界における strategic review や、学習分析の動向である。

3章では提案するアプリケーションや測定データが備えるべき性質、および、そのようなアプリケーションの例として、本研究で開発したアプリケーションを論じる。

アプリケーションが備えるべき性質で重要なものは、ユーザーの各操作をユーザーの認識や方略を表す「用語」と関連付けられることである。ユーザーのメンタルモデルや UI (User Interface)のオブジェクトをそのように設計する。また、障害のある人も一緒に使えるようなアクセシビリティを備える必要がある。このためには、Web アプリケーションとして実装するのが有利である。

測定データでは、前記の UI オブジェクトをライフサイクル全体にわたって追跡できる必要がある。機密性の高いコンテンツを扱えるために、コンテンツそのものを保存しない測定が望ましい。このことは、分析手法にも当てはまる。

このような性質を満たすアプリケーションとして、ワークシート作文の Topic Writer、文章の断片を取捨選択・並べ替えて作文するジグソー・テキスト、プログラム・コードの断片を取捨選択・並べ替えるプログラミングのジグソー・コード、15 パズルなどを開発した。ジグソー・テキストやジグソー・コードではゲームになぞらえて、断片をピース、ユーザーをプレイヤー、問題をパズル問題、パズル問題を解くことをプレイと呼んでいる。これらは、実際の授業などで使われている。本研究は、思考プロセスを顕在化するようなアプリケーションが備える性質や、プロセスを明らかにする方法を提案する。ジグソー・コードなどは、その例である。提案を踏まえて、新しいアプリケーションが開発されたり、既存のアプリケーションがエンハンスされたりすることを期待する。

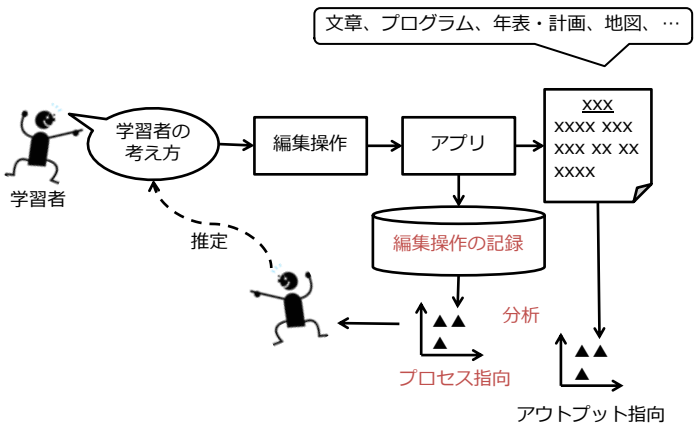



**図1**　プロセスの研究

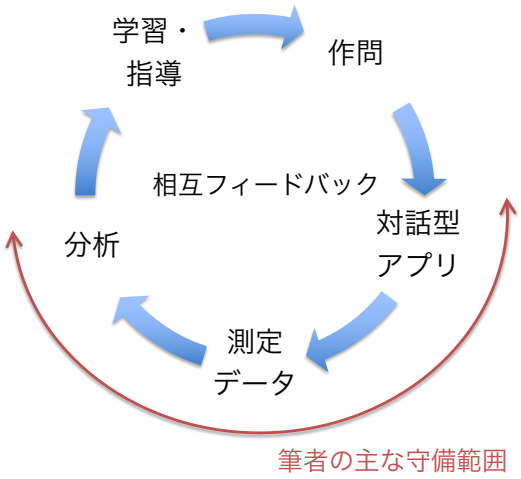



**図2**　研究の構成

**2部**

2部では分析手法を論じる。測定データ(操作ログ)の分析方法は、それによって思考プロセスを検出するものであり、本研究の中核をなす。本研究では新たな分析手法を提案するだけでなく、既存の手法についても批判的に検討する。

4章では時間的な共起分析(temporal co-occurrence analysis)を論じる。これは、前記のUIオブジェクトが前後して操作対象となる頻度を分析するものである。アウトプットのテキストにおいて語が近くに出現する頻度を分析する、従来の共起分析にならった分析手法である。この分析によって、ユーザー(プレイヤー)にとって互いに関連するピースのペアを見いだせた。

5章では時間的な共起分析の応用例を論じる。時間的な共起分析を、不要なピースを含んだ問題(パズル問題)を解くジグソー・コードに適用すると、作問者が意図して紛れ込ませた間違いピースが、正しいピースとの間で取捨選択されたかどうかを検出できる。このような取捨選択は、試行錯誤の一部であり、本研究の目的にかなう。

6章では解の状態遷移を分析する。ジグソー・コードのようなアプリケーションでは、解答の状態をピースの並びで表現することで、解答が作り上げられる様子を状態遷移として表現できる。この状態遷移において閉路を検出し、その閉路の起点が最終的な解答の部分列であるならば、その閉路は意図的な試行錯誤とみなせること示した。アプリケーションがアンドゥ機能を持っていて、それを使ったならば意図的な試行錯誤とみなせるだろう。アンドゥを使った場合、元の状態遷移を逆にたどることになる。本研究は、逆にたどらない遷移についても意図的な試行錯誤とみなせることを示した。

7章ではアプリケーション操作の時間間隔を分析する。ジグソー・テキストやジグソー・コードのピース操作の時間間隔にはプレイヤーの熟達度が現れる。ピースをソフトウェア開発のスクラムにおけるプロダクト・バックログ・アイテムとして、作る順番を決めるといいうパズル問題を解くとき、プロダクト・オーナー経験者の方が未経験者よりも、操作の時間間隔の分散が小さく平均値も小さいことが分かった。ピース操作の時間間隔にはプレイヤーの熟達度が現れるが、

この熟達とはジグソー・テキストというアプリケーションの熟達ではなく、パズル問題の内容、この場合はプロダクト・オーナーの経験である。ただし、熟達度は試行錯誤の内容やプレイ戦略そのものではなく、本研究の狙いからは若干離れる。

8章ではプロセスの分析に正解との距離を導入することを論じる。正解との間の何らかの距離を適用して、解くプロセスの時間経過に伴う距離の変化を分析し、距離が大きくなるときにはプレイヤーが迷っていると判定する研究がある。このような判定が妥当かどうか、本研究では検証した。クイックソートなどのアルゴリズムでコンピューターに数を整列させ、途中の数の並び順を記録して、レーベンシュタイン距離など様々な距離の変化を分析した。その結果、整列のプロセスで、必ずしも距離が単調非増加せず、距離が広がることがあるし、正解に達したにもかかわらず再び誤った順序に変わることすらあることが分かった。整列アルゴリズムは、同じルールの判定と並べ替えを繰り返しているだけであり、途中で悩んでいるみなすのは妥当ではない。ただし、検証した範囲では、単調非増加する距離も存在した。

**3部**

9章では提案手法のコンピューター・システムとしての品質・性能を論じる。以上のようなアプリケーションや分析手法が、実験室ではなく実際の運用で、コンピューター[*1]・システムとして成立するかどうかは検証が必要である。従来、教育工学研究では、このような検証の報告が少ない。本研究では、サーバーへ送信される測定データの欠損を、提案手法による分析結果への影響の観点から評価して、問題ないレベルであることを確認した。また、本研究で開発したシステムは、200人規模など、いくつかの学校の授業で実際に使われてきた。これら事例をあげることで、提案アプリケーションや分析手法が実際に使えるものであることを示す。

10章では本研究をアクセシビリティの観点から論じる。障害者差別解消法の改正など、我が国は障害のある人もない人も共に暮らせる社会を目指している。新しい技術を開発し導入を目指す研究は、障害がある人でも、それによって利益を得られるよう努力する必要があるだろう。本研究では、ジグソー・テキストにスクリーン・リーダーで操作できるUIを用意した。これによっ

---

*1 「コンピューター」は、理工系では「コンピュータ」と表記されることもある。長音記号の使い方に関する、この有名な例について、本稿では一般に広く使われている「コンピューター」を採用する。本学はメールシステムにGoogleを使っているが、メールを「コンピューター」で検索した場合と「コンピュータ」で検索した場合とでヒットするメールやハイライトされるタームが異なる。「同義語の表記ゆれ」として扱われているとは考えにくい。このように検索結果が異なる傾向は、今回試した範囲ではGoogle ScholarやJ-STAGEでも見られ、Bingでは見られない。自社製品での表記ルール変更にあたって、Microsoft社は理由の1つに「コンピュータが一般消費者の日常必需品となるにつれて、末尾の長音を省略する傾向が強い、工業系、自然科学系の表記に対するユーザーの違和感が増している」ことをあげた(「マイクロソフト、自社製品でのカタカナ表記ルールを変更--「ブラウザ」は「ブラウザー」に」 ↗)。近年の機械学習技術の使われ方を見ると、一般と異なる表記をあえて使い続ける意義は再考すべきであろう。

て、画面を見なくても並べ替え問題を解ける。このUIは前記のアプリケーションの条件を満たすものである。予備実験を実施し、このUIで解くプロセスを分析したところ、スクリーンリーダーで解く方が目で見て読んで解くよりも、最初の操作にかける時間が長い傾向が見られた。UIによって解くプロセスが異なることは、異なる思考が起きていることを示唆し、ひいては同じ問題を解いてるといえるかどうかの疑問に達する。今後、さらなる研究が必要である。

最後に、残された課題・展望や、アプリケーションを使った事例などをあげる。

**記法など**

図表には全体で通し番号がついている。図表は複数個所から参照されるが、最初の参照は太字で強調される。

参考文献も全体で通し番号がついている。文献も複数箇所から参照されるが、最初の参照は太字で強調される。

# 目次

# 1部

# 1 背景と目的

本研究のきっかけでありモチベーションとなったのは、文書の一貫性(coherence)支援である。これについては、コンピューターの歴史を綴った論文誌に載っているアプリケーション・ソフトウェアの評判に端的に現れている。曰く、スプレッドシートは革命的なアプリケーションである。なぜなら、1つのセルの値を変えると、全体の一貫性が保たれるように、他のセルの値が瞬時に書き換わる。他方でワードプロセッサーは便利だが革命的ではないとの評判であり、もっぱら販売戦略に位置づけて語られている。では、何ができればワードプロセッサーも革命的となるか？

本研究の重要性は教育分野のニーズにある。教育では、答えのない課題に試行錯誤で取り組める人の育成を重要視し始めている。これをデジタル教材システムの研究に当てはめると、従来は児童・生徒・学生の成績といったアウトプットをもって研究結果を評価できた。しかし、課題に答えがない(他者や以前と比較・評価できる答えがない)とき、このように研究結果を評価できない。

一貫性のある情報を試行錯誤しながら作り上げるプロセスを評価できるアプリケーションを開発できれば、これらのモチベーションとニーズに応えられるだろう。

### 1.1　文書の一貫性

文章の大域的な整合性を著者が構築したり読者が読み取ったりするときの思考を問題とするとき、言語学や英語教育やアカデミック・ライティングの研究では、このような整合性を英語でcoherenceと呼び、日本語では一貫性と訳されている[1]。

### 1.2　スプレッドシートと比べたワードプロセッサーの評判

アプリケーション・ソフトウェアの歴史的な評判(あくまで評判)を調べると、情報(文章)の一貫性を支援するという観点では、ワードプロセッサーがスプレッドシートに比べて評価されていないことが分かる。

IEEE Annals of the History of Computing[*2]という査読付きジャーナルで、Ceruzziらは次のように述べている:

> Out of all those software areas, two in particular stand out: word processing and spreadsheets. Although the term word processing derives from the minicomputer era, the concept of working with words and turning rough text into polished documents is obviously much older. By contrast, **the VisiCalc spreadsheet paradigm was new.** People had, of course, been working with rows and columns of numbers long before the punched card era. But no mainframe or minicomputer user, no matter how well-connected they were, could make **a simple "what-if" change to a single spreadsheet cell and watch those changes propagate through the rest of the data**.[2]
> (**強調**は筆者によるもの。)

さらに:

> The electronic spreadsheet became a "must-have" tool for anyone doing financial calculations. The ability to do **instant "what-if" analyses** thoroughly changed how mergers and acquisitions, and budgets, were analyzed. **Many alternative assumptions could be examined to determine the projected outcomes. Financial analysis moved into an entirely different level** because of this new technology.[3]
> (**強調**は筆者によるもの。)

日本の文献ではどうか。情報処理学会が2010年に日本のコンピューター史をまとめた書籍では、「1・7・5 日本語処理技術」で日本語入力について詳しく述べている[4]。

その後、次のように述べている:

---

*2 IEEE Annals of the History of Computing: https://ieeexplore.ieee.org/xpl/RecentIssue.jsp?punumber=85 ↗

> ワープロは日本人がキーボード文化に馴染んだきっかけでもあった。また、メーカーにとっては一般大衆向け製品を製造販売する体制造りを必要とした。いずれも次のPC時代の布石としての意味は大きい。[4]「2・1・7 日本語ワードプロセッサ時代の幕開け」

同書ではさらに、次のように指摘している:

> 196年、米国IBMはタイプライタに処理装置・記憶装置を付けたMT/ST(Magnetic Tape/Selectric Typewriter)を発表し、word processingという概念を提唱した。MT/ST最大のメリットは打ち損じを簡単に修正でき、また文書の再利用を可能にしたことである。これはオフィス業務にとって大きな進歩であった。[4]「3・6 日本語ワードプロセッサ(ワードプロセッサ)」

続けて文字コードについても、ワードプロセッサと同じレベルの節を設けて詳しく述べている。

1995年のNHKスペシャル「新・電子立国 (6) 時代を変えたパソコンソフト－表計算とワープロ・日米開発史」[*3]では、時代を変えたパソコンソフトとして、米国からて表計算(スプレッドシート)と日本からATOKを取り上げた。後に詳しく書籍化された内容も同様である**[5]**。表計算に対する評価は、前述のIEEEの論文と同様である。それと対で取り上げているのは、ワープロというより日本語入力である。文章の"what-if"や一貫性維持といった観点はない。情報機器で日本語を扱える重要性は言うまでもなく、歴史を振り返るときにこのような構成になるのは理解できる[*4]。

文章の一貫性は、アカデミック・ライティングなどで重要視されているが、ワードプロセッサーがそれに応えたという評判はない。では、どうすれば文章の一貫性を支援できるのか？これが本研究の動機である。

### 1.3 教育で重視される試行錯誤

教育分野では、答えのない課題に試行錯誤で取り組める人材の育成を重要視し始めている**[6]**。従来、問題解決プロセスは、アウトプットに基づいて定性的に評価することで研究されてきた。そこで難しいのは、普遍的な正解がない場合、解答(アウトプット)の答え合わせを評価方法として使えないことである(**図3**)。

---

*3 NHKスペシャル　新・電子立国 `https://www2.nhk.or.jp/archives/movies/?id=D0009040317_00000` ↗

*4 WYSIWYG (What You See Is What You Get)は革命的かもしれないことは付け加えておく。ディスプレイの表示と紙への印刷結果が異なるのはかなり問題である。ただし、紙への印刷結果と同じ画面に直接編集するのは別の話で、テキストを編集するたびに図がアチコチ飛び回ることを煩わしく感じるユーザーも多いのではないか。

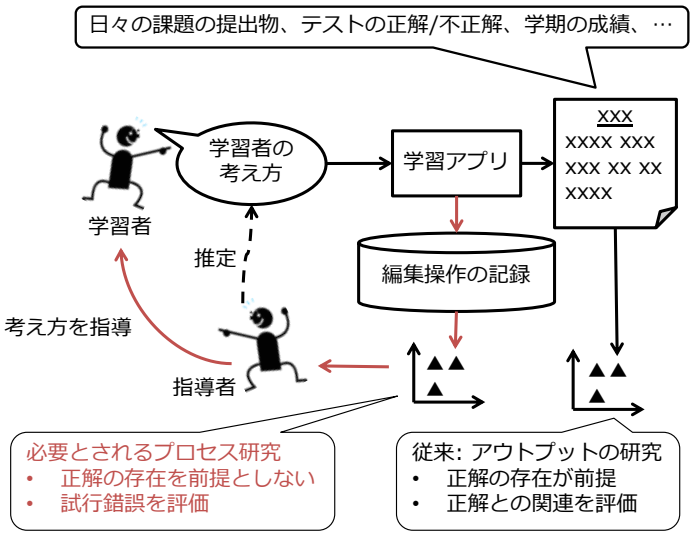



**図3**　教育におけるプロセスの研究

近年、教育では考え方／思考力といったプロセスが重要視され、試行錯誤を肯定的に評価し、かつ評価にあたって試行錯誤に種類を想定している。中央教育審議会は次のように指摘している:

> 人工知能がいかに進化しようとも、それが行っているのは与えられた目的の中での処理である。一方で**人間は**、感性を豊かに働かせながら、どのような未来を創っていくのか、どのように社会や人生をよりよいものにしていくのかという目的を自ら考え出すことができる。多様な文脈が複雑に入り交じった環境の中でも、場面や状況を理解して自ら目的を設定し、その目的に応じて必要な情報を見いだし、情報を基に深く理解して自分の考えをまとめたり、相手にふさわしい表現を工夫したり、**答えのない課題に対して、多様な他者と協働しながら目的に応じた納得解を見いだしたりすることができる**という強みを持っている。
> (**強調**は筆者によるもの。)

さらに:

> このために必要な力を成長の中で育んでいるのが、人間の学習である。**解き方があらかじめ定まった問題を効率的に解いたり、定められた手続を効率的にこなしたりすることにとどまらず**、直面する様々な変化を柔軟に受け止め、感性を豊かに働かせながら、どのような未来を創っていくのか、どのように社会や人生をよりよいものにしていくのかを考え、主体的に学び続けて自ら能力を引き出し、**自分なりに試行錯誤したり、多様な他者と協働したりして、新たな価値を生み出していくために必要な力を身に付け**、子供たち一人一人が、予測できない変化に受け身で対処するのではなく、主体的に向き合って関わり合い、その過程を通して、自らの可能性を発揮し、よりよい社会と幸福な人生の創り手となっていけるようにすることが重要である。

としている[6](p.10)。国立教育政策研究所の教育課程研究センターは学習評価を解説する事例**[7]**(pp.54-59)で、数式のパラメータを決めるときに「適当な数を入力していき調整する」ようなものを「見通しを持たない試行錯誤」とし、「目安を求めてから調整する」ようなものを「見通しを持った試行錯誤」としている。そのうえで、見通しのない試行錯誤を行った場合に「おおむね満足できる」、見通しのある試行錯誤を行った場合に「十分満足できる」と解説している。試行錯誤に種類があることと、いずれも肯定的に評価していることが特徴である。

ただし、試行錯誤の現状について、何らかの統計的な報告があるわけではない。また、試行錯誤を好意的にとらえるべき根拠が示されているわけではない。

学習分析において試行錯誤は、好意的に評価されてるとは言い難い。例えば、Perkinsらは、被験者/学習者が問題を解くプロセスを分析して、学習者が正解から遠ざかる様子を表現して「tinker(下手にいじくり回してる)」と述べている:

> Students often program by means of an approach we call **tinkering**-they try to solve a programming problem by writing some code and then making small changes in the hopes of getting it to work. In some cases this strategy can be effective, while at other times it interferes with students' progress in solving programming problems.**[8]**
> (**強調**は筆者によるもの。)

別の研究では、いろいろやっているが正解に近づかない様子を「wheelspin」と表現している**[9]**:

> However, not all students are able to acquire skills within an ITS(Intelligent Tutoring Systems), and some spend a considerable amount of time stuck in the mastery learning loop without any learning occurring. Aside from simply wasting the learner's time, such an experience is presumably frustrating as learners are repeatedly presented with problems they are clearly unable to solve. We refer to this phenomenon as "**wheel-spinning**," referring to a car stuck in mud or snow; its wheels are spinning rapidly, but it is not going anywhere. Similarly, students are being presented with many problems, but are not making progress towards mastery. Later, we will discuss possible connections with other negative behaviors such as gaming.**[10]**
> (**強調**は筆者によるもの。)

wheelspinとはタイヤの空転のことで、冬の函館ではおなじみの光景である。Tinkerにしろwheelspinにしろ、ポジティブなことばではない。まっすぐ最短距離を正解に向かうことが良いことで(それが良いことに、筆者も異存はないが)、試行錯誤を肯定的に評価しない傾向が学習分析にあり、かつ、そのことに無自覚な様子が、各論文から見て取れる。

### 1.4 研究の目的

本研究の目的は、人が情報に一貫性をもたらすための思考プロセスを、試行錯誤を含めて検出できる中立的な(ニュートラルな)手法を明らかにすることである。将来的には、これによって、あるクラスとか学年全体といった集団について、試行錯誤の実態を明らかにできるだろう。中立的なとは、「試行錯誤は下手ないじくり回しである」とか「試行錯誤は空転である」とか「試行錯誤はつまづきである」といった前提を持たないということである。試行錯誤が本当に良いことなのか、やはり役に立たないのか、議論するのはその後である。

プロセスに着目して一貫性の実現を支援するアプローチには、本研究のヒントとなった先例がある。次の章では、それらを検討する。

# 2 関連する動向

ここでは関連する研究や取り組みの動向を述べる。本研究のヒントとなった、医薬品業界における strategic review や、学習分析の動向である。

### 2.1 医薬品業界のstrategic review

医薬品業界では、文書のレビュー方法の1つにstrategic reviewがある。strategic reviewはレビュー対象の文書をnon-linearに読むことが特徴で、レビューの質が読み方から分かると考えられている[11]。

レビューには2つの種類があり、1つはstrategic review、もう1つはinspectional reviewである(**図4**)。本研究にとっては、strategic reviewの「非線形」が重要である。Strategic reviewは、その文書が目的に適っているか、製品(医薬品など)の戦略を支援するかをレビューする。Inspectional reviewは内容の正確さなどをレビューする。Strategic reviewは「非線形」が特徴で、ほとんどのレビュー者が注目すべきことである。Inspectional reviewは「線形」が特徴で、ほとんどのレビュー者が注目することではない。

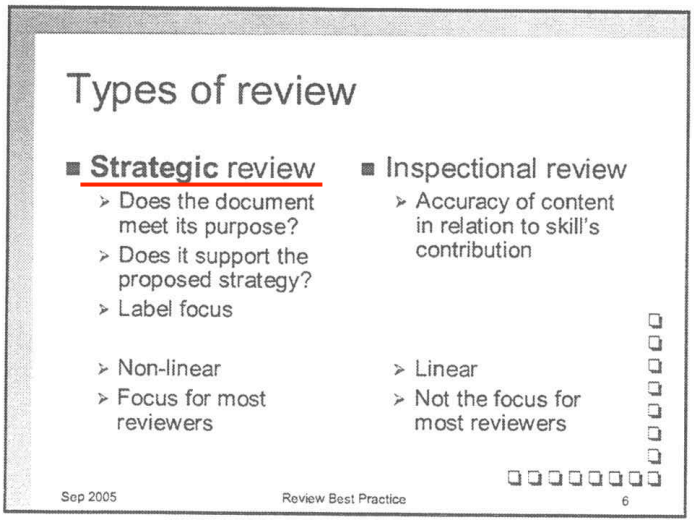



**図4** Types of review
"Review best practice"[11]から引用。赤い強調は筆者による。
(以降、自然言語部分が英語の図は英語の論文や発表資料より引用)

「非線形」とは読み方のことである(**図5**)。途中を飛ばしてIntroductionとDiscussionを交互に読む、ObjectivesとConclusionsを交互に読むといった読み方の非線形を指す(**図6**)。Strategic reviewやinspectional reviewには、それぞれ適切な時期、文書のステージがあって、初期のレビューではstrategic reviewのみ行い、resultsやdiscussionに集中する(**図7**)。Strategic reviewの原則はconsistency (整合性)である。文書と製品のlabelとの整合性、複数の文書をまたがる整合性、文書内の整合性である(**図8**)。

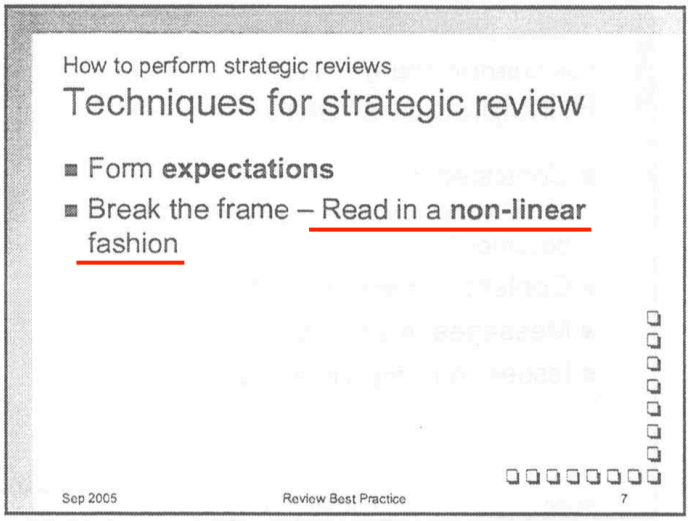



**図5**　Strategic reviewのテクニック

"Review best practice"[11]から引用。赤い強調は筆者による。

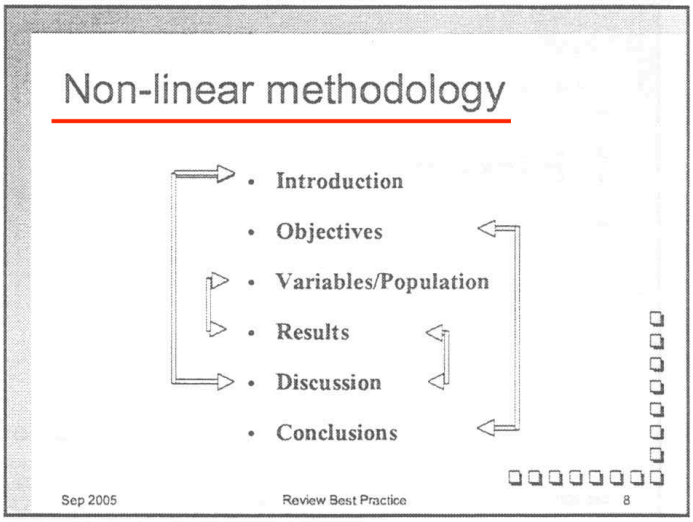



**図6**　非線形な手法

"Review best practice"[11]から引用。赤い強調は筆者による。

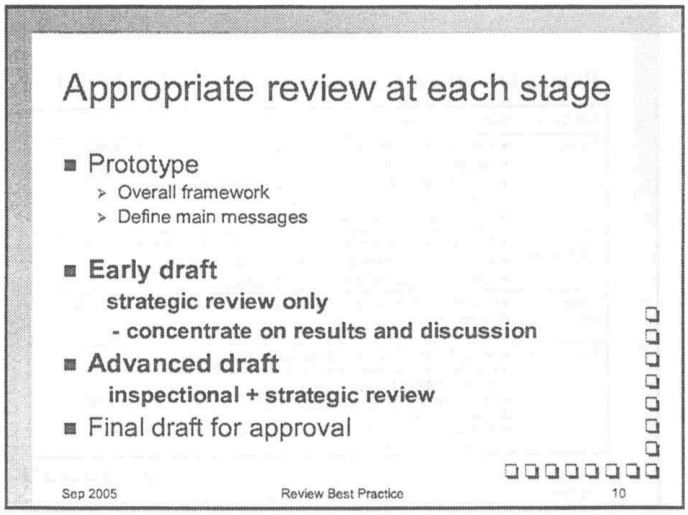



**図7**　ステージごとの適切なレビュー
"Review best practice"[11]から引用

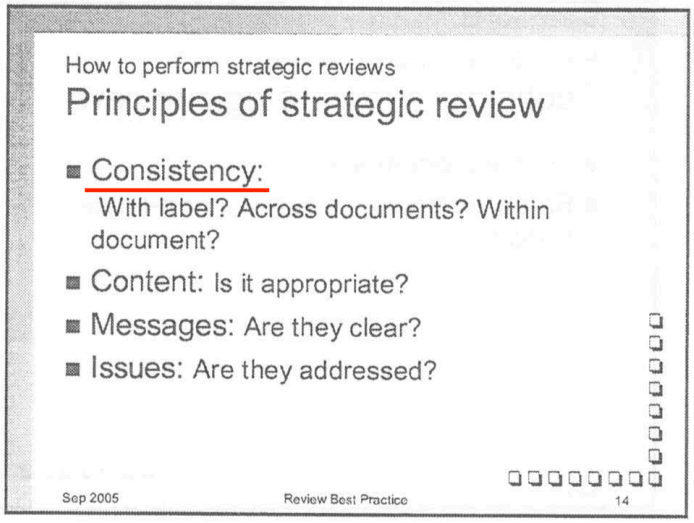



**図8**　Strategic reviewの原則
"Review best practice"[11]から引用。赤い強調は筆者による。

すなわち、整合性を問題にすると、文書を非線形に読むことになるはずだというのである。どのように読むかによって、何をしているかの思考が分かる。思考内容が読み方に現れるというのが、このレビュー手法の特徴であり、本研究の発想の元である。

なお、ここで整合性(consistency)ということばが使われているが、本稿では先行研究に合わせて一貫性(coherence)を使うことにする。一貫性については、前の1.1節でとりあげた。

## 2.2 Single-Document Quality Control と Paired-Document Quality Control

医薬品業界の文書については、GCP(Good Clinical Practice、医薬品の臨床試験の実施の基準)を満たす文書の作成にあっての留意点などをまとめた文献**[12]**でも、「第5章 文書の点検及び保存・管理」において次のように述べている。

> 文書の点検方法には、大きく分けて2つの方法がある。1つ目の方法はSingle-Document Quality Controlであり、もう一つの方法はPaired-Document Quality Controlである。

Single-Document Quality Controlの内容は、記載漏れ、誤字脱字、同一文書内での矛盾である。Paired-Document Quality Controlの内容は、2つ以上の文書間で比較して点検するものである。Paired-Documentでは、矛盾や一貫性を問題にして、例えば「予定した解析方法と、実際の解析方法の矛盾」などが例としてあげられている。

一貫性を問題とするところが前述のstrategic reviewと共通している。ただし、点検(レビュー)の順番については、Single-DocumentをPaired-Documentよりも先に実施するとしている。文書間の一貫性よりも誤字脱字を先に問題とするのか？については、strategic reviewとは異なるかもしれない。

## 2.3 KWIC

KWIC (Keyword In Context)は索引やアドホックな検索結果を表示するときに使われる表示形式の1つである。索引語や検索語(keyword)を、それが登場する文脈(context)とともに表示する。これによって、その検索語の使われ方の一貫性(coherence)を確認しようというものである。ただし、KWICが論じられるときは、一貫性(coherence)よりも、コンコーダンス(concordance、「一致」、「調和」といった意味)という用語が使われる。コンコーダンスには用語集という意味もあり、例えば、シェイクスピアのコンコーダンス[*5]であれば、ハムレットの有名なセリフ「To be, or not to be, that is the question.」の「question」の文脈を調べられる。

本稿のある時点の原稿から、miテキストエディタ[*6]を使って、キーワード「検出」を検索した結果のKWIC表示を**図9**に示す。ここでは、context(文脈)とはkeyword(検索語)「検出」の前後のテキストのことである。テキスト・マイニング分野ではKH CoderのKWICコンコーダンス[*7]を思い出す人も多いだろう。

---

*5 Open Source Shakespeare: `https://www.opensourceshakespeare.org/` ↗

*6 miテキストエディタはMac用の日本語テキストエディタである: `https://www.mimikaki.net/` ↗

*7 KH Coder: `https://khcoder.net/` ↗


**図9** ある時点で本稿で「検出」を検索した結果のKWIC表示

## 2.4 学習分析

近年、学習アプリケーションを自分で作って学習・教育に使って研究できる時代になってきた。安価で高性能なクラウド・プラットフォームの充実によって、数百人程度のクラスで使うアプリケーションを以前よりも容易に開発・運用できるようになった。Web標準の機能が充実して、PCやスマートフォンやタブレットのブラウザがそれを実装したことで、BYOD (Bring Your Own Device)の環境にも対応できる。これによって、必要なデータを取れるように学習アプリケーションを自分で開発して授業で使えるようになった。実際、プログラミング学習の学習分析などでは、そのような研究が増えている。

学習分析(Learning Analytics)と言えば、週・月・年といった時間スケールで、学習者の提出物、テストの点数、期末の成績などのアウトプットを分析することを想像するかもしれない。本研究で取り上げるのは、ときに数秒間隔で起きる操作の繰り返しによって数分で完了することもあるプロセスを測定して、学習者の考え方を分析するものである**[13]**。

学習分析の研究の構成は次にように整理できる: (1) 学習・指導の計画に従って問題を作り、(2) 学習アプリケーションに入力して学習者に解かせる、(3) 学習アプリケーションには問題を解くプロセスを測定する機能があり、データとして記録される、(4) この測定データを分析して、

(5) 学習者や作問を評価し、次の学習・指導に反映する。作問、アプリケーションなどは相互に関連していて、学習・指導の必要に応じてアプリケーションや測定データを改善する。本稿もこのサイクルに沿って、アプリケーションと測定データから順に論じる。

## 2.5 プロセスの分析

ユーザーの画面操作の分析は新しいものではない。Webサイト、ECサイトのアクセス解析には馴染があるだろう。これら分析にはユーザーを誘導する目的があって、それが実現したかをモニターしている。同様の例に、オンラインゲームも挙げられるだろう。こちらも目的を持って分析されていて、ユーザーを飽きさせないようになど工夫の手がかりとなる。

### 2.5.1 プログラム・コードの行数の時系列

吉田ら**[14]**は、独自に開発したJavaScriptの学習環境Sumatraを使って、段階的にコーディングを進める漸進的プログラミングの授業で、各段階(ステップ)ごとに保存されるコード行数を分析した。中間試験の成績が上位の受講生は行数の上下がほとんど見られないのに対し、下位の受講生は行数が上下することを見出した。コード行数の上下は、プログラミングの苦戦の現れと推定された。

### 2.5.2 正解との距離の時系列

正解との距離の時系列を分析する研究がある。これについては、後の8章で検討する。

## 2.6 教師と学生の閲覧ページの同期

教師は学生の理解の進み具合に授業のスピードを合わせたい。学生のデジタル教材の閲覧状況をリアルタイムに測定・分析・可視化して教師にフィードバックすることで、これを実現した事例がある。

島田ら**[15]**は、デジタル教材システムBookRollの閲覧状況をリアルタイムにヒートマップ形式で教師に提示するシステムを開発した。これによって学生たちの閲覧状況を1分単位で確認できる。教師が説明しているページよりも以前のページを多くの学生が閲覧してるなら、授業が進みすぎている可能性がある。このようなモニタリングを導入したところ、教師と学生で見ているページの同期率が上昇した。

# 3 対話型アプリケーションと測定データ

ここでは提案するアプリケーションや測定データが備えるべき性質、および、そのようなアプリケーションの例として、本研究で開発したアプリケーションを論じる。各アプリケーションは後の各分析手法の章でも説明されるが、ここでは、そのような性質の観点から論じる。

アプリケーションが備えるべき性質で重要なものは、ユーザーの各操作をユーザーの認識や方略を表す「用語」と関連付けられることである。ユーザーのメンタルモデルやUI (User Interface)のオブジェクトをそのように設計する**[16]**。また、障害のある人も一緒に使えるようなアクセシビリティを備える必要がある。このためには、Webアプリケーションとして実装するのが有利である。

測定データでは、前記のUIオブジェクトをライフサイクル全体にわたって追跡できる必要がある。機密性の高いコンテンツを扱えるために、コンテンツそのものを保存しない測定が望ましい。このことは、分析手法にも当てはまる。

このような性質を満たすアプリケーションとして、ワークシート作文のTopic Writer、文章の断片を取捨選択・並べ替えて作文するジグソー・テキスト、プログラム・コードの断片を取捨選択・並べ替えるプログラミングのジグソー・コード、15パズルなどを開発した。ジグソー・テキストやジグソー・コードではゲームになぞらえて、断片をピース、ユーザーをプレイヤー、問題をパズル問題、パズル問題を解くことをプレイと呼んでいる。これらは、実際の授業などで使われている。本研究は、思考プロセスを顕在化するようなアプリケーションが備える性質や、プロセスを明らかにする方法を提案する。ジグソー・コードなどは、その例である。提案を踏まえて、新しいアプリケーションが開発されたり、既存のアプリケーションがエンハンスされたりすることを期待する。

### 3.1 要件

提案するアプリケーションは、次のような性質を備えるべきである。特に、UIオブジェクト、追跡性は重要。

#### 3.1.1 UIのオブジェクトと操作を記述することばの粒度

ボードゲームやカードゲーム、野球などのゲームを見ていると、そのゲームでの行動・操作からプレイヤーの戦略を判断することができる。したがって、ユーザーの問題解決を支援する対話型コンピューターアプリケーションが、ユーザーの思考や判断に近い行動をするように設計されていれば、そのアプリケーション上のオブジェクトに対するユーザーの操作を計測・分析することで、その思考を推測することができる。本稿では、このようなアプリケーションを "パズル"、アプリケーションのユーザーを "プレイヤー"、"コード行 "や "文章 "などのオブジェクトを "ピース"、パズルを解くことを "プレイ "と呼ぶ。

本研究は、ユーザーインターフェース(User Interface, UI)を構成するオブジェクトが、ユーザーの認識や戦略を記述するために使用される用語と関連付けられるようにアプリケーションを設計することを提案する。

オブジェクトを「ページ」(1ページ目、2ページ目.....）とした場合、分析の結果、「プレイヤーはまず2ページ目あたりを読み、次に4ページ目あたりに移動する傾向がある」ということが分かるかもしれない。このとき、ページをめくる頻度から、選手の授業への関与度を評価することができる。オブジェクトが「導入」「方法」「結果」などに関連する場合、「プレイヤーはまず「方法」を読み、次に「結果」を読んで、読むのをやめる傾向がある。」ということが分かるかもしれない。文章の構造に関連付けた動きから、プレイヤーの戦略は、記事を簡潔に把握することであると分かる(**図10**).

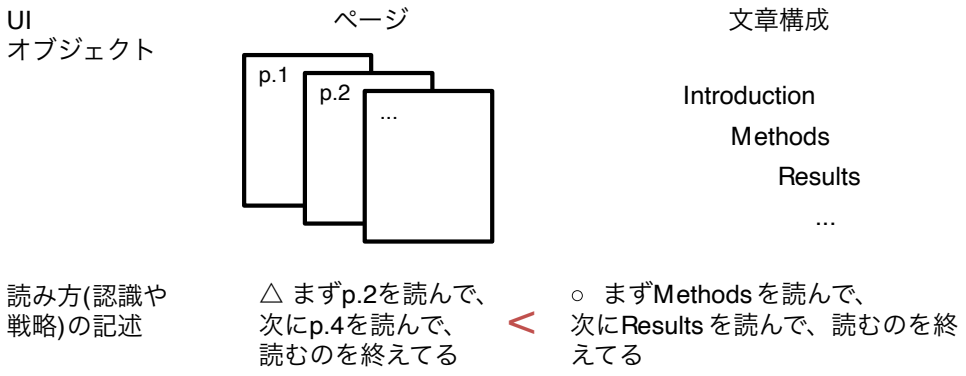



**図10** UIのオブジェクトと操作を記述することばの粒度を合わせる

### 3.1.2 UIオブジェクトの追跡

前記のように導入したオブジェクト、すなわちテキストの断片やパズルのピースを永続して追跡できることは重要である。特に、削除して復活という操作を追跡できることが、試行錯誤の分析には必要である。書き手が削除したテキストを、後から思い出してもう一度書いたとき、これを復活として、つまり試行錯誤の一環として検出するには、テキストの内容的な同定といった処理が必要となる。

プログラミングでは、コメントアウトという手段がよく使われる。ワードプロセッサーにも同様の機能があれば、オブジェクト追跡の課題は緩和されるかもしれない。削除した断片が見えなくなっているだけで、後から復活でき、復活させる前であれば、見え消し表示にもできるといった機能である。

### 3.1.3 秘密

人に、部外者に見せられない、見せたくないある種の文章は、真剣に取り組まれる文章の例であろう。ユーザーのオリジナルなコンテンツの内容を保存しないことは、真剣な作文、真剣な取り組みのプロセスを分析するにあたって重要な要件である。

### 3.1.4 匿名性

秘密と同様に、「誰がプレイしたか」の情報を持たない。各プレイとプレイヤーを結びつける対応表を、アプリケーションの外に置くことは重要である。それを持たなくても、その範囲で有意義な情報を引き出せることが、プロセスを分析するにあたって重要な要件である。

### 3.1.5 システム連携

とはいえ、教育の現場では各プロセス、あるいはプロセスの結果・アウトプットを児童生徒学生を対応付ける必要がある。本研究が提案するアプリケーションを利用したうえで、そのような対応付をシステム的に実現する方法を用意する。

### 3.1.6 データ欠損の検証

ユーザー操作を記録して分析するので、ちゃんと記録できているか、後から検証できる仕組みがあること。これがないと、分析結果の信頼性を評価/保証できない。後の9章でも触れる。

### 3.1.7 マルチ・プラットフォーム

これらアプリケーションを使った実験やワークショップを用意に実施するために、あるいは研究の成果を広く社会に還元するためは、アプリケーションがマルチプラットフォームに対応していることが望ましい。GIGAスクール対応でも、iOS、Windows、ChromeOSなど様々である。

#### 3.1.8 Webアプリケーション

Webアプリケーションであれば、インストール不要で、かつ一定の安全性が保たれる。アドホックな社会人講座、オムニバス授業、体験講義などでも使えるように、インストール不要なアプリケーションが望ましい。企業内のパソコンでは、インストールに許可が必要なところもある。

#### 3.1.9 アクセシビリティ

障害者差別解消法の改正など、我が国は障害のある人もない人も共に暮らせる社会を目指している。本研究のような新しいアプリケーション(技術)を開発し導入を目指す研究は、障害がある人でも、それによって利益を得られるよう努力する必要があるだろう。

マルチ・プラットフォームと併せて考えると、Webアプリケーションとして実装することが有利であろう。

### 3.2 アプリケーションの例

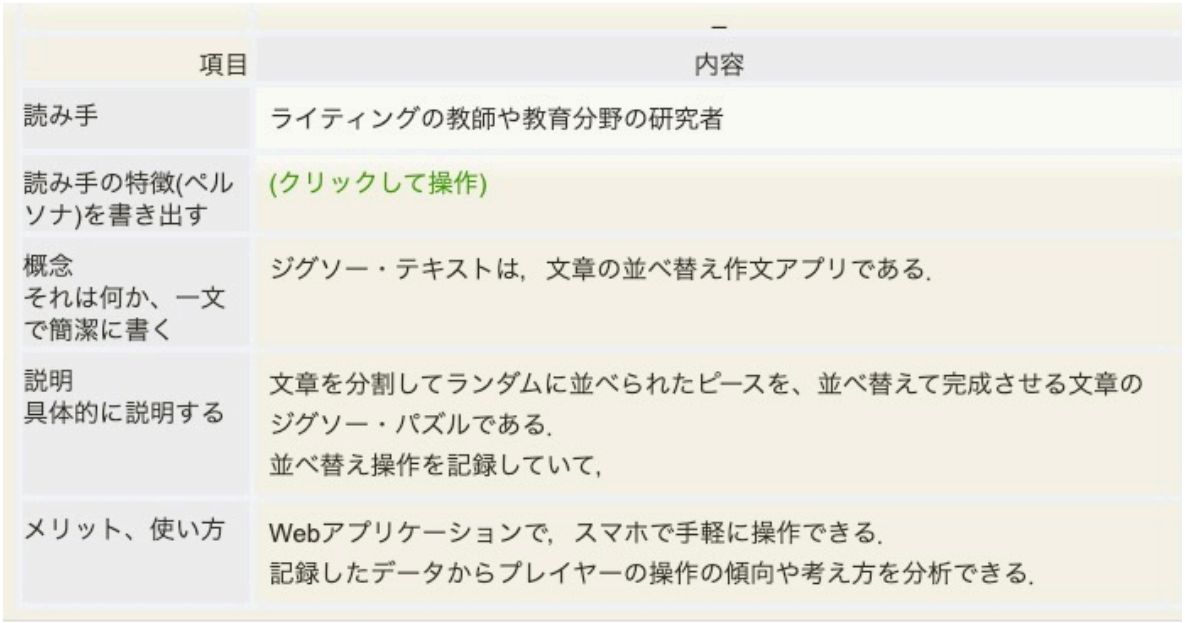
そのような性質を備えたアプリケーションの例を示す。これは、いずれも本研究で開発したものである。どれもWebアプリケーションとして実装されていて、マルチ・プラットフォームで使える。

#### 3.2.1 Topic Writer

Topic Writerはロジック・ツリー、言い換えるとワークシート作文のWebアプリケーションである。ユーザーは、用語説明やビジネスメールなど文章の「型」、すなわちロジック・ツリーを選択し、選択した型に沿って自由に作文する**[17][18][19][20][21]**。

**図11**は、読み手を意識して用語を説明するワークシートを使ってジグソー・テキストを説明する文章を書いている途中である。「読み手」や「概念 それは何か、一文で簡潔に書く」など「項目」列の行見出しが文章の「型」、「ロジック・ツリー」にあたる。「内容」列の「ジグソー・テキストは、文章の並べ替え作文である。」などセルの中身が、ユーザーが書いた文章である。

Topic Writerはユーザーの作文プロセスにおける編集操作を記録している。ユーザーがどのセルを編集したか、セルのIDを記録する。その一方で、見出し項目や、ユーザーが書いた内容は記録しない。しかし、そのとき、編集対象のセルに対応した見出しのIDは併せて記録しているので、記録データを分析するとき、具体的な内容は分からないが「メリット、使い方」を編集したのだということは分かる。



| 項目 | 内容 |
| --- | --- |
| 読み手 | ライティングの教師や教育分野の研究者 |
| 読み手の特徴(ペルソナ)を書き出す | (クリックして操作) |
| 概念<br>それは何か、一文で簡潔に書く | ジグソー・テキストは，文章の並べ替え作文アプリである． |
| 説明<br>具体的に説明する | 文章を分割してランダムに並べられたピースを、並べ替えて完成させる文章のジグソー・パズルである．<br>並べ替え操作を記録していて， |
| メリット、使い方 | Webアプリケーションで，スマホで手軽に操作できる．<br>記録したデータからプレイヤーの操作の傾向や考え方を分析できる． |

**図11**　Topic Writer

Topic Writerは、当初、それぞれが書いた文章をそれぞれのDropboxアカウントに保存していた。後に、ローカルのパソコンなどに直接保存する方式に変更した。

**UIオブジェクト**

Topic Writerでユーザーの思考を記述することばに対応するオブジェクトはワークシートのセルである。セルを編集したことが記録され、セルに書かれたテキストの文字数も記録されるが、テキストそのものは記録されない。具体的なテキストを、ユーザーの思考を記述することばとしていない。

**追跡性**

ワークシートのセルは削除できず、新たに追加できない。セル内のテキストそのものは記録されず、ある編集後のテキストが、以前のテキストと同じかどうかを判定できない。

**秘密**

セルに書かれたテキストは、ユーザーが自分のPCなどに保存することはできるが、分析対象となる測定データには含まれない。また、見出しのテキストそのものも記録に残らない。測定データだけを分析する分析者は、これらを知ることはできない。

**匿名性**

ワークシートを使って編集される文章にはユニークなIDが振られる。これをdocument idと呼ぶ。編集操作はそのdocument idと対応付けて記録される。しかし、document idとユーザーとを結びつける情報をシステムは持たない。Topic Writerを使って授業などを行うときには、受講生にdocument idを提出させる必要がある。

**アクセシビリティ**

Topic Writer は jQuery および jQuery UI[*8] を使って作られている。jQuery UI は W3C の WAI-ARIA[*9] に対応していてアクセシビリティ向上が図られている。これによってワークシートの非同期読み込みもスクリーン・リーダーに対応している。

### 3.2.2 ジグソー・テキスト

ジグソー・テキストは、分割されランダムに並べられた文章のピースを、並べ替えて完成させる文章のジグソー・パズルである。言い換えると、文章の断片群を読解して再構成する並べ替え作文である。Web アプリケーションであり、ドラッグ＆ドロップで並べ替える[22]。Topic Writer とは異なり、文章のテキストは固定で、自分で変えられない。**図 12** の例は「オレオレ詐欺」を説明する記事が 7 個のピースに分割されている。「ひとつは…金銭を要求する。」などがパズルのピースである。**図 13** のジグソー・テキスト 2 は、さらにピースの取捨選択が加わったものである。

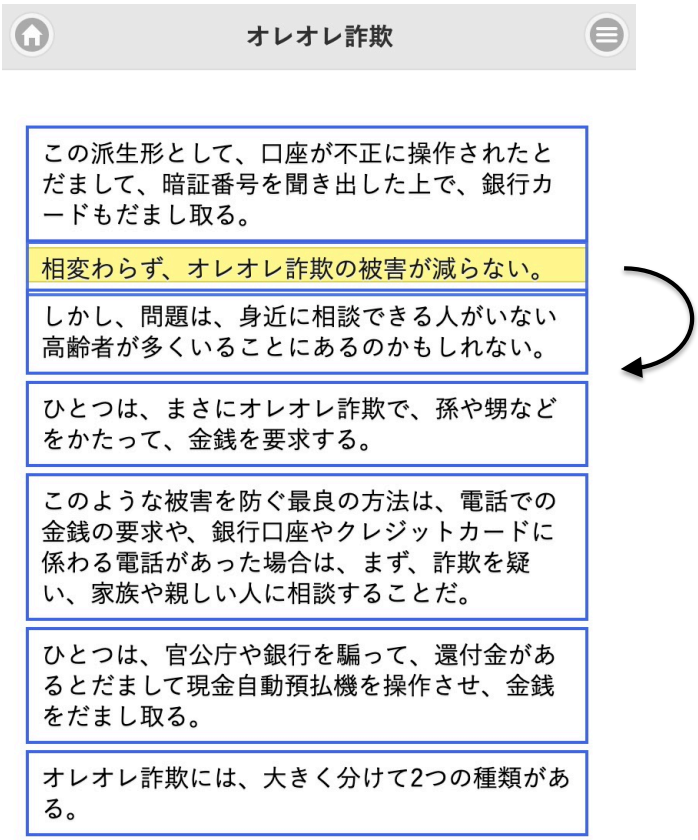



**図 12** ジグソー・テキスト

*8 jQuery UI: https://jqueryui.com/ ↗

*9 ARIA in HTML: https://www.w3.org/TR/html-aria/ ↗

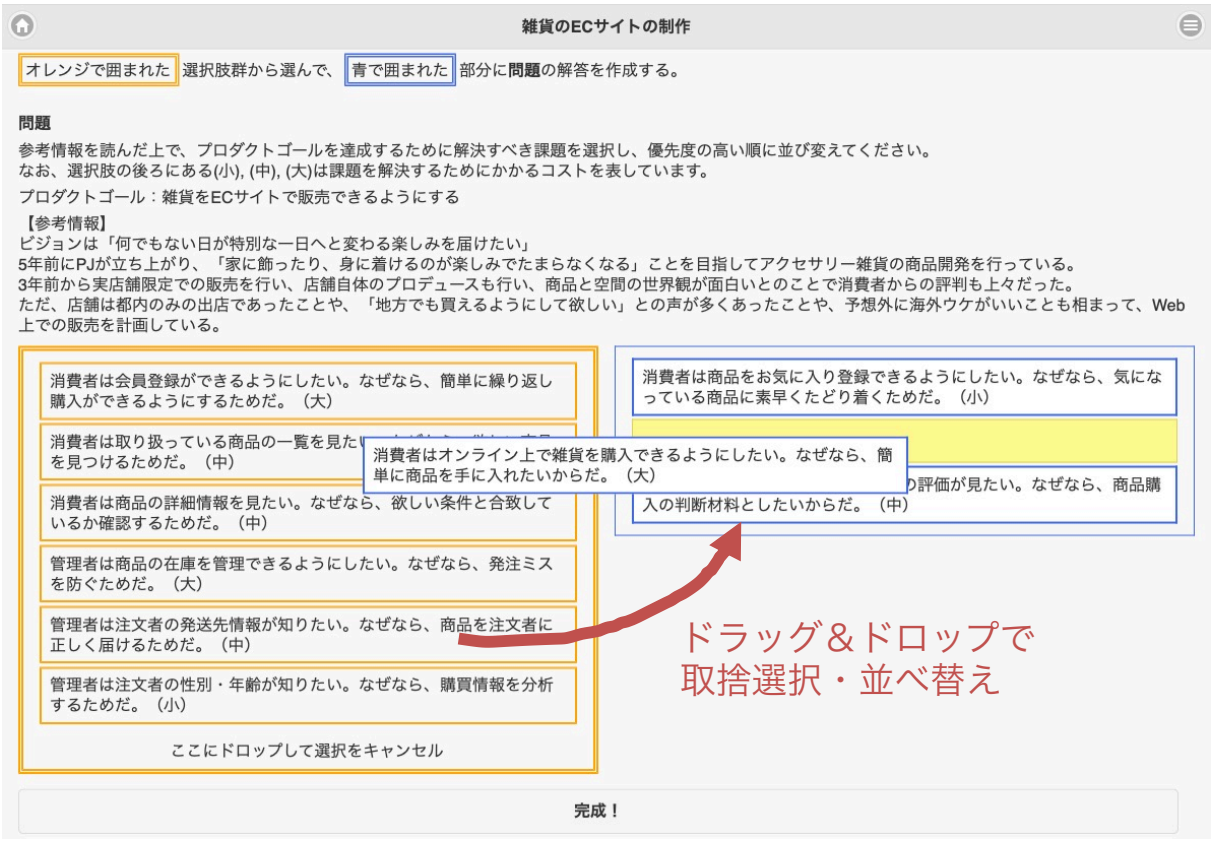



**図13**　ジグソー・テキスト2、取捨選択・並べ替え型

ジグソー・テキストはプレイヤーの並べ替え操作を測定している。まず、パズルを解き始めたときと完成したときに、その時刻と、そのときのピースの並び順が記録される。ピースはドラッグ&ドロップで並べ替えられる。ドラッグし始めたときとドロップしたときに、その時刻と、ドラッグやドロップされたピース、それぞれのときに前後にあったピースや、ピース全体の並び順が、ピースのIDで記録される。ピースのテキストは記録されない。

**UIオブジェクト**

ジグソー・テキストでユーザー(プレイヤー)の思考を記述することばに対応するオブジェクトは、パズル問題のピースである。

**追跡性**

ピースは削除されない。ユーザーが新しく追加することはできない。

しかし、図13のジグソー・テキスト2の場合、左の枠から右の枠にピースを移動することが、任意のものではないが、ピースの追加に相当する。また、右の枠から左の枠にピースを移動することが、ピースの削除に対応すると言ってよい。このように、一定の範囲でコンテンツの追加・削除ができて、一旦は削除した同じピースを再び追加する操作を検出できる。

**秘密**

プレイヤーの操作に伴って、操作対象となったピースのIDが記録されるが、ピースのテキストは記録されない。測定データだけを分析する分析者は、どんなピースが操作されたのか知ることはできない。

ジグソー・テキストでは、同じパズル問題を解いた皆が、同じピースを並べ替える。そこで、ピースのIDに基づいた分析で並べ替え操作の傾向を見出そうというアプリケーションである。

**匿名性**

各プレイにはIDが割り当てられ、これをdocument idと呼んでいる。呼び名は、先に開発され使われたTopic Writerを引き継いだものである。ジグソー・テキストはプレイヤーの情報を持たない。ジグソー・テキストを授業で使う場合、受講生にdocument idを提出させるか、次のシステム連携で収集することで、ジグソー・テキストの外部で受講生とdocument idとを対応付ける必要がある。

**システム連携**

ジグソー・テキストをiframeを使ってWebページに埋め込むと、ジグソー・テキストとの間でWindow.postMessage()を使って通信でき、プレイのdocument idなどを受け取れる。これによって受講生のdocument idを収集する事例がある(**図114**)。

**アクセシビリティ**

ジグソー・テキストはjQueryおよびjQuery UIを使って作られている。Webにはスクリーン・リーダーで操作する仕組みが用意されているが、Webアプリのドラッグ&ドロップは、スクリーン・リーダーで操作できない。(一方、iPhoneのアプリ、例えばリマインダーでは、VoiceOverというスクリーン・リーダーで操作して、アイテムをドラッグ&ドロップで並べ替えることができる。)ジグソー・テキストには、ドラッグ&ドロップ以外の方法でピースを並べ替えるUIを開発した。ドラッグ&ドロップによるUIと、同じパズル問題データを読み込んでプレイできる。これについては10章で詳しく述べる。

### 3.2.3 ジグソー・コード

ジグソー・コードはジグソー・テキストのプログラム・コード版である。操作の仕方は同様だが、例えば、フォントがコーディング用のものになっているなどの違いがある(**図14**)。

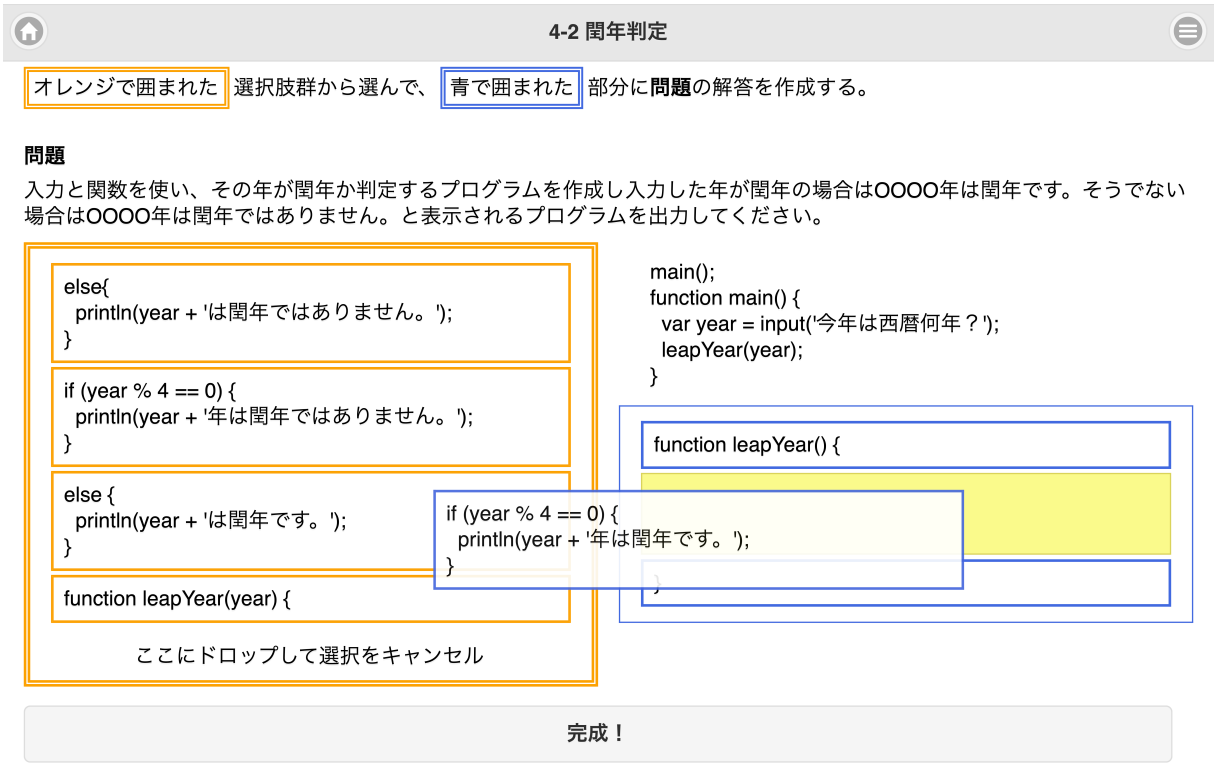



**図14　ジグソー・コード**

UIオブジェクトなどについては、ジグソー・テキストと同様である。

### 3.2.4　V字エディタ

V字エディタはTopic Writerの発展型で、ロジック・ツリーのノード、あるいは文章の断片の配置を変えて編集できる。ユーザー操作を測定していることは、Topic Writerと同様である。通常は画面や紙面上で上から下にリニアにI字型に配置される文章の部分たちを、画面上でV字型あるいはΛ字型に折り曲げて配置して編集できるV字エディタを開発する。これは、ソフトウェア開発のVモデル[23]を参考にしたものである。Webアプリケーションとして実現している。

UIオブジェクトなどについては、Topic Writerと同様である。

V字エディタは一風変わっているので、使い方について、他のアプリケーションよりも細かい説明が必要であろう。V字エディタには4つのビューがある。最終的な出力である1段落に見える「インライン」(**図15**)、それをセクション構造に展開した「通常」(**図17**)、通常ビューをV字に折り曲げた「V字」(**図18**)、V字を上下逆さまにした「Λ字」(**図19**)である。

ファイル　表示　20220312 ce164 jigsaw code v3.json

**プログラム・コードの並べ替えパズル「ジグソー・コード」の性能改善**

プログラム・コードの並べ替えパズル「ジグソー・コード」はSaaS (Software as a Service)のGoogle App Engineを利用している．2020年度の後期にジグソー・コードを使って，約230人の受講生に演習を実施したところ，問題を読み込んで操作を始められるまでに2分程度かかるという問題が起きた．処理の応答が遅いと頻繁に再読み込みする操作が見られ，状況を悪化させてしまうことも判った．また，操作ログの一部が保存されなかったことも判った．そこで，後期の後半に設計を見直して性能改善を図った．全てを1つのサービスで処理していたものを分割して，操作ログ保存などを別サービスで処理するようにした．時間のかかる処理を早めにあきらめタイムアウトするようにした．操作ログの保存処理はリトライし，重複する可能性が生じるので各操作をシリアル番号で識別した．授業時間には各サービスのインスタンス数を増やしておく運用とした．その結果，インスタンス数が増えたが，従来よりも処理の応答時間が短くなった．タイムアウトが発生しているが頻度は少なかった． 操作ログ保存はリトライしていることを確認できた．2021年度は性能上の問題は起きなかった．本研究の全体は受講生の操作をモニターしてリアルタイムに介入することを目指している．登録だけでなく検索が加わっても大丈夫なように，操作ログの処理を改善する．

596文字

**図15**　V字エディタ: 最終的な出力としては1つの段落である。

ファイル　表示　20220312 ce164 jigsaw code v3.json

**プログラム・コードの並べ替えパズル「ジグソー・コード」の性能改善**

**1.** 背景 プログラム・コードの並べ替えパズル「ジグソー・コード」はSaaS (Software as a Service)のGoogle App Engineを利用している．2020年度の後期にジグソー・コードを使って，約230人の受講生に演習を実施したところ，問題を読み込んで操作を始められるまでに2分程度かかるという問題が起きた．処理の応答が遅いと頻繁に再読み込みする操作が見られ，状況を悪化させてしまうことも判った．また，操作ログの一部が保存されなかったことも判った． **2.** 目的 そこで，後期の後半に設計を見直して性能改善を図った． **3.** 研究方法 全てを1つのサービスで処理していたものを分割して，操作ログ保存などを別サービスで処理するようにした．時間のかかる処理を早めにあきらめタイムアウトするようにした．操作ログの保存処理はリトライし，重複する可能性が生じるので各操作をシリアル番号で識別した．授業時間には各サービスのインスタンス数を増やしておく運用とした． **4.** 結果 その結果，インスタンス数が増えたが，従来よりも処理の応答時間が短くなった．タイムアウトが発生しているが頻度は少なかった．操作ログ保存はリトライしていることを確認できた． **5.** 考察・結論 2021年度は性能上の問題は起きなかった． **6.** おわりに・今後 本研究の全体は受講生の操作をモニターしてリアルタイムに介入することを目指している．登録だけでなく検索が加わっても大丈夫なように，操作ログの処理を改善する．

596文字

**図16**　V字エディタ: しかし、内部的には構造があり、見出しを付けられる。これは論文の抄録の例。

ファイル　表示　20220312 ce164 jigsaw code v3.json

**プログラム・コードの並べ替えパズル「ジグソー・コード」の性能改善**

**1. 背景**

プログラム・コードの並べ替えパズル「ジグソー・コード」はSaaS (Software as a Service)のGoogle App Engineを利用している．2020年度の後期にジグソー・コードを使って，約230人の受講生に演習を実施したところ，問題を読み込んで操作を始められるまでに2分程度かかるという問題が起きた．処理の応答が遅いと頻繁に再読み込みする操作が見られ，状況を悪化させてしまうことも判った．また，操作ログの一部が保存されなかったことも判った．

**2. 目的**

そこで，後期の後半に設計を見直して性能改善を図った．

**3. 研究方法**

全てを1つのサービスで処理していたものを分割して，操作ログ保存などを別サービスで処理するようにした．時間のかかる処理を早めにあきらめタイムアウトするようにした．操作ログの保存処理はリトライし，重複する可能性が生じるので各操作をシリアル番号で識別した．授業時間には各サービスのインスタンス数を増やしておく運用とした．

**4. 結果**

その結果，インスタンス数が増えたが，従来よりも処理の応答時間が短くなった．タイムアウトが発生しているが頻度は少なかった．操作ログ保存はリトライしていることを確認できた．

**5. 考察・結論**

2021年度は性能上の問題は起きなかった．

**6. おわりに・今後**

本研究の全体は受講生の操作をモニターしてリアルタイムに介入することを目指している．登録だけでなく検索が加わっても大丈夫なように，操作ログの処理を改善する．

596文字

**図17**　V字エディタ: インラインの構造をブロックに展開した画面。

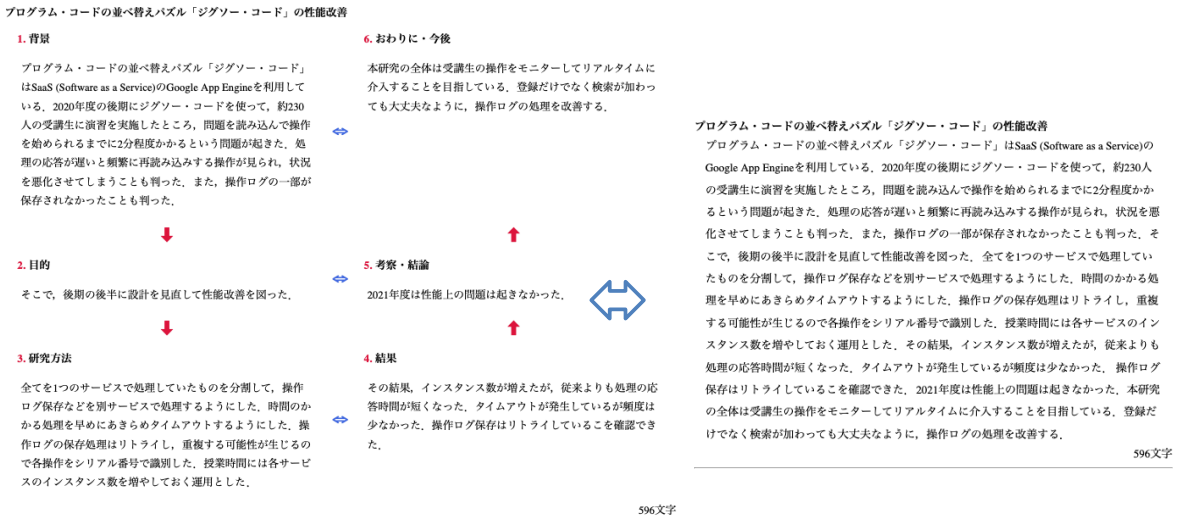



**図18**　V字エディタ: ブロックを左上から下へ、下から折り返して右上に、V字に配置した画面。水平に対応するブロックが内容的に対応するように編集する。

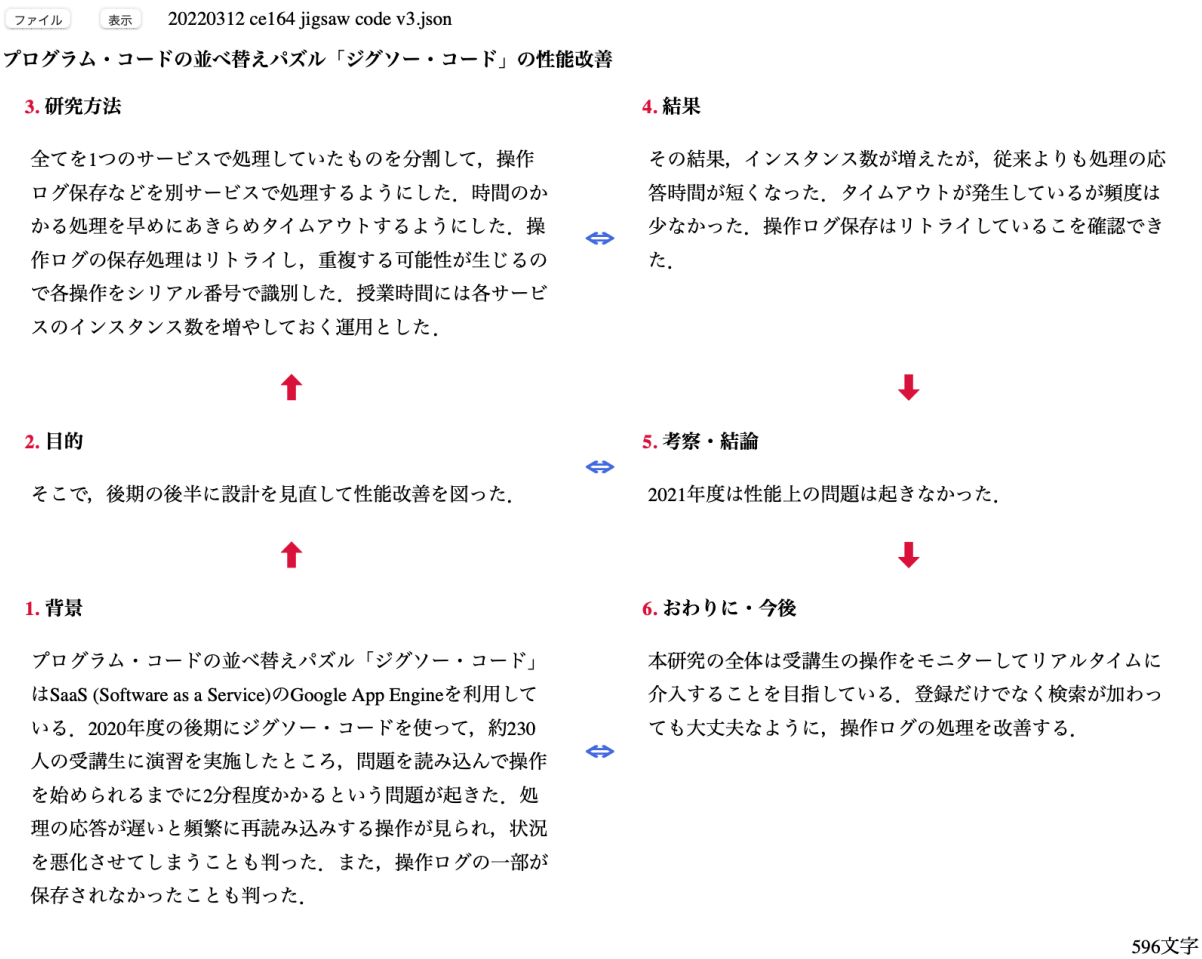
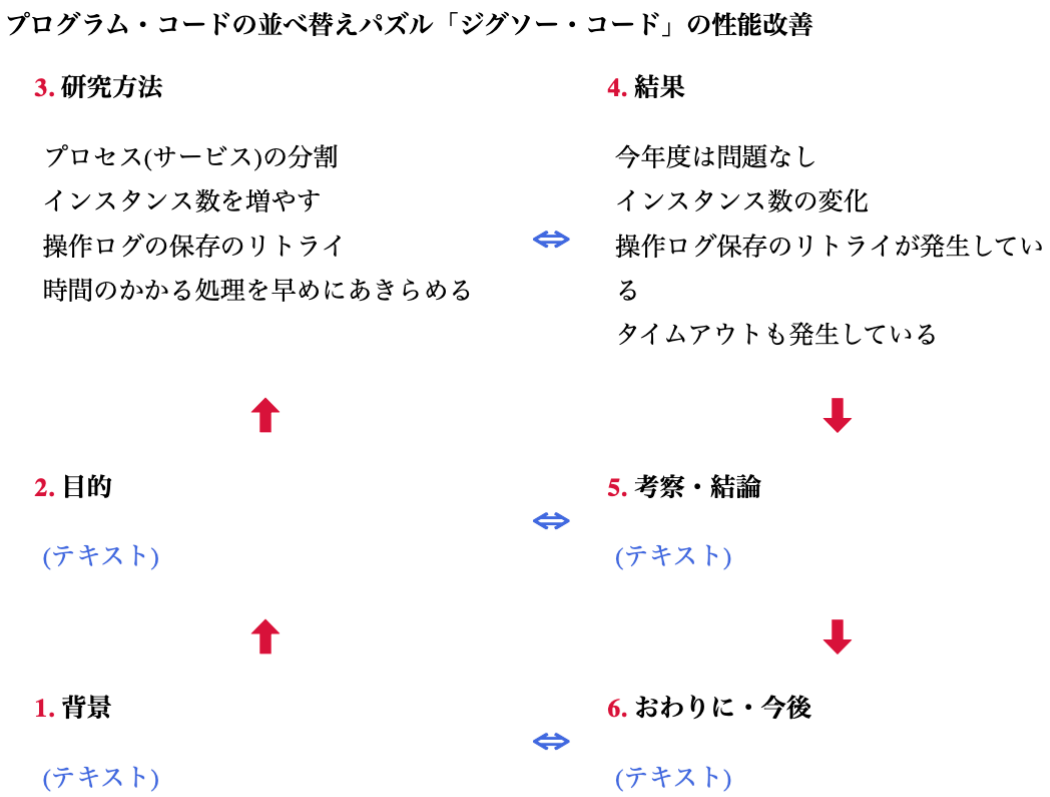



**図19**　V字エディタ: 左下→上→右下へと、Λ字に配置した画面。

Λ字ビューは、図のワークシートであれば、研究方法と結果から書き始めるように促すものである。まず自分がやったことと結果を書き始める。キーワードレベルででもよしとする(**図20**)。実験が済んでいるなら書けるし、淡々と書きやすいであろう。日本語としての読みやすさ、滑らかさなどは、最後にインライン表示で整えるのがよいであろう。




112文字

**図20**　V字エディタ: Λ 配置での書き始め。通常の感覚で左上から書いていくと、実際は途中から書いていくことになる。

### 3.2.5　15パズル

15パズルは、いわゆる15パズルをWebアプリケーションとして実装したものである。遊び方は従来の15パズルと同様である(**図21**)。通常の15パズルと異なるのはユーザー操作を測定していることである。

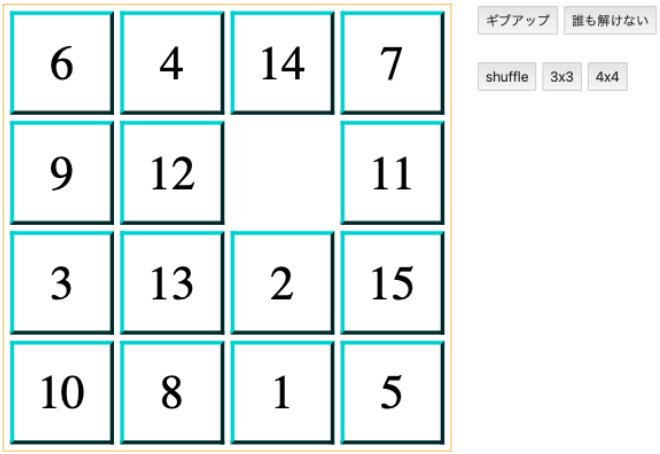



**図21**　15パズル

15パズルのUIオブジェクトなどは、ジグソー・テキストと同様である。

### 3.3 結論

例にあげたアプリケーションは、V字エディタのような複雑なUIのものも含めて、UIを構成するオブジェクト -- ジグソー・テキストのピース(断片)やTopic Writerのセルや15パズルのピース -- は、分析によってユーザーの思考、例えば試行錯誤を記述したいことばと対応している。測定し分析しようとするのはセルやピースであるが、これらは削除されずライフサイクルにわたって追跡できる。ユーザー操作を記録しているが、ユーザー個別のコンテンツ -- Topic Writerに入力したテキストなど -- は記録しない。どれもWebアプリケーションである。

このようなバラエティを見れば、他のアプリケーションの発想も生まれるだろう。今後、同様の性質を備えた新しいアプリケーションが開発されたり、既存のアプリケーションがエンハンスされたりすることを期待する。

# 2部

# 4 時間的な共起分析

ここでは時間的な共起分析(temporal co-occurrence analysis)を論じる。これは、前記のUIオブジェクトが前後して操作対象となる頻度を分析するものである。アウトプットのテキストにおいて語が近くに出現する頻度を分析する、従来の共起分析にならった分析手法である。この分析によって、ユーザー(プレイヤー)にとって互いに関連するピースのペアを見いだせた。

近年求められる思考力とは、ときには答えのない課題にも取り組む能力である。われわれは、作文、プログラミング、パズルなどに取り組む思考過程の特徴を可視化する手法として、編集操作の時間的な共起分析を提案する。本手法は、いくつかの操作が時間的に近くで行われる頻度を集計する。このように集計する動機は、テキスト分析における語の共起と同様である。ある複数の操作が頻繁に時間的に近くで行われるとき、それらの操作は何らかの定型操作であったり、考え方の類似性の手がかりになると解釈できる。本稿では、ワークシート作文やプログラミング・パズルを解く過程を記録したデータの具体例をあげて、本手法が捉えるデータの特徴を示す。課題を解く一連の操作全体を対象とする分析手法と比較すると、部分を集計する本手法は、試行錯誤を含むような思考過程に対して有効である。

## 4.1 はじめに

思考力が教育において重要視され、「答えが1つではない課題」や「答えのない課題」や、「理解してること・できること」を使って「未知の状況」にも対応できることが期待されている[6] **[24]**。

Computer-based testing (CBT)の導入が進み、従来の紙ベース試験とは異なる手法で思考力を評価できると期待され、研究されている**[25]**。近年注目されているプログラミング教育はコンピューター上で行われるので、紙ベースの授業よりも容易に、児童・生徒・受講生が課題に取り組むプロセスを測定してデータを得られるだろう。文献**[26]**では、プログラムの穴抜き問題を、解答の所要時間などでクラスタリングしている。

### 4.1.1 着想

思考力を論じるとき「考え方」ということばが使われる。このことは「考えることに方法がある」と認識されている現れであろう。考えることに方法があるなら、考えながら何かを操作するとき、考える方法が操作のパターンに表れると期待できる。ITを利用したパズル・アプリケーションであれば、アプリの操作を記録(測定)する仕組みを、われわれ自身が適切に設計することで、解く考え方がパズル操作の測定データに反映されると期待できる。分・秒単位で行われる操作を測定・分析することで、当人も意識しない情報を抽出できるかもしれない。

### 4.1.2 課題

この手法で考え方を研究するとき、分析対象の測定データはパズル操作の時系列データとなり、これに応じた分析手法が必要となる。答えのない・未知の課題に対して、理解してること・できることを動員して、試行錯誤して取り組むプロセスの特徴を抽出できる分析手法である。

### 4.1.3 本章の構成

本章では、まず関連する研究をとりあげて、提案手法の位置づけを示す。次に、提案手法を示すが、共起概念そのものは古くから使われているものであり、概要を示すにとどめる。次に、ユーザーの編集操作を記録・測定するアプリを具体的に示す。そして、それらアプリが測定した具体的な系列データをあげて、提案手法がそこからどのような情報を抽出するかを示す。最後に、提案手法と他の手法との違いを考察する。

## 4.2 関連研究

### 4.2.1 従来のテキスト分析

テキスト分析における共起分析といえば、思考プロセスによって書かれた・産出されたテキストにおける、語同士の共起関係を分析することである。単純化して言ってしまえば、文字列上の

位置に基づいてテキストを分析する[27]。テキストの意味的な内容が時間の経過を含み、その経過を踏まえて分析する研究もあるが、テキストそのものが書かれるプロセスを共起の観点から分析するものではない

本稿で提案する手法は、テキストを書く・産出するプロセスに着目し、それを通して完成テキストの部分間の関係を分析するものである。単純化して言ってしまえば、時間軸上の位置に基づいてテキストを分析するものである。

### 4.2.2 系列パターンのクラスタの共起分析

時系列データに共起概念を導入した先行研究はある。文献[28][29]は、事象のクラスタを共起の観点から分析して、燃料電池の損傷間および地震間の相互作用を抽出する手法を提案している。

ここで導入された概念は本稿の共起を含む。すなわち、この文献が定式化した共起関係は事象の集合同士の共起を扱っており、この集合が要素を1つだけ含み互いに素な集合とすれば、本稿の共起概念と一致する。

本稿で提案する手法は、そのような汎用性には欠けるが、よりシンプルな共起概念を作文やプログラミングといった人間の知的活動に適用するものである。また、本章で取り上げる事例では、1つの系列に含まれる事象の数、すなわち、1つの文章の編集や1つのプログラミング・パズルを解くプロセスに含まれる編集操作の数はせいぜい数10個と少ない。本章では、共起分析の結果の図表だけでなく、系列データそのものをとりあげて、共起の組が系列データ全体の中でどのように現れるかを見る。

### 4.2.3 Parson's Problemの分析

4.4.1で述べるジグソー・コードによく似たParason's Problemを解く過程を分析した研究がある[30]。この研究では、プログラム・コードの断片が正しい位置に置かれた時間的な順序について、最適な順序を分析している。**図26**で示すように、このような並べ替えプロセスは試行錯誤を含み、同じ断片・ピースを何度も動かすことがある。この研究では、これら試行錯誤に該当するデータを落としてしまうことが課題だと述べられている。

## 4.3 編集操作の共起分析

われわれは編集操作の時間的な共起分析を提案してきた。本手法は、いくつかの操作が時間的に近くで行われる頻度を集計する。このように集計する動機は、テキスト分析における語の共起と同様である。ある複数の操作が頻繁に時間的に近くで行われるとき、それらの操作の間には意味のある関係があり、例えば、何らかの定型操作を構成したり、考え方の類似性の手がかりと解釈できる。

### 4.4　編集操作を測定するアプリケーション

われわれの取り組みから、ユーザーの編集操作を記録・測定するアプリを具体的に示す。ワープロ、表計算、ドローソフトなど、どのようなアプリであっても、同様の測定は可能である。

#### 4.4.1　並べ替え作文のジグソー・テキスト、並べ替えプログラミングのジグソー・コード

ジグソー・テキストは、分割されランダムに並べられた文章のピースを、並べ替えて完成させる文章のジグソー・パズルである。言い換えると、文章の断片群を読解して再構成する並べ替え作文である。Webアプリケーションであり、ドラッグ＆ドロップで並べ替える[22]。**図22**の例は「オレオレ詐欺」を説明する記事が7個のピースに分割されている。「ひとつは…金銭を要求する。」などがパズルのピースである。**図23**は、さらにピースの取捨選択が加わったものである。

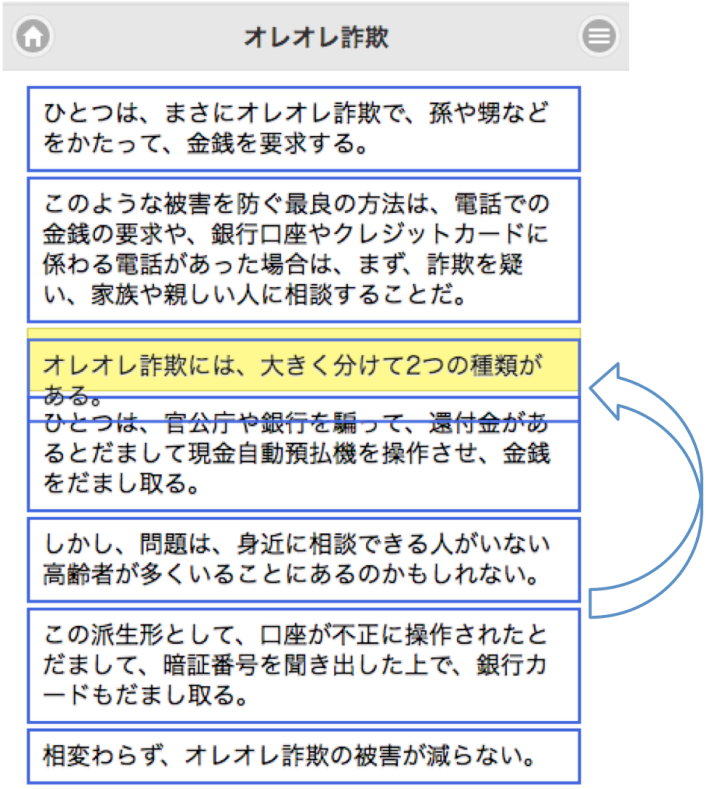



**図22**　並べ替え作文のジグソー・テキスト

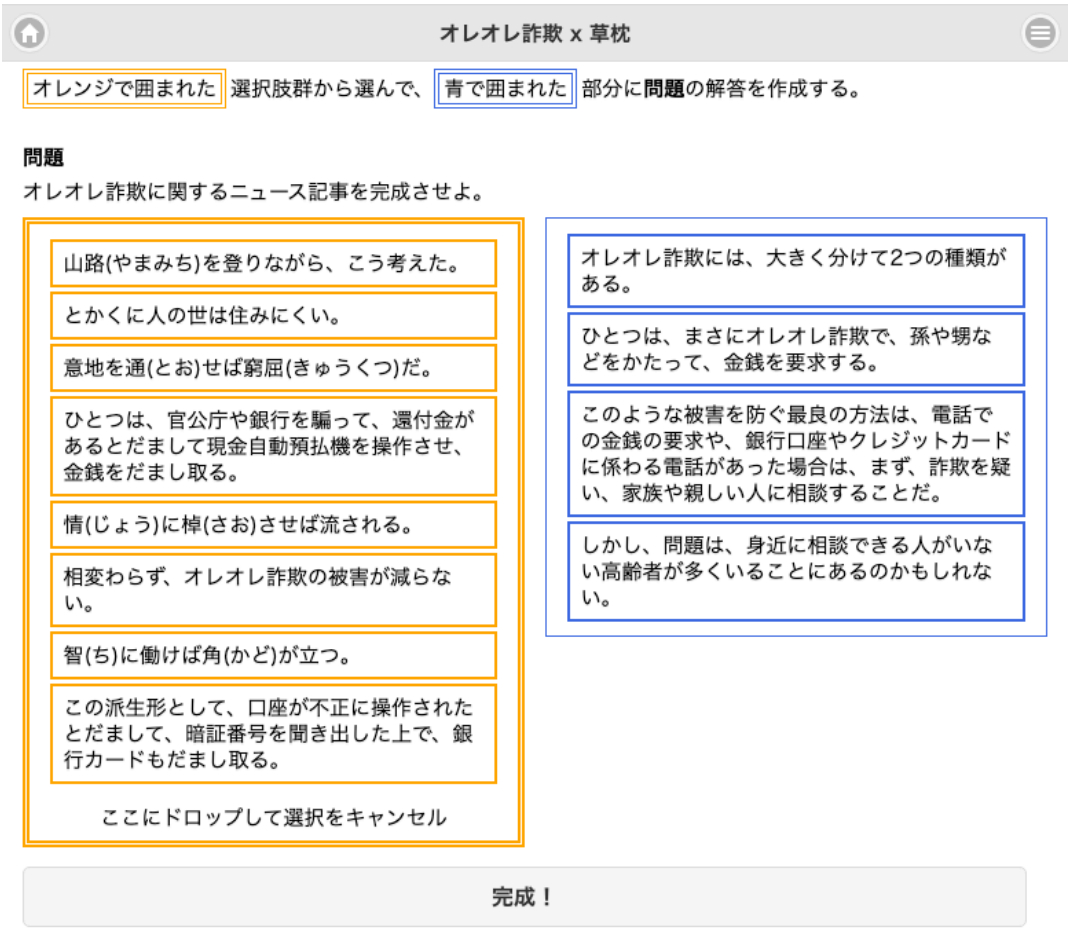



**図23**　ピースを取捨選択して並べ替えるジグソー・コード2

ジグソー・テキストはプレイヤーの並べ替え操作を測定している。まず、パズルを解き始めたときと完成したときに、その時刻と、そのときのピースの並び順が記録される。ピースはドラッグ&ドロップで並べ替えられる。ドラッグし始めたときとドロップしたときに、その時刻と、ドラッグやドロップされたピース、それぞれのときに前後にあったピースや、ピース全体の並び順が記録される。**図24**は、IDがs3のピースをドラッグし始めたときと、ドロップしたときに記録されるデータを示している。s3などはピースに割り当てられたIDである。IDの添字の1、2、…は必ずしも正解の順序に対応しない。

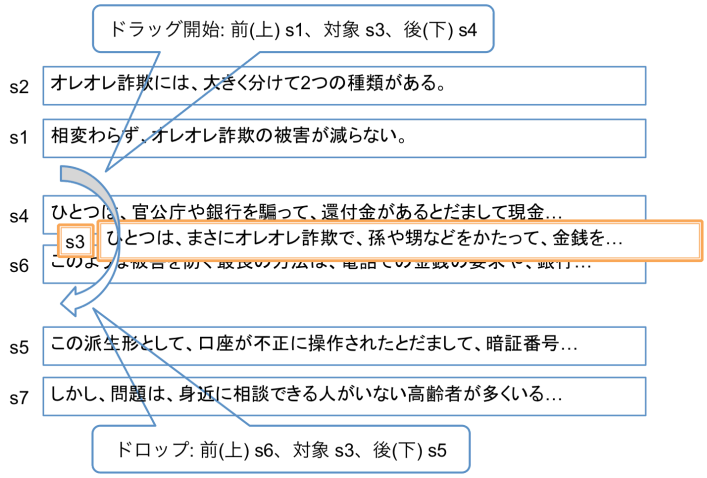



**図24**　ジグソー・テキストによる並べ替え操作の測定

ジグソー・テキストと同様に、プログラムのコード断片を並べ替えて完成させるジグソー・コードがある**[31]**。取捨選択して並べ替えるジグソー・コード2もある。具体例については、後の**図34**で示す。

### 4.4.2　ロジック・ツリー作文の Topic Writer

Topic Writerは、ロジック・ツリーによる文章エディターである。ユーザーは、用語説明やビジネスメールなど文章の「型」、すなわちロジック・ツリーを選択し、型に沿って作文する[17][18][19][20][21]**[32]**。


**図25**　ロジック・ツリー作文の Topic Writer

**図25**は、読み手を意識して用語を説明するワークシートを使ってジグソー・テキストを説明する文章を書いている途中である。「読み手」や「概念 それは何か、一文で簡潔に書く」など「項目」列の行見出しが文章の「型」、「ロジック・ツリー」にあたる。「内容」列の「ジグソー・テキストは、文章の並べ替え作文である。」などセルの中身が、ユーザーが書いた文章である。

Topic Writerはユーザーの作文プロセスにおける編集操作を記録している。ユーザーがどのセルを編集したか、セルのIDを記録する。その一方で、ユーザーが書いた内容は記録しない。しかし、そのとき、編集対象のセルに対応した見出しも併せて記録しているので、記録データを分析するとき、具体的な内容は分からないが「メリット、使い方」を編集したのだということは分かる。

### 4.5 系列データと共起分析

4.4節のアプリが測定した具体的な系列データをあげて、提案手法が共起に基づいてそこから抽出する情報を示す。まず系列データを示し、提案手法が抽出する情報を示す。さらに、本手法の有益さとして、その情報をどのように解釈できるかを示す。実際の解釈の妥当性については、個々の研究や実践において、実験・観察のデザインや他の情報も加えて判断されるべきである。それは本稿の範囲外である。この事情は他の分析手法と同様である。

#### 4.5.1 ジグソー・テキスト、ジグソー・コードの測定データ

パズルを解くプレイは、個々のピースをドラッグ&ドロップで移動する操作それぞれを事象として、それら事象の時系列で表せる。個々の事象を、そのとき移動対象となったピースのIDで表すことにする。

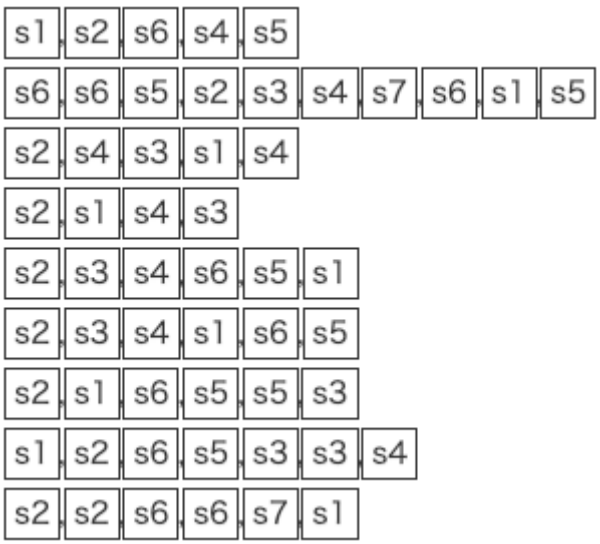



**図26** ジグソー・テキストを解くプレイの系列データ

あるグループの9人が図22のパズルを解いたときのデータは図26のようになる。図の行が個々のプレイに該当し、完成までの所要時間は1分から3分程度である。プレイは左から右に進んでいき、全部で9個のプレイが表示されている。s1などが「ピースs1をどこかへ動かした」という事象である。系列の長さは動かした回数であるが、正解に至る回数がさまざまであると分かる。同じピースを複数回動かしたプレイもある。1つのプレイの中で一度も動かされなかったピースもある。

図26の系列データを、2-gramで有向の共起関係をマトリックスで表示したのが**図27**である。

| n \ n+1 | s1 | s2 | s3 | s4 | s5 | s6 | s7 |
|---|---|---|---|---|---|---|---|
| s1 | 0 | 2 | 0 | 2 | 1 | 2 | 0 |
| s2 | 2 | 1 | 3 | 1 | 0 | 3 | 0 |
| s3 | 1 | 0 | 1 | 4 | 0 | 0 | 0 |
| s4 | 1 | 0 | 2 | 0 | 1 | 1 | 1 |
| s5 | 1 | 1 | 2 | 0 | 1 | 0 | 0 |
| s6 | 1 | 0 | 0 | 1 | 5 | 2 | 1 |
| s7 | 1 | 0 | 0 | 0 | 0 | 1 | 0 |

**図27**　ジグソー・テキストの系列データの共起マトリックス

行見出しがN番目に動かしたピースのID、列見出しがその次N+1番目に動かしたピースのIDである。s3の後でs4を動かした事象が4回あった、などが分かる。

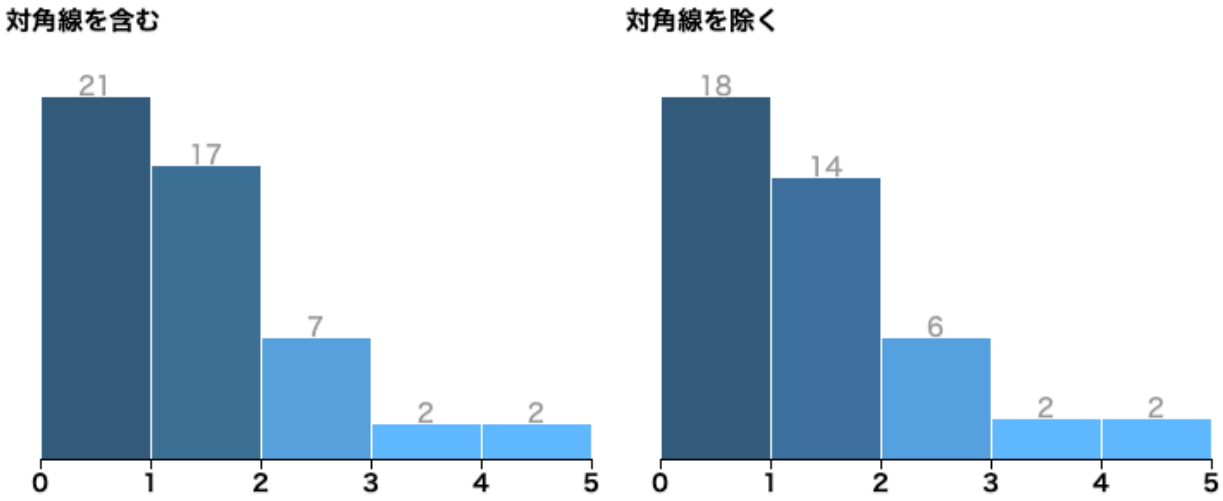



**図28**　共起マトリックスのセルの値のヒストグラム

図27の共起マトリックスのセルの値の頻度を**図28**に示す。図中の対角線とは、共起マトリックスの左上から右下への対角線のことである。対角線上の値は、同じピースを続けて動かしたことを意味し、操作ミスの可能性もあるので、対角線を除いた頻度も併せて表示した。この例では頻度の分布にあまり違いがない。

対角線を除いた場合、セルの値の平均が1、標準偏差が1.2である。図27は、これに基づいて、平均 + 標準偏差より値が大きいセルを黄色、平均 + 標準偏差x2よりも値が大きいセルを赤で強調している。この強調は、頻度の高い共起関係を抽出したことになる。この例では、共起頻度のヒストグラムが右肩下がりになっていて、強調されたセルには意味がありそうである。

図24から、ピースIDとピースの内容とを関係付けられる。共起分析の結果に、このグループが正解者たちであることやピースの内容を加えて、人間(分析者)が解釈すれば、s2は「…2つある」、s3は「ひとつは…」、s4は「ひとつは…」となっているので、プレイヤーたちは「2つある、ひとつは…、ひとは…」と構文的に読解して並べ替えた可能性があるだろう。この例では、共起分析は有益な示唆を与えているといえよう。

| n \ n+1 | s1 | s2 | s3 | s4 | s5 | s6 | s7 |
|---|---|---|---|---|---|---|---|
| s1 | 2 | 3 | 0 | 1 | 2 | 5 | 3 |
| s2 | 3 | 1 | 3 | 6 | 3 | 1 | 0 |
| s3 | 3 | 1 | 1 | 3 | 3 | 0 | 3 |
| s4 | 1 | 0 | 4 | 1 | 6 | 3 | 2 |
| s5 | 5 | 0 | 1 | 2 | 4 | 4 | 4 |
| s6 | 0 | 1 | 3 | 2 | 1 | 4 | 6 |
| s7 | 3 | 1 | 0 | 0 | 5 | 3 | 4 |

**図29**　ジグソー・テキストの系列データの共起マトリックス、その2

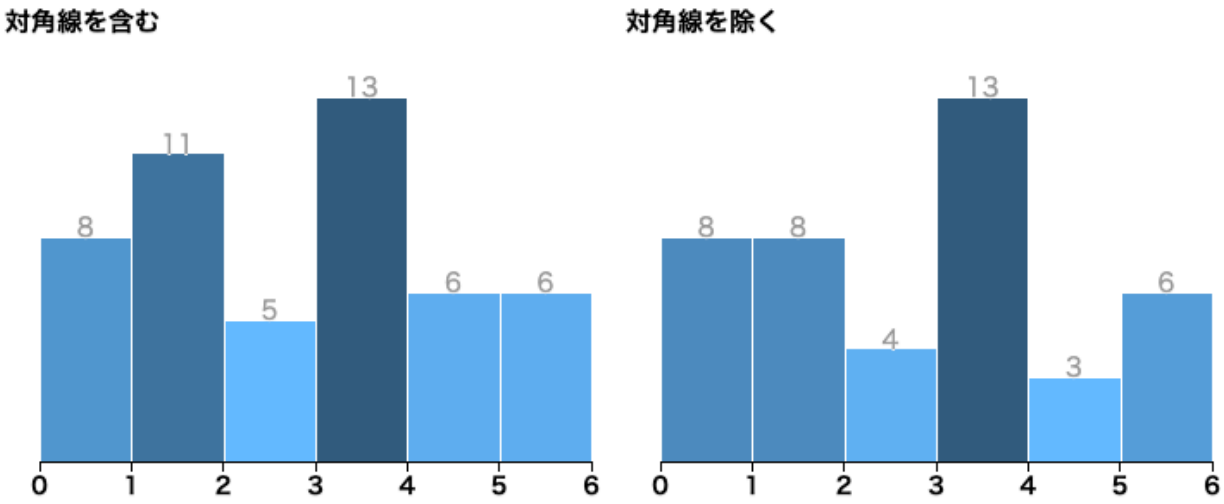



**図30**　共起マトリックスのセルの値のヒストグラム、その2

一方、**図29**および**図30**は、別のグループ21人が同じ「オレオレ詐欺」パズルを解いたときの共起マトリックスと、そのセル値のヒストグラムである。所要時間は1分から5分である。さきのグループの図27および図28と比べて、共起頻度の特徴が弱いように思える。さきの結果と比べてヒストグラムの勾配が緩く、赤のセルがない。

この共起分析の結果は、さきの結果と比べると分析にあまり貢献しない、あるいは特徴が弱いことを示すと言えるだろう。このグループは、さきのグループと同時にパズルを解いた、不正解者グループである。

### 4.5.2　Topic Writerの測定データ

Topic Writerで、図25の、読み手を意識した用語説明のワークシートを使って作文したときの、共起マトリックスが**図31**である。6件の作文について集計されている。90分の授業後にも編集することが許されており、所要時間は10分から2時間である。cc_37などは図27のピースと同様に、ここでは記入枠のIDである。「other」とある行・列は、これら枠以外への編集操作である。

| n \ n+1 | cc_37 | cc_47 | cc_50 | cc_13 | cc_16 | other |
|---|---|---|---|---|---|---|
| cc_37 | 9 | 6 | 0 | 0 | 0 | 0 |
| cc_47 | 0 | 18 | 5 | 0 | 2 | 0 |
| cc_50 | 1 | 0 | 11 | 8 | 1 | 0 |
| cc_13 | 0 | 0 | 3 | 26 | 6 | 0 |
| cc_16 | 0 | 1 | 1 | 4 | 13 | 2 |
| other | 0 | 0 | 0 | 0 | 0 | 12 |

**図31**　Topic Writerの編集操作の共起マトリックス

| 項目 | 内容 |
| --- | --- |
| 読み手 | cc_37 |
| 読み手の特徴(ペルソナ)を書き出す | cc_47 |
| 概念<br>それは何か、一文で簡潔に書く | cc_50 |
| 説明<br>具体的に説明する | cc_13 |
| メリット、使い方 | cc_16 |

**図32**　「用語説明 読み手」ワークシートの記入欄のID

図31が、図27などのジグソー・テキストの場合と異なるのは、左上から右下への対角線およびその右隣のセルである。これらセルの値すなわち共起頻度がとても高い。Topic Writerもジグソー・テキストも同じ「作文」ということばでアプリを説明しているが、測定された系列データの性質が異なる場合があると分かる。

この共起マトリックスの解釈としては、対角線のセルは同じ枠を続けて編集したこと、右隣のセルは1つ隣の枠へ移動して編集したことに対応することから、ユーザーが上から下へ書いていったこと示すと考えられる。すなわち、ごく当たり前に思われている作文パターンを、データから読み取ることができる。

図31で、対角線と対角線の右隣とother行・列を除いたセルの値の平均値は0.6、標準偏差が1.6、**平均** + **標準偏差** × 2 = 2.6である。**平均** + **標準偏差** × 2を超えるセルは、さきの当たり前の作文パターンとはズレて特徴的である。枠の見出しも考慮した解釈としては、「概念」、「説明」、「メリット、使い方」の3枠の書き分けを考えたのであろう。

### 4.6　系列全体を対象とする分析との違い

最適な手順をマイニングするといった、系列データ全体を対照とする分析との違いを考察する。とはいえ、そのような手法はさまざまであるし、それぞれ工夫はあるので、原理的な観点から比較する。

### 4.6.1 高頻度の共起組の出現位置

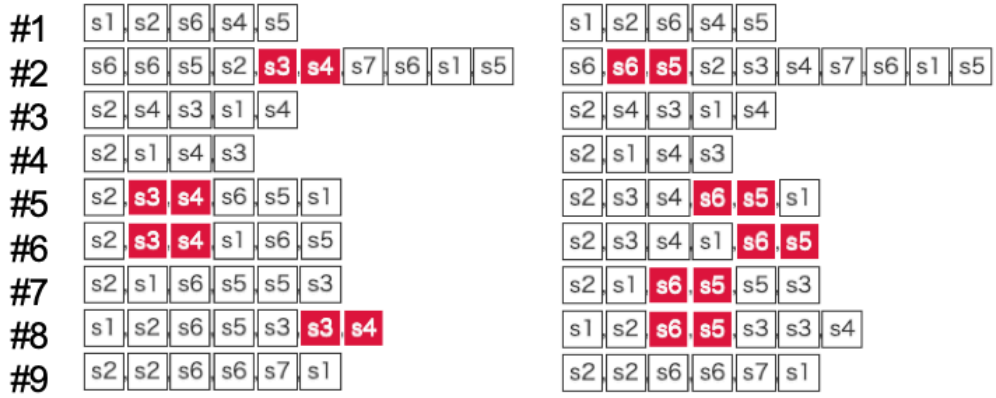



**図33** 共起頻度の高い組を強調した系列データ

図26で、共起頻度の高い組を強調したのが**図33**である。#1などは、個別のプレイを示す。同じプレイについて、左側はs3,s4の組を、右側はs6,s5の組を強調している。

提案手法では共起頻度が高い組に着目するが、この例では、それらが系列データ全体において占める位置は一定でない。正確に何回目でこの組で動かすという傾向が見られない。提案手法では、「共起頻度の高い組が現れる」という点で、#2、#5、#6、#8を同一クラスターに分類することは可能であろう。個々のプレイ、例えば#2単独での共起マトリックスでそのプレイを表せば、並べ替え操作の回数によらず、$7 \times 7 = 49$次元のベクトルで表せる。#2ならば「s6,s6,s5,s2,s3,s4,s7,s6,s1,s5」といった系列全体について順序に着目する手法は、提案手法とは異なる分類結果になると予想される。これは優劣ではなく、単に抽出する特徴の違いである。

また、#2の「s6,s6,s5,s2,s3,s4,s7,s6,s1,s5」といった系列全体について、クラスタリングするような手法では、まず、#2と#3とでは系列の長さが違うといった課題を解決しなくてはならない。提案手法では、そのような問題はない。

### 4.6.2 試行錯誤

```
s1    // 自分の車に30Lの燃料を入れたい。
s2    // 自分の車(myCarインスタンス)を生成して、燃料(fuel)をいれてください。
s3    // ただし、myCarの燃料(fuel)の初期値は0とする。
s4    class Car {
s5      var fuel: Int = 0
s6      func refuel(addFuel: Int) {
s7        self.fuel += addFuel
s8      }
s9    }
s10   var myCar = Car()
s11   myCar.refuel(addFuel: 30)
```

**図34**　ジグソー・コードのパズル

**図34**はジグソー・コードのパズルである。そのパズルで正解した12人の共起マトリックスが**図35**[31]、このときの系列データで共起頻度が平均 + 標準偏差 x 2より大きい組を強調したのが**図36**である。これによると、s10とs11が前後して動かされることが多い。

解釈としては、この2行は問題文の「自分の車(myCarインスタンス)を生成して、燃料(fuel)をいれてください。」に対応しており、この2行の組の共起頻度が高いのは納得できる。

| n \ n+1 | s4 | s5 | s6 | s7 | s8 | s9 | s10 | s11 |
|---|---|---|---|---|---|---|---|---|
| s4 | 0 | 2 | 2 | 1 | 1 | 2 | 2 | 3 |
| s5 | 1 | 1 | 2 | 2 | 0 | 1 | 3 | 3 |
| s6 | 0 | 1 | 0 | 3 | 3 | 2 | 1 | 0 |
| s7 | 1 | 1 | 1 | 0 | 4 | 0 | 0 | 0 |
| s8 | 1 | 0 | 0 | 2 | 0 | 3 | 1 | 0 |
| s9 | 0 | 2 | 1 | 2 | 0 | 1 | 3 | 2 |
| s10 | 2 | 2 | 2 | 0 | 0 | 2 | 4 | 6 |
| s11 | 2 | 3 | 3 | 0 | 0 | 0 | 5 | 0 |

**図35**　ジグソー・コードの共起マトリックス

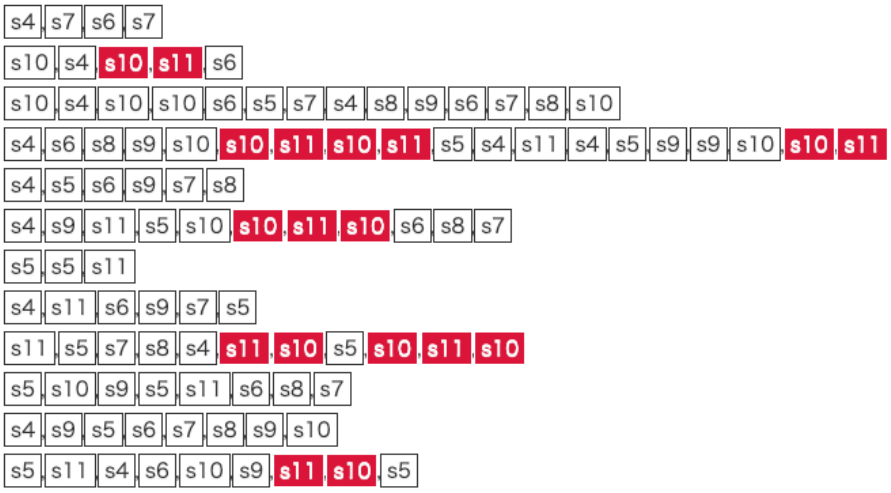



**図36**　共起頻度の高い組を強調した系列データ

図36によると、s10、s11の組は、1つのプレイの中でセットで複数回で動かされていることが分かる。例えば、プレイヤーがこの組を含んで試行錯誤していれば、このような系列データになるだろう。かつ、闇雲に試行錯誤してるわけではなく、「s10、s11が関連している」という知ってる・分かっていることは押さえつつ試行錯誤していると考えられる。

文献[30]では、例えば「s6,s6,s5,s2,s3,s4,s7,s6,s1,s5」という系列データが得られた場合、各ピースについて最後の操作のみを拾って「-,-,-,s2,s3,s4,s7,s6,s1,s5」などとして扱う。「-」は無視した事象である。このようにすると図22ではなく図23のようなUIであれば、各プレイの操作数は同じになり4.6.1で指摘したような、系列の長さが異なるという課題は解決済である。その代わり、試行錯誤の状況は無視してしまうのである。

これも優劣ではなく、抽出する特徴が異なる、あるいは研究や実践の前提の違いによるものであろう。正解が必ず1つ、そして1つだけ存在し、正解に向かって最短距離で解くことを求められるような問題に、この手法は向いている。困難な問題や答えのない問題に取り組んで試行錯誤すること前提とした問題を解くプロセスならば、提案手法が向いているだろう。

### 4.6.3　部分解、または答えのない問題

不正解だったプレイでも、正解だったプレイで共起頻度の高かった組でピースが移動されることがある。**図37**は、不正解だったプレイについて、正解だったプレイで共起頻度の高かった組を図36と同様に強調表示したものである。s10,s11の組が強調されているが、不正解だったプレイの中でこの組の頻度が高かったわけではない。上から4番目のプレイでは、この組で3回、ピースを移動している。

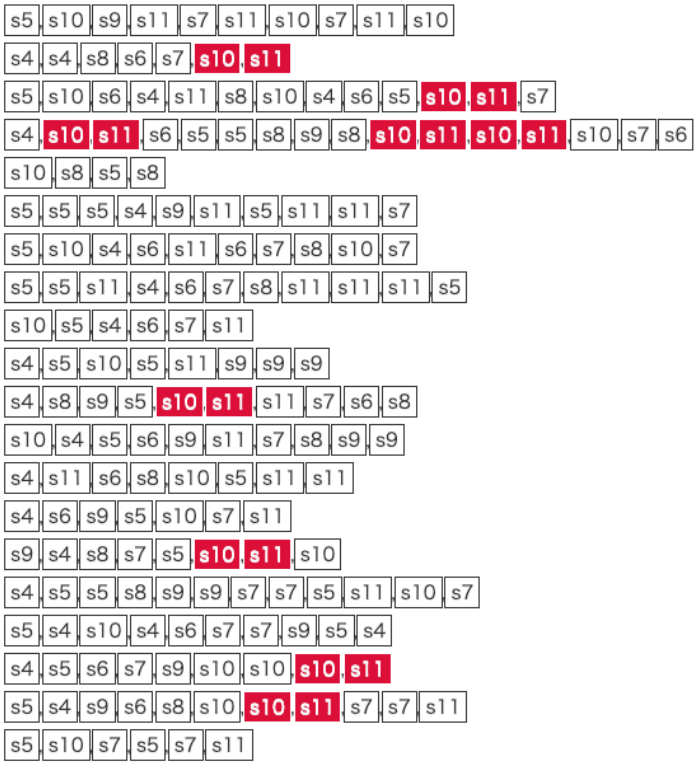



**図37**　不正解のプレイについて、共起頻度の高い組を強調した系列データ

しかし、「for {」と「}」とを組で動かす場合、完成順序の部分列の一致と対応付けるのは困難である。

**解釈と考察**

図37は、最終的な完成順序は不正解であっても、思考過程では、正解プレイと同じ思考を含んでいる場合があると言える。全体の並びは分からなくても、部分的な並びは分かっている。これは、並べ替え過程を評価することで、並べ替え結果を判定するのとは異なる評価を下せる可能性を示唆している。また、このパズル問題に答えがなかったとしても、なお、プレイを評価できる可能性や、評価方法を示唆している。

### 4.6.4　長い編集操作

編集操作、パズル操作の数が多い場合、共起する編集操作の組同士の共起を分析する必要があるかもしれない。分析手法としては、文献[29]で提案された手法などが該当する。

アプリケーションとしては、ジグソー・テキストのピース数が多い場合が、長い編集操作に該当するだろう。例えば、宮沢賢治の「雨ニモマケズ」**[33]**の、空行を含む38行を38個のピースとしたパズルである。あるいは、論文や小説の段落をピースとしたパズルも該当する。われわれは、このような大きなパズルを用意しているが、解くのに大変時間がかかり、かつ正解に達しない。いまのところ、実験や授業での実践などは行っていない。長い文章については予備実験の分析例があり、11.3節で述べる。いずれにしても今後の課題である。

## 4.7 結論

作文やプログラミングといった人間の知的プロセスを、系列データにおける編集事象の共起から分析する手法を提案した。ユーザーの編集操作を記録・測定するアプリを具体的に示し、それらアプリが測定した具体的な系列データをあげて、提案手法がそこから有益な情報を抽出できることも示した。また、提案手法と他の手法との違いを考察して、提案手法が、試行錯誤や答えのない課題に取り組むプロセスの分析に有効であると示した。

本章で取り上げた系列データは、たかだか数10個の編集事象から構成されるものである。もっと長い系列データについては、共起する編集操作の組同士の共起を分析する必要があるかもしれない。

# 5 取捨選択の検出

ここでは時間的な共起分析の応用例を論じる。時間的な共起分析を、不要なピースを含んだ問題(パズル問題)を解くジグソー・コードに適用すると、作問者が意図して紛れ込ませた間違いピースが、正しいピースとの間で取捨選択されたかどうかを検出できる。このような取捨選択は、試行錯誤の一部であり、本研究の目的にかなう。

プログラミングのプロセスの学習分析では、学生が「何かに迷っている」ことを検出できる研究が増えてきたが、具体的に「何に迷っているか」を機械的に検出するのが困難である。われわれは、取捨選択・並べ替え型のプログラミング・パズルで、取捨選択操作の時間的な共起分析によって、学生の迷いを検出する手法を提案する。提案手法は、学生がどちらを選ぶか迷っている可能性が高い、ピースのペアを具体的に提示する。問題作成者および教師による判断を正解として、提案手法で検出した迷いを評価したところ、提案手法の適合率が44%、再現率が92%となり、提案手法による提示が問題作成者および教師による判断をほぼ含む結果となった。このとき、ピースの全ての組み合わせに対して、実際に迷われた組み合わせの割合は2%であった。取捨選択操作の時間的な共起は、問題作成者および教師に対して、学習者の具体的な迷いを提示する手法として有効であることを示した**[34]**。

## 5.1 はじめに

2020年からのコロナ禍は学習・教育のデジタル化を加速した。プログラミング学習も同様で、学習環境が充実するとともに、プログラミングのプロセスに関係する様々な学習データを取れるようになってきた。

そこで、これからの学習分析(Learning Analytics)研究は、学習者が「何を分からないのか」の検出にとどまらず、その状況をどのように解決しようとしているのか、試行錯誤を肯定的に捉えて具体的な内容を検出することが求められるだろう。

しかし、5.2節で述べるように、プログラミングのプロセスを測定・分析する研究では、学生が「何かが分からない」でいることを検出できる研究は増えてきたが、その状況を具体的に「どう解決しようとしてるか」を機械的に検出するのは困難である。

そこで、本研究では試行錯誤の一種として取捨選択に注目する。プログラミング・パズルの「ジグソー・コード」を題材にして、学生の取捨選択操作を具体的に検出して、学生の取り組みや作問を評価するのに役立つようなデータ分析手法の開発に取り組む。

本章ではまず、5.2節でプログラミング・パズルや分析手法などの先行研究を検討する。その中で、本研究の題材であるジグソー・コードも紹介する。5.3節で本研究の目的を定義する。5.4節では提案する分析手法、提案手法を検証するためのデータ収集方法、および検証方法を述べる。5.5節で結果と検討、5.6節で考察し、5.7節でまとめる。

## 5.2 関連する研究

従来のプログラミングのプロセスを測定・分析する研究では、学生が「何かが分からない」と問題を抱えていることを検出できる研究は増えてきたが、その状況を具体的に「どう解決しようとしてるか」を機械的に検出するのは困難である。

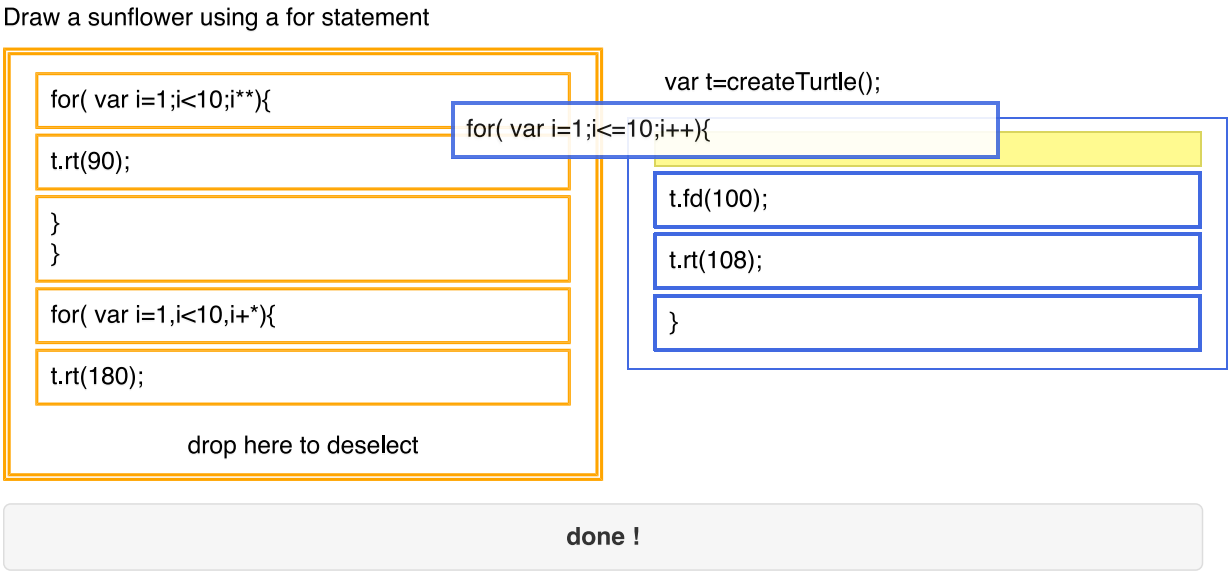



**図38**　取捨選択・並べ替えパズルのジグソー・コードと、その問題「2-2 ひまわりを描くプログラム」

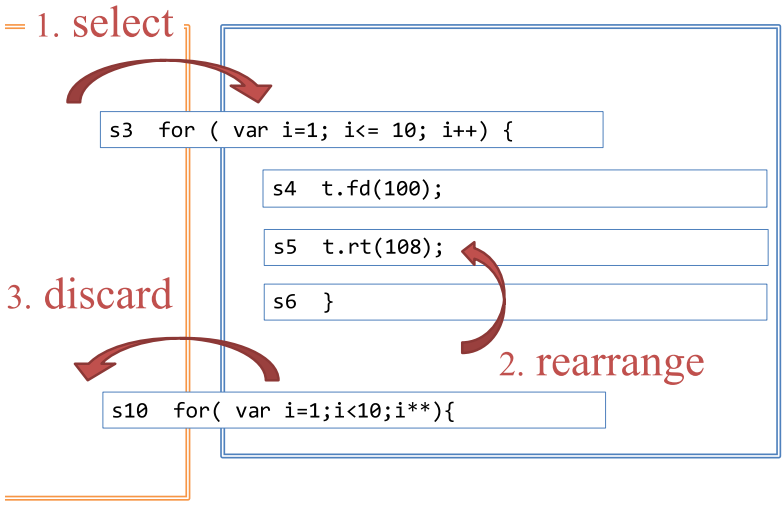



**図39**　ジグソー・コードのピース操作：取る、捨てる、並べ替える。ここでは、取る操作と捨てる操作に焦点を当てる。

### 5.2.1　編集距離に基づいて問題解決プロセスを評価

森永らは、レーベンシュタイン距離を修正した距離を考案し、その距離を使って問題を解くプロセスを分析すること提案している。その中で「…(解答者が)仮説的な考えを試しながら『建設的な試行錯誤』でアルゴリズムを設計せずパターンで回答しようとしていた過程が時系列変化に現れる」可能性を指摘している**[35][36]**。

Maharjan らはプログラム行を取捨選択・並べ替えて完成させる Parsons Problem について、「正解との編集距離は、解答の正しさの度合いを示す」として、レーベンシュタイン距離を修正

した編集距離に基づく分析手法を提案している。そして「正解との編集距離の列が非単調で頻繁に上下に振れるのは、試行錯誤を示唆している」と述べている**[37]**。

浦上らは、学習者が最終的に完成させたプログラムを正解とし、途中の実行時点でのプログラムを正解と比較して同一であった行を「正解行数」としている。そして、ある実行時点で、正解行数がそれまでの最大値を上回らないとき、その実行においては完成に近づいていないので、つまづいていると判定している**[38]**。これは正解とのハミング距離に基づいてつまづきを判定してると言える。

編集距離による分析は、課題の内容に立ち入らずに幅広く応用できるメリットがある。しかし、解答プロセスにおける編集距離の増加は、必ずしもつまづきや試行錯誤の現れではない**[39]**。例えば、コンピューターによる数の整列の途中では、正解(整列済の数列)との距離が広がることがあるが、整列アルゴリズムがつまづいたり試行錯誤してるわけではない**[40]**。

### 5.2.2 構文要素の粒度での測定や機械学習

山口ら**[41]**は、自由なプログラミングのプロセスの測定・分析では、コード行よりも構文要素の単位で測定する方が有益なことを示唆している。それと同時に、測定・分析の対象の粒度は構文要素でよいのか、もっと大きな単位が必要ではないかと問題提起している。クイックソートのピボット、塩分濃度の計算、タートルを動かす方向と距離などといった問題を解決するプロセスは、トークンよりも大きなかたまりで変化を記述する必要があるのではないか。

Kawaguchi et al.**[42][43][44]**は、C言語のプログラミングについて、30秒ごとのソースコードのスナップショットからトークン粒度の差分ベクトルを生成して個々の編集操作を表現することを提案している。そして、これらソースコード編集差分ベクトルを入力とし、「数値の型の理解不足」といった学習状況を出力とした機械学習モデルを開発・評価している。このシステムは「○○ができない」という状態を推定するものだが、本研究の背景は、その「できない」状況を解決するための行動の検出を求めている。この機械学習モデルでそのような推定が可能なのか、研究が待たれる。

### 5.2.3 プログラミング・パズルとジグソー・コード

プログラミング教育では、0からコードを書かせるのではなく、シャッフルされたプログラムの断片群から取捨選択・並べ替えて完成させるタイプの教材が使われることがある。5.2.1で取り上げたParsons Problem**[45]**[37]や短冊型プログラミング問題**[46]**が、そのようなタイプの教材である。本稿では、このような教材をプログラミング・パズルと呼ぶことにする。

ジグソー・コードは、そのようなプログラミング・パズルの1つである。プログラムをいくつかのピース(断片)に分割し、さらに不要なピースも混ぜてシャッフルしたものから、取捨選択・並べ替えて正しいプログラムを完成させるジグソー・パズルである。われわれが開発・運用して授業や実験で使っている**[47]**。

**図38**は、取捨選択型のジグソー・コード2の画面である。左側のオレンジで囲まれた選択肢枠内の最初の並び順は、解答を始めるたびにシャッフルされている。右側の枠の上下には、あらかじめ固定ピースが配置されていることもある。解答者は、左側の枠から、適切なピースを選んで、右側の青で囲まれた解答枠にドラッグ&ドロップで移動する。後で不要と分かったピースは、右から左にドラッグ&ドロップで戻す。左右どちらの枠内でも、ピースをドラッグ&ドロップで並べ替えることができる。これらの操作を繰り返して、ピースを右側の解答枠内に適切な順序で並べてプログラムを完成させる**[48][49]**。解答を完成させるとdocument idと呼んでいるIDが表示される。Document idは問題提示から解答完成までのプロセスごとにユニークに発行される。

ジグソー・コードは、ユーザーの並べ替え操作を測定している。左の枠から右の枠へピースを移動して選ぶ、右の枠から左の枠へピースを移動して捨てる、それぞれの枠内でピースを移動して並べ替える、これらすべての操作について、操作対象のピースIDや操作時刻などをdocument idと共に記録している。

生徒が左の枠から右の枠にピースを移動させたときは「ピースを選択した」と言い、右の枠から左の枠にピースを移動させたときは「ピースを捨てた」と言う(**図39**).

ここで、いくつかの用語を定義する(図39):

取る(select)
: 生徒が左の枠から右の枠にピースを移動させるとき、「ピースを選択する」と言う。

捨てる(discard)
: 右側から左側にピースを移動させるとき、「ピースを捨てる」と言う。

解答(solution)
: 解答は右手の枠にあるピースとその順番である。解答は解くプロセスで変化する。

最終的な解答(final solution)
: 最終的な解答は、解答プロセスの最後の解答であり、答えとも呼ばれる。必ずしも正解である必要はなく、パズルそのものが正解を持つ必要もない。

選択する(choose)
: あるピースが最終的な解答にあるとき、学生はそれを選択したことになる。解くプロセスで何度かそのピースを捨てたかもしれないが、最終的な解答にそのピースを入れることに決めたのである。すなわち、本稿では「取る(select)」と「選択する(choose)」を区別する。

選択肢(choice)
: 選択肢とは、そこから生徒が選ぶ複数のピースたちのことであり、教師が生徒に選ぶことを検討させることを目的としたピースたちのことである。本研究では、生徒の取捨選択の操作から生徒にとっての実際の選択肢を検出する方法を提案する。

特に本研究を英語で記述するとき、前記の定義は重要である。筆者が日本語で「取捨選択」と言うとき、「取る(採用する)」操作や「捨てる(棄却する)」操作を経て最終的な「選択」があることを意味する。しかし「取捨選択」という日本語を単純に英語にすると「select」などと翻訳され、途中の操作が意味から外れて伝わる可能性がある。

### 5.2.4 ジグソー・コードの操作の時間的な共起分析

ジグソー・コードが測定した操作ログの分析では、操作過程のパターンを見つけるために、われわれは操作対象となったピースの時間的な共起行列を使ってきた。この共起行列は、あるピースを動かした次にどのピースを動かしたかの回数を集計する**[50]**[16]。続けて動かされる回数が多いとき、それら2つのピース間には何か特別な関係があると考えるのが時間的な共起分析の動機である。この発想は、テキスト分析においてテキストの空間的な位置の共起を分析する、従来の共起分析と同様である。**図40**は、図38のパズル(問題)を223人の学生が解いたときの、操作ログを集計した時間的な共起行列である。行や列の見出しにあるs3やs4は「2-2 ひまわりを描くプログラム」に含まれるピースの内部的なIDである。図では、s3行の次にs4列を動かした回数が全部で91回あったことが分かる。

| n \ n+1 | s3 | s4 | s5 | s6 | s7 | s8 | s9 | s10 | s11 |
|---|---|---|---|---|---|---|---|---|---|
| s3 | 23 | 73 | 63 | 15 | 1 | 19 | 6 | 5 | 10 |
| s4 | 24 | 29 | 95 | 76 | 1 | 23 | 6 | 8 | 6 |
| s5 | 16 | 94 | 35 | 85 | 4 | 22 | 4 | 8 | 8 |
| s6 | 7 | 12 | 22 | 8 | 5 | 12 | 7 | 7 | 7 |
| s7 | 0 | 0 | 1 | 3 | 1 | 3 | 1 | 3 | 0 |
| s8 | 7 | 19 | 36 | 17 | 3 | 14 | 0 | 3 | 5 |
| s9 | 2 | 5 | 10 | 2 | 1 | 7 | 3 | 0 | 1 |
| s10 | 18 | 6 | 3 | 7 | 1 | 5 | 6 | 16 | 7 |
| s11 | 14 | 19 | 10 | 2 | 0 | 5 | 2 | 3 | 10 |

**図40**　並べ替え操作の時間的な共起行列

セルの値が**平均**＋**標準偏差**より大きいセルを黄色(薄い背景色)、**平均**＋**標準偏差**×2より大きいセルを赤(濃い背景色に白い文字)で強調している。図40では、s3の次にs4を動かした回数の「91」が比較的多いことが分かる。**図43**は図38の正解と、各ピースのIDを示している。IDがs1のオブジェクトもピースである。ここに配置するピースの内容は作問者の自由に委ねられるが、指示の役割のピースを配置することが多い。図では「ひまわり(sunflower)を描け」と指示している。

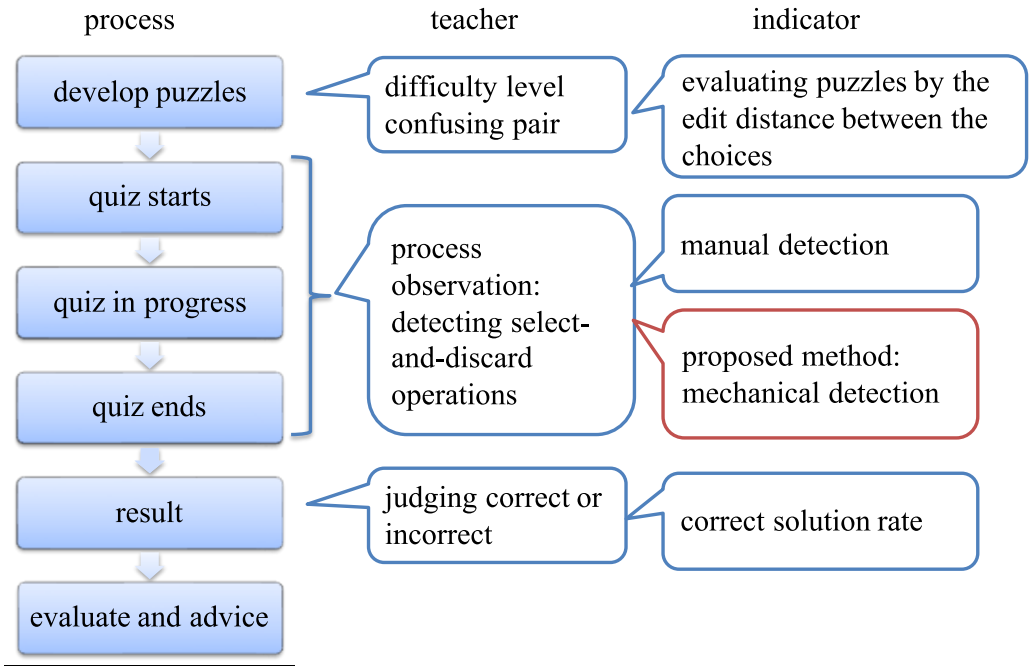



**図41**　取捨選択操作を検出する目的

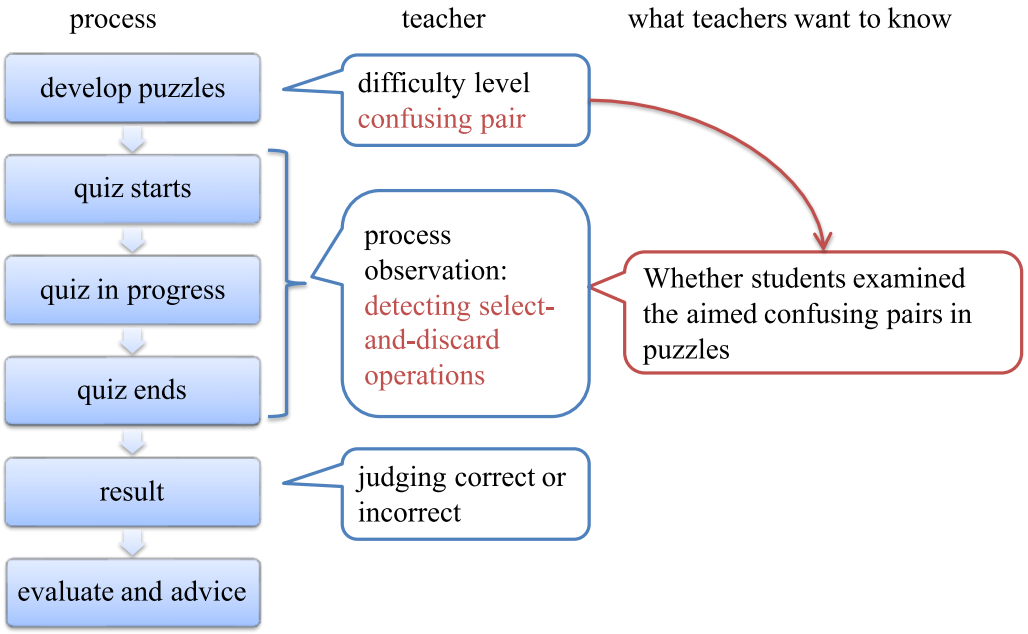



**図42**　パズル問題作りから評価、アドバイスに至る全過程におけるこの研究の位置づけ。

厳密に言えば、本稿の範囲では、提案手法は時間的というより手続き的である。しかし、この種の分析を用いた他の研究に従い、本研究では、これを時間的と呼ぶ。

s1 Draw a sunflower using a for statement

| | |
|---|---|
| s7 | }<br>} |
| s8 | t.rt(90); |
| s9 | t.rt(180); |
| s10 | for( var i=1;i<10;i**){ |
| s11 | for( var i=1,i<10,i+*){ |

| | |
|---|---|
| s2 | var t=createTurtle(); |
| s3 | for( var i=1;i<=10;i++){ |
| s4 | t.fd(100); |
| s5 | t.rt(108); |
| s6 | } |

**図 43**　パズル「2-2 ひまわりを描くプログラム」のピース ID と正解。

### 5.2.5　他の分析手法

ジグソー・コードの測定データからは、その他に、最初に／最後に動かしたピース、各ピースが動かされた回数、あるピースを動かした後に別のピースを動かすまでの所要時間などを分析できる。

中村ら[48][49]はジグソー・コードの操作ログを、「最後に動かされることが多い」などの観点で分析した。それらが最後の動かされることが多いのは、それらについて最後まで考えていたからなどと推定した。

## 5.3　目的

本研究の目的は、試行錯誤の内容を具体的に検出して、学生の取り組みや作問を評価するのに役立つ、データ分析の手法を提案することである。

本研究では、取捨選択・並べ替え型のプログラミング・パズルを題材とする。

本稿における「取捨選択」とは、パズルを解くプロセスにおいて、特定の複数のピースを「どれかが正しい可能性のある」選択肢(2択、3択、…)と捉えて、そこからピースを取り、あるいは取ったピースを捨てて、最終的にピースの選択を確定させる一連の操作を指す。測定・分析による検出の内容は、解答者がどのピースを選択肢と捉え、どれを・どのように選択したかである。

取捨選択・並べ替え型のプログラミング・パズルでは、パズル問題の作成にあたって、学習者が学習のポイントを理解したか確認したり、難易度を調整するために、教師や問題作成者はよく似た間違えやすい複数の選択肢を含ませることがある。本研究の提案方式の使い途は、学生が実際にどのピースを選択肢としたかを、いち早く教師などに提示するものである(**図 41**)。

パズル問題の作成から、パズルによる小テストの実施／解答者の操作、小テストの結果、講評・アドバイスにいたる全行程では、各段階でさまざまな指標を得ることができる。これらの指標は、互いに関係があるとしても別々の指標である。本研究で得られる指標は、パズルを解くプロセスにおける取捨選択に関する指標である(**図 42**)。

## 5.4　研究方法

われわれは、学生がどちらを選ぶか取捨選択している可能性が高いピースのペアを、ある特定の操作の共起分析によって提示する手法を提案する。検証は、提案方式が「実際に起きた」とし

て検出した取捨選択操作が、作問者にとって納得できるものか(正確性)を評価する。これが納得できるものであれば、提案手法による検出を、作問で作り込んだ選択肢が妥当なものであったかといった評価に使える(図41)。

### 5.4.1 提案手法: 取捨選択操作の時間的な共起分析

提案手法では、5.2.4の共起分析と異なり、左の枠から右の枠へピースを取る操作と、右の枠から左の枠へピースを捨てる操作のみ、すなわち取捨選択操作のみについて共起を集計する。2つの取捨選択操作の間に、右の枠内での並べ替え操作が入る場合があるが、そのような操作があったとしても2つの操作を共起としてカウントする。また、図40と異なり、2つの操作の順序を問わずに集計する。

図39で、まず、s7を取り、続いてs6を枠内で上に移動し、s4を捨てたとする。5.2.4の共起分析ではs7とs6、s6とs4が共起するが、s7とs4は共起とみなさない。提案手法では、s7とs4が共起したとみなし、s7とs6およびs6とs4の共起はカウントしない。図39のように操作された場合、共起行列は、従来方式では**図44**の左側のようにs6行s4列とs7行s6列のセルに1を加える。提案方式では図の右側のように、s4行s7列のセルに1を加える。

| | s3 | | s5 | | s10 |
|---|---|---|---|---|---|
| s3 | | | +1 | | |
| | | | | | |
| s5 | | | | | +1 |
| | | | | | |
| s10 | | | | | |

| | s3 | | s5 | | s10 |
|---|---|---|---|---|---|
| s3 | | | | | +1 |
| | | | | | |
| s5 | | | | | |
| s10 | | | | | |
| | | | | | |

**図44** 図39の操作について、全移動操作を対象とする手法(左、4章)による共起行列と本章の提案手法(右)による共起行列。

**図45**は図40と同じ操作ログについて集計した取捨選択の共起行列である。どちらを先に選んだか順序を問わないので、共起行列の右上に集計されている。**図46**は図45のセルの値の頻度分布である。s3-s10の取捨選択操作の回数35回が、他のピースの組み合わせに比べて多いことが分かる。

**表1** 結果： 提案手法で提案されたペア(左)とパズルが狙ったペア(右)：下線ペアは他方に含まれない。下線を引いたペア(左)はfalse positive(偽陽性)、下線を引いたペア(右)はfalse negative(偽陰性)。

| # | Title | Detected by proposed method | Correct selection (aimed) |
|---|---|---|---|
| 1-1 | 砂時計の容器を作るプログラム | s4-s9, s5-s9 | s4-s9, s5-s9 |
| 1-2 | 濃度判定プログラム | s3-s10, s5-s12, s4-s7 | s3-s10, s4-s7, s4-s11, s5-s8, s5-9, s5-s12 |
| 1-3 | 整数入れ替えプログラム | s3-s9, s7-s12, s5-s11, s8-s13 | s3-s9, s4-s10, s4-s14, s5-s11, s7-s12, s8-s13 |
| 2-1 | PCR 検査陽性率 | s4-s13, s8-s14 | s4-s13, s8-s14 |
| 2-2 | ひまわりを描くプログラム | s5-s8, s3-s10, s3-s11 | s3-s10, s3-s11, s5-s8 |
| 3-2 | 二乗を計算するプログラム | s4-s8, s2-s9 | s4-s8, s4-s10, s2-s9 |
| 3-3 | 九九出力プログラム | s2-s5, s4-s8 | s2-s5, s4-s8 |
| 4-1 | ドーナツを描くプログラム | s4-s9, s5-s11 | s4-s9, s5-s11 |
| 4-2 | 閏年判定プログラム | s3-s7 | s3-s7, s4-s9 |
| 5-1 | GO TO travelの料金判定プログラム | s5-s8, s4-s7, s6-s8 | s4-s7, s5-s8 |
| 5-2 | GO TO travelの料金判定プログラム2 | s10-s12, s4-s11, s4-s14 | s4-s11 s4-s14, s10-s12, s9-s15 |

**表2** 結果2： TTP = True Positive, FP = False Positive, FN = False Negative, TN = True Negative, 移動可能なピースの数(No. of pieces), すべてのペアの可能数(No. of candidates), ペアの取る操作と捨てる操作の割合(ratio).

| # | Title | TP | FP | FN | TN | No. of pieces | No. of candidates | ratio |
|---|---|---|---|---|---|---|---|---|
| 1-1 | 砂時計の容器を作るプログラム | 2 | 0 | 0 | 64 | 11 | 66 | 0.03 |
| 1-2 | 濃度判定プログラム | 3 | 0 | 3 | 52 | 10 | 55 | 0.05 |
| 1-3 | 整数入れ替えプログラム | 4 | 0 | 2 | 74 | 12 | 78 | 0.05 |
| 2-1 | PCR 検査陽性率 | 2 | 0 | 0 | 89 | 13 | 91 | 0.02 |
| 2-2 | ひまわりを描くプログラム | 3 | 0 | 0 | 42 | 9 | 45 | 0.07 |
| 3-2 | 二乗を計算するプログラム | 2 | 0 | 1 | 53 | 10 | 55 | 0.04 |
| 3-3 | 九九出力プログラム | 2 | 0 | 0 | 26 | 7 | 28 | 0.07 |
| 4-1 | ドーナツを描くプログラム | 2 | 0 | 0 | 43 | 9 | 45 | 0.04 |
| 4-2 | 閏年判定プログラム | 1 | 0 | 1 | 27 | 7 | 28 | 0.04 |
| 5-1 | GO TO travelの料金判定プログラム | 2 | 1 | 0 | 25 | 7 | 28 | 0.07 |
| 5-2 | GO TO travelの料金判定プログラム2 | 3 | 0 | 1 | 75 | 12 | 78 | 0.04 |

| n \ n+1 | s3 | s4 | s5 | s6 | s7 | s8 | s9 | s10 | s11 | s12 |
|---|---|---|---|---|---|---|---|---|---|---|
| s3 | 0 | 2 | 1 | 2 | 2 | 4 | 0 | 35 | 0 | 2 |
| s4 | - | 3 | 1 | 0 | 16 | 2 | 0 | 1 | 8 | 3 |
| s5 | - | - | 2 | 2 | 2 | 7 | 3 | 3 | 0 | 20 |
| s6 | - | - | - | 0 | 1 | 0 | 1 | 2 | 1 | 1 |
| s7 | - | - | - | - | 3 | 0 | 0 | 2 | 5 | 2 |
| s8 | - | - | - | - | - | 3 | 0 | 0 | 1 | 2 |
| s9 | - | - | - | - | - | - | 0 | 0 | 0 | 0 |
| s10 | - | - | - | - | - | - | - | 3 | 0 | 1 |
| s11 | - | - | - | - | - | - | - | - | 2 | 0 |
| s12 | - | - | - | - | - | - | - | - | - | 1 |

**図45**　取る-捨てる操作の時間的な共起行列

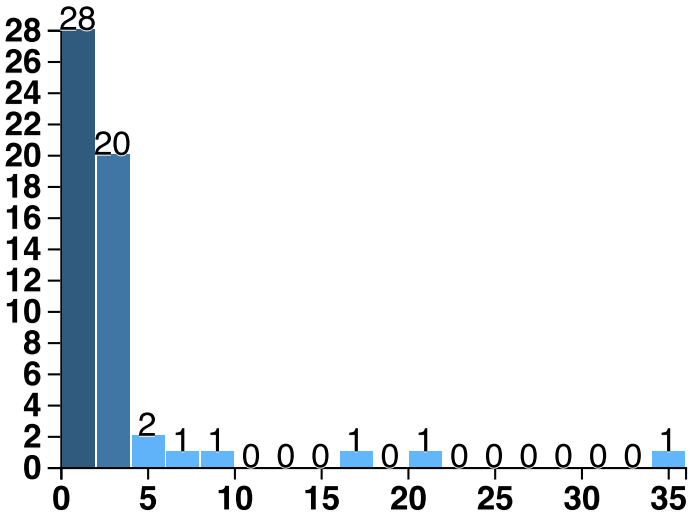



**図46**　取る-捨てる操作の時間的な共起のヒストグラム

対角線上のセルは、ある同じ1つのピースを取った後に捨てた、あるいは捨てた後に取り直した回数である。これは、「ある1つのピースを選ぶか選ばないか考えた」などと解釈されることになるだろう。

### 5.4.2　検証のためのデータ収集

提案方式の検証に使うため、ジグソー・コードの問題を作るにあたっての作問意図と、その問題を解くプロセスの測定データを収集する。

大学の文理融合系学部の1年生を対象とした必修授業「コンピューティング実習」で、授業内の小テストでジグソー・コードを使い、学生の操作を測定する。受講生は約230人。授業は、

JavaScript、HTMLやタートルのライブラリを使って変数、条件分岐、繰り返し、関数など、プログラミングの初歩を学ぶものである。授業はオンラインで行う。

小テストの作問にあたっては、学習のポイントを考えさせるように意図して、不要な選択肢ピースを混ぜる工夫をして学生に取捨選択させる。作問意図は、取捨選択が起きるであろうピースのペアのリストとしてデータ表現する。小テストの内容は、図形を描くためにタートルを動かす順序、塩分濃度の計算、Go Toトラベル割引料金の計算の仕方などを考えさせるものである。1回の授業で2〜3問、合計13問を、新たに開発して出題する。

小テストの実施については、まず初回の小テストの前に、実験の概要、ジグソー・コードの操作方法およびdocument idの提出方法を説明する。小テストは授業の最初に、前回までの内容を理解しているかの演習として実施する。解答後に、問題の解説を提示する。

### 5.4.3 提案手法の検証

検証は、前節の作問意図と、解答プロセスの測定データを使って行う。提案方式が「実際に起きた」として、測定データから検出した取捨選択操作が、作問者にとって納得できるものか(正確性)を評価する。

提案方式によって、解答プロセスの測定データから取捨選択操作の時間的な共起行列を作る。値が**平均** + **標準偏差** × 2よりも大きいセルを選び、セルの行列の見出しを取り出すと、それが取捨選択されたペアとなる。図45では、s3-s10、s4-s7、s5-s12が検出された取捨選択ペアである。

そして、作問意図のペアのリストを正解として、提案方式が検出したペアの適合率(precision)と再現率(recall)を評価する。適合率は、提案手法が検出した操作の中に、作問者の意図に適合した操作がどれだけ含まれているかという正確性を示す。再現率は、作問者の意図に適合する操作のうちで、どれだけの操作を提案手法が検出したかという網羅性を示す。

また、提案手法が検出した取捨選択のペアのうち作問意図に適合するのペア、すなわちTrue Positiveの数が、全組み合わせの数に対して小さいことを確認する。これが大きいと、当てずっぽうで検出しても適合する可能性が大きくなる。

## 5.5 結果

作問意図に対して提案手法の適合率が96%となり、提案手法による提示が作問意図にほぼ含まれる結果となった。再現率は82%であった。また、全組み合わせの数に対するTrue Positiveの割合は5%であった。

**表1**は、提案方式による検出と正解、すなわち問題作成者および教師による判断を比較した表である。「s4-s9」などは取捨選択の組み合わせを示し、「s4にするかs9にするか考えた」ことを表す。

**表2**で、TPはTrue Positiveで、提案方式による検出が正解だった数である。FPはFalse Positive、FNはFalse Negative、TNはTrue Negativeである。ピース数は動かせるピースの数である。取捨選択候補数は、2つのピースの間でどちらを選ぶかの場合の数に、ピース自身を選ぶ

かどうか考える場合の数を加えた数である。すなわち、動かせるピースから2つを選ぶ組み合わせ数に、ピース数を加えた数である:

**取捨選択候補数** $= {}_nC_2 + n, n$ : **動かせるピース数**

TN は取捨選択候補数から TP、FP、FN の合計を引いたものである:

TN = 取捨選択候補数 - TP - FP - FN

表2から適合率、再現率および取捨選択候補数に対する TP の割合を計算すると**表3**となる。

**表3**　適合率(Precisioin)と再現率(recall)

| | |
|---|---|
| precision | 96% |
| recall | 82% |
| Percentage of TPs to the number of discarded candidates | 5% |

表1では、「2-2」などでは False Negative のペアを検出できず、問題番号「5-2」では False Positive のペアを検出している。これらについて具体的に見ていく。また、取捨選択の内容についても具体的に見る。

### 5.5.1　適合率と False positive

提案方式による検出のうち、False Positive(偽陽性)にあたるものでも、具体的に検討してみると外れとも言い切れないものであった。「5-1 GO TO travel の料金判定プログラム」の内容は**図47**である。提案方式の False Positive にあたる「s6-s8」、加えて s5 は次の3つである:

```
s5   var discount = regularPrice * 0.35;
s6   }
s8   return discount;
     }
```

s6 と s8 とはそれほど似ていないが、s5 と組み合わせて s5+s6 と s8 とならば取捨選択は考えられる。このペアは作問者として意図したものではない。この False Positive ケースは、実際に起きた取捨選択操作に基づいて提案方式が作問者に気づきを促す可能性を示唆するものである。

### 5.5.2　再現率と False negative

「4-2 閏年」(**図48**では、提案手法は s4-s9 のペアを検出しなかった(False negative、偽陰性)。これらピースは次の通りである:

```
s4   if (year % 4 == 0) {
       println(year + 'is a leap year.');
     }
```

```
s9   if (year % 4 == 0) {
       println(year + 'is not a leap year.');
     }
```

教師は、これらのピースによって、閏年を決定する際に生徒を混乱させることを狙った。しかし、提案された手法がこれらのピースの中から「取る」と「捨てる」の操作を検出しなかったということは、生徒が閏年の決定方法をよく知っていたことを示唆している(これはとても良いことだ！)。そういうこともありうる。

「2-2 ひまわりを描くプログラム」では、例えばs4-s11のペアを検出しなかった。s7を加えると、これらは次のピースである:

「1-2 濃度判定プログラム」パズル(**図49**)では、提案手法は、例えば「s4-s11」のペアを検出できなかった。これらは、s7を加えて次のようになる:

```
s4   var answer= B/(A+B)*100;
s7   var answer=(A+B)/B*100;
s11  var answer=(A+B)/A/100;
```

表1によると、提案方式は「s4にするかs7にするか」という取捨選択操作を検出しているが、「s4にするかs11にするか」という取捨選択操作を検出しなかった。s11はあからさま過ぎて、取捨選択操作に結びつかなかったのかもしれない。

s5-s8のペアも検出されなかった。これらはそれぞれ次である:

```
s5   answer =Math.floor(answer);
s8   answer= Math.ceil(answer);
```

両者の取捨選択操作を提案方式が検出しなかったことは、「数字の切り捨て」をfloorで行うのかceilで行うのか学生たちが取捨選択しなかった可能性を示唆している(これも良いことだ!!)。

s1 Calculate the travel fee when using the Go To Travel Campaign. The discount rate for Go To Travel is 35%.

```
s7  var discount = gotoPrice * 0.35;
s8  return discount;
    }
s9  println('Travel fee when using Go To Travel is ' + travelCost + ' yen.');
    }
```

```
s2  main();
    function main() {
      var regularPrice = input('Please enter your travel fee');
      var gotoPrice = travelCost(regularPrice);
      println('Travel fee when using Go To Travel is ' + gotoPrice + ' yen.');
    }
s3  function travelCost(regularPrice) {
s4  var discount = regularPrice * 0.35;
s5  return regularPrice - discount;
s6  }
```

**図47**　「5-1 Go To Travel rate #1"、Go To Travelレートを使って旅行料金を計算するプログラムを完成させる。s1も問題のピースである。正しい解答は右側のボックスにある。"Go To Travel "とは、COVID-19回復のために国内旅行を割引する日本政府のキャンペーンである。

s1 Complete a program that uses a function that takes a year as input and determines if the year is a leap year. The function prints "N is a leap year" if the input year N is a leap year and "N is not a leap year" otherwise.

```
s7  function leapYear() {
s8  else {
      println(year + 'is a leap year.');
    }
s9  if (year % 4 == 0) {
      println(year + 'is not a leap year.');
    }
```

```
s2  main();
    function main() {
      var year = input('What year is this in the Western calendar?');
      leapYear(year);
    }
s3  function leapYear(year) {
s4  if (year % 4 == 0) {
      println(year + 'is a leap year.');
    }
s5  else{
      println(year + 'is not a leap year.');
    }
s6  }
```

**図48**　“4-2 閏年判定プログラム”

s1　Aに水の量Bに食塩の量を入力する事で食塩水の濃度はoo%ですと出力されるプログラムを作成してください。

s2　なお濃度を出す際は、数字を切り捨てにして表示されるようにしてください。

```
s3  var A=input('水の量は何リットル?');
    var B=input('食塩の量は何グラム?');
s4  var answer= B/(A+B)*100;
s5  answer =Math.floor(answer);
s6  println('食塩水の濃度は'+answer+'%です');
```

```
s7  var answer=(A+B)/B*100;
s8  answer= Math.ceil(answer);
s9  answer= Math.ceil(answer):
s10 var A,B=input('水の量は何リットル?');input('食塩の量は何グラム?');
s11 var answer=(A+B)/A/100;
s12 answer =Math.floor(answer):
```

**図49**　1-2 濃度判定プログラム

### 5.5.3 取捨選択操作の内容

今回検出された取捨選択は、必ずしもJavaScriptの構文など、プログラミング言語の教科書に出てくるような、「if () { ...} else {}」といったレベルのものではない。

5.5.2のs4とs7は濃度計算の分母と分子が逆である。これは演算子の理解の問題とも取れるが、濃度計算の仕方の問題ともとれる。

表1において、s5とs8は、パズル「2-2 ひまわりを描くプログラム」の取る&捨てるペアである。両者の違いは回転の度合いである。一方、s3とs10の違いは、"以下 "や "増加 "といった演算子である。

s1 Complete the program to draw an hourglass. (The turtle should start at the top left corner and return to the top left corner. Also, you may use up to 6 pieces.)

| | |
|---|---|
| s8 | //2nd<br>t.rt(90);<br>t.fd(50); |
| s9 | //1st<br>t.rt(120);<br>t.fd(50); |
| s10 | //2nd<br>t.rt(120);<br>t.fd(50); |
| s11 | t.rt(60);<br>t.fd(50); |
| s12 | t.fd(100); |

| | |
|---|---|
| s2 | var t = createTurtle(); |
| s3 | //1st<br>t.rt(90);<br>t.fd(50); |
| s4 | //1st<br>t.rt(120);<br>t.fd(100); |
| s5 | //1st<br>t.lt(120);<br>t.fd(50); |
| s6 | //2nd<br>t.lt(120);<br>t.fd(50); |
| s7 | t.fd(50); |

**図50** 「1-1 砂時計の容器を作るプログラム」、正解は右の枠にある

| n \ n+1 | s2 | s3 | s4 | s5 | s6 | s7 | s8 | s9 | s10 | s11 | s12 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| s2 | 7 | 8 | 2 | 2 | 2 | 1 | 0 | 4 | 0 | 1 | 1 |
| s3 | - | 5 | 12 | 5 | 1 | 4 | 3 | 5 | 0 | 1 | 2 |
| s4 | - | - | 7 | 9 | 11 | 13 | 4 | 30 | 5 | 1 | 4 |
| s5 | - | - | - | 7 | 14 | 3 | 4 | 19 | 8 | 4 | 3 |
| s6 | - | - | - | - | 9 | 9 | 3 | 11 | 8 | 3 | 3 |
| s7 | - | - | - | - | - | 5 | 3 | 9 | 1 | 3 | 2 |
| s8 | - | - | - | - | - | - | 1 | 2 | 1 | 0 | 0 |
| s9 | - | - | - | - | - | - | - | 7 | 4 | 4 | 0 |
| s10 | - | - | - | - | - | - | - | - | 3 | 2 | 0 |
| s11 | - | - | - | - | - | - | - | - | - | 0 | 0 |
| s12 | - | - | - | - | - | - | - | - | - | - | 2 |

**図51**　「1-1 砂時計の容器を作るプログラム」パズルの、取る-捨てる操作の時間的な共起行列

「1-1 砂時計の容器を作るプログラム」(**図50**)はs4-s9、s5-s9の共起頻度が高く(**図51**)、これら取捨選択が行われたと推定できる。s4、s5、s9はそれぞれ次のピースである:

```
s4   t.rt(120);
     t.fd(100);

s5   t.lt(120);
     t.fd(50);

s9   t.rt(120);
     t.fd(50);
```

これらの間に構文上の違いはなく、違いはタートルの進む向きと距離である。プログラム言語の構文ではなく、砂時計の形を描く手順的な思考の過程で起きた取捨選択と考えられる。

## 5.6　考察

### 5.6.1　時間的な共起分析と取捨選択操作の適合率

検証の結果から、取捨選択操作の共起行列は意味のある現象を捉えてると言える。図51のように、提案方式による共起行列のセルの値には実際に偏りが見られた。セルに対応する個々の連続操作の中には偶然に連続したものがあるかもしれない。しかし、約230人の操作を集計したうえで頻度に偏りがあるのであれば、そこには理由があると考えられる。このように考えるのは、テキスト分析における共起分析と同様である。

そして、共起行列の偏りが作問意図と適合することが判った。このとき、作問意図の通りに取捨選択が起きた可能性が高いと、教師が考えて授業を進めるのは妥当であろう。

### 5.6.2 再現性

作問意図のペアに対して、学生が一手で正解ピースを選べば取捨選択の操作は起きない。再現性や網羅性という用語を使ったものの、検出されない場合には、取捨選択が起きたのに検出しない場合と、取捨選択が起きなかった場合とがあることに留意すべきである。

### 5.6.3 パズル問題の構造

問題の構造が、2択、3択、…で考えさせるように作られていると、提案手法がうまく機能すると考えられる。

5.5.2で見たように、「2-2 ひまわりを描くプログラム」(図43)を解くとき、s4-s7の関係は容易に2択に還元できるだろう。実際、提案方式はs4-s7を検出している。

その一方で、「3-2 二乗を計算するプログラム」(**図52**)を解くとき、s4やs10に関して次のようなことを考えるだろう「べき乗演算子は**なのでxの二乗はx**2だ、x*2ではない。x*xでもxの二乗になる。べき乗や乗算には、インクリメントx++のように代入を伴わずに値を変える演算子がない。二乗の計算をどこですればよいだろう？…」。この考え方は、容易には2択に還元されないだろう。実際、表1では、提案方式はs4-s10を検出せずFalse Negativeになっている。s4-s8やs2-s9は互いに似ていて、これらの取捨選択は提案手法で検出されている。

s1 Using repetition, complete a program that calculates from one squared to ten squared and displays "N squared is M."

| | |
|---|---|
| s7 | x + 1; |
| s8 | println(x+' squared is '+ x*2) |
| s9 | var x=0; |
| s10 | x*x; |
| s11 | }<br>} |

| | |
|---|---|
| s2 | var x = 1; |
| s3 | while (x <= 10) { |
| s4 | println(x + ' squared is ' + x * x); |
| s5 | x++; |
| s6 | } |

**図52** 「3-2 二乗を計算するプログラム」、正解は右の枠

### 5.6.4 リアルタイム分析

本章(そして本稿全体でも)では事後的に取捨選択を検出しているが、解答者たちが小テストを解いている最中に、一定の時間が経過したところで「続々と取捨選択しつつある」ことを検出できるだろう。最後まで待たなくても、問題作成者の狙い通りに解答者たちが取捨選択したのか、しなかったのかが判れば、小テスト後のコメントを変える時間の余裕が生まれる。このような分析を実現するための、ストリーミング分析といったシステム的な仕組みは今後の課題としたい。

### 5.6.5 True Positiveの割合とパズル問題の設計の評価

取捨選択候補数に対するTrue Positiveの割合は5%であった。5%であれば当てずっぽうで当たるとは言えない。

取捨選択候補は意図してパズルに仕込んだものである。そして、意図したとおりに、たった5%の候補について適合率96%で学生が試行錯誤している。このことから、問題作成者と解答者との間に、一定の共通理解があったと言えるのではないか。本稿では分析方式を主題として評価しているが、別の観点からは問題作成者の狙いがよく当たった、良い問題だったと言えるのではないか。本稿の提案方式を道具として使って、作問そのものを主題として評価する研究は今後の課題としたい。

### 5.6.6 取捨選択と正解／不正解

提案方式そのものは学生の取捨選択について中立である: 検出する取捨選択を良いとも悪いとも評価しない。また本章では、取捨選択を正解／不正解の観点から分析しなかった。**図53**は、図38の正解者についての提案方式による共起行列、**図54**は不正解者についてのものである。これらを見ると、正解者も不正解者も、どちらも同じ組み合わせについて取捨選択したことが分かる。それと同時に、解答者の人数が約230人であったのに共起頻度がせいぜい20であることを考えると、これらの組み合わせで取捨選択しなかった解答者も、正解者／不正解者のそれぞれにいたことが分かる。前節で指摘した「一定の共通理解」を踏まえると、このことは、正解／不正解とは別の評価指標が、取捨選択に関連して成立する可能性を示唆しているだろう。

| n \ n+1 | s3 | s4 | s5 | s6 | s7 | s8 | s9 | s10 | s11 | s12 |
|---|---|---|---|---|---|---|---|---|---|---|
| s3 | 0 | 1 | 0 | 1 | 1 | 3 | 0 | 20 | 0 | 2 |
| s4 | - | 2 | 1 | 0 | 9 | 1 | 0 | 0 | 6 | 3 |
| s5 | - | - | 0 | 2 | 2 | 5 | 2 | 2 | 0 | 17 |
| s6 | - | - | - | 0 | 1 | 0 | 1 | 1 | 1 | 1 |
| s7 | - | - | - | - | 1 | 0 | 0 | 0 | 3 | 0 |
| s8 | - | - | - | - | - | 1 | 0 | 0 | 1 | 1 |
| s9 | - | - | - | - | - | - | 0 | 0 | 0 | 0 |
| s10 | - | - | - | - | - | - | - | 2 | 0 | 1 |
| s11 | - | - | - | - | - | - | - | - | 2 | 0 |
| s12 | - | - | - | - | - | - | - | - | - | 1 |

**図53** 正解者の取る-捨てる操作の時間的な共起行列

| n \ n+1 | s3 | s4 | s5 | s6 | s7 | s8 | s9 | s10 | s11 | s12 |
|---|---|---|---|---|---|---|---|---|---|---|
| s3 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 15 | 0 | 0 |
| s4 | - | 1 | 0 | 0 | 7 | 1 | 0 | 1 | 2 | 0 |
| s5 | - | - | 2 | 0 | 0 | 2 | 1 | 1 | 0 | 3 |
| s6 | - | - | - | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| s7 | - | - | - | - | 2 | 0 | 0 | 2 | 2 | 2 |
| s8 | - | - | - | - | - | 2 | 0 | 0 | 0 | 1 |
| s9 | - | - | - | - | - | - | 0 | 0 | 0 | 0 |
| s10 | - | - | - | - | - | - | - | 1 | 0 | 0 |
| s11 | - | - | - | - | - | - | - | - | 0 | 0 |
| s12 | - | - | - | - | - | - | - | - | - | 0 |

**図54**　不正解者の取る-捨てる操作の時間的な共起行列

## 5.7　結論

以上から、取捨選択操作の時間的な共起は、問題作成者および教師に対して、学習者の具体的な取捨選択操作を検出する手法として有効である。

取る／捨てる操作に限った時間的な共起分析によって、プログラミング・パズルにおける、解答者の取捨選択操作を具体的に検出できた。提案方式の適合率は96%、再現率は82%、実際に取捨選択した組み合わせは全組み合わせの5%で小さかった。提案方式の、解答者の取捨選択操作を具体的に検出する方式としての有効性を確認できた。

学習者が問題を解くプロセスについて、従来の研究はつまづきの検出にとどまっていた。これに対して、本手法はつまづきに際して学習者が何を選択肢として捉え、何を選択したかを具体的に検出する点に新規性がある。また、学習者の考え方を重視し試行錯誤の内容を肯定的かつ具体的に評価する新しい教育において、本研究は学習者が手順的な思考の過程で行う取捨選択を検出できるものであり、この点でも有効性を確認できた。

# 6 解の状態遷移における閉路

ここでは解の状態遷移を分析する。ジグソー・コードのようなアプリケーションでは、解答の状態をピースの並びで表現することで、解答が作り上げられる様子を状態遷移として表現できる。この状態遷移において閉路を検出し、その閉路の起点が最終的な解答の部分列であるならば、その閉路は意図的な試行錯誤とみなせること示した。アプリケーションがアンドゥ機能を持っていて、それを使ったならば意図的な試行錯誤とみなせるだろう。アンドゥを使った場合、元の状態遷移を逆にたどることになる。本研究は、逆にたどらない遷移についても意図的な試行錯誤とみなせることを示した。

近年、日本の教育システムは試行錯誤を好意的にとらえ、その戦略を評価するようになっている。試行錯誤の評価は、学習分析にとって非常に重要となっている。ジグソーコードは、解答プログラムのコードブロックに、誤りブロックをも加えて、シャッフルしたものを生徒に与え、ブロックを正しい順番に並べ替えるよう求めるプログラミング演習である。各生徒の解答行動はログに記録される。本章では、解答の状態遷移のサイクルが学生の試行錯誤の過程を示していることを示す。230名の大学生がジグソーパズルの11問を解いた。各解答行動における解の状態遷移を分析したところ、状態遷移にサイクルが見られた。頻出するサイクルを含む97%のプレイにおいて、各サイクルの最初の状態は最終解の部分状態であった。解の状態遷移にサイクルがあることは、生徒が他の解を探した後にサイクルの最初の状態に戻ったことを示している**[51]**。

## 6.1 はじめに

日本の教育制度は試行錯誤を好意的にとらえ、試行錯誤の戦略を評価する。正解がないように見える問題でも、納得のいく解を見つけることができるのは人間の強みである。したがって、試行錯誤の評価は学習分析において非常に重要になってきている。

しかし、生徒の試行錯誤のプロセスを機械的に推定する方法は、バックトラックを見つける以外にほとんどなかった。ゼロからコードを書くのではなく、シャッフルされた解答プログラムのコード断片を正しい形に並べ替えるプログラミング演習の解答プロセスを評価する研究者もいる。しかし、それらの研究者は、試行錯誤を除外したり、正解の存在に依存したりしている。

本論文では、このような演習問題を解く際の生徒の試行錯誤のプロセスを、プロセスの状態遷移における閉路(cycle)を検出することによって、正解の存在とは独立に推定することを提案する。

### 6.1.1 教育における試行錯誤

近年、日本の教育システムは試行錯誤を好意的にとらえ始め、その戦略(次の段落で述べる)を評価している。つまり、試行錯誤のプロセスには戦略があるということである。中央教育審議会は、「人間は人工知能と異なり、多様な文脈が複雑に絡み合った環境においても、自ら目標を設定し、解のない問題に対しても納得のいく解を見出すことができる。人間は学習を通じてこのような能力を身につける。」と述べている[6](p.10)。同審議会は、子どもたちが「決められた問題を決められた手順で解くという効率性を超えて」試行錯誤できるようにする必要があると指摘した[6](p.10)。したがって、解のない問題の解決プロセスは、学習分析の対象と考えることができる。

国立教育政策研究所教育課程研究センターは、学習評価を解説した参考資料を公表した[7]。これらの資料では、試行錯誤の例として、数式のパラメータを決定する際に、「無作為に選んだ数値を入力して調整する」ことを「見通しのない試行錯誤」と定義し、さらに「大まかな推定値を得てから調整する」ことを「見通しのある試行錯誤」(p.59)と定義し、見通しのある試行錯誤と見通しのない試行錯誤は、それぞれ「十分満足」「おおむね満足」(p.59)と説明している。すなわち、試行錯誤にはさまざまなタイプがあり、肯定的に評価している。

学習分析において、試行錯誤の評価はますます重要になってきている。正解の最適過程からの編集距離を評価したり、正解率との相関を分析したりすることは、必ずしも似ていないかもしれない、なぜならば正解がない場合もあるからである。

### 6.1.2 試行錯誤を無視する研究

Parsons problem (Parson's problem、Parsons puzzlesとも呼ばれる)は、プログラミングの練習問題で[30][45][37]プログラムのコードブロックをシャッフルし、学生に正しい位置に並べ替えさせるプログラミング演習である。6.3.1のジグソー・コードに似ている(**図55**)。生徒がパズルを解

く行動の表現として、Kumarは解答順序を提案した。解答順序とは、ブロックを正しい位置に置いたときの時間的順序の並びとして定義される。ブロックの時間的順序とは、解く過程でブロックが何度も正しい位置に置かれる(そしてそこから取り除かれる)可能性があるが、生徒が最終的にその位置に置いた時の順序である[30]。

もし生徒がs3 (コードブロックのID), s5, s4, s2, s1, s5, s2, s4の順にブロックを動かし、それらのブロックが正しい位置にあれば、解答順序は[s3, s1, s5, s2, s4]となる。この場合、解答順序は生徒がs5を2回目に動かし、s4を3回目に動かし、s2を4回目に動かしたことを示さない。

Kumarが述べたように、このアプローチでは試行錯誤の情報が失われる。彼は、バックトラックやループ行動などの試行錯誤は非生産的であり、問題解決戦略の一部ではなく、生徒の誤解(misconception)の表れであるとみなした。「誤解」という概念は、正しい解が存在することを意味する。このような試行錯誤の見方は、プログラミング言語の構文を丸暗記するときのParsons puzzleの使い方によるものかもしれない。

### 6.1.3 バックトラックを意図的な試行錯誤とみなす研究

Helminenら**[52]**は、Parsons problemを解くプロセスを解答の状態遷移として表現し、ダイアグラムとして可視化することを提案した(後の研究方法手法の節6.3.2で述べる)。個々の遷移には、同じ遷移で以前に訪れた状態に戻るループ(本稿ではこれを「閉路」と呼ぶ)が存在する。彼らはさまざまな種類のループを特定した。バックトラックは、「操作を正確に逆の順序で元に戻すことで以前の状態に戻る 」ループの一種である。循環ループは、「異なる中間状態を経由して以前の状態に戻る」ループの一種である。Helminenらは、バックトラックが意図的なものであると仮定することができるのに対し、循環ループが意図的な試行錯誤であるか偶然の産物であるかの判断は難しいと述べている。

本研究は、意図的な試行錯誤を示す循環ループもあることを明らかにする。

### 6.1.4 正しい解答の存在に依存する研究

久野は、短冊問題によって**[53]**[25]、暗記ではなく、思考力、判断力、表現力を評価することを提案した。例えば、理路整然と議論する能力を評価することを提案した。短冊問題は、Parsons problemやジグソー・コードと同じプログラミング演習に属する(6.3.1で述べる)。短冊はジグソー・コードのピースと似ている。

上野ら[46]は、短冊形問題の解答プロセスを解の状態遷移として表現し、それをダイアグラムとして可視化することを提案している(研究方法の節で後述)。彼らは解答を短冊の順序として表現している。研究方法の節で説明するプロセスとは異なり、彼らの方法は、すべての学生の遷移を1つのダイアグラムに集約する。このような図を使用することで、例えば、学生はdefやendのような外側の部分からプログラムを構成する傾向があることがわかった。しかし、彼らは、つまずきポイント（試行錯誤）を特定するには、ダイアグラムに含まれる情報が多すぎると指摘し

た。彼らは、生徒が正解と不正解の両方を出した遷移によって試行錯誤を定義した。したがって、答えのない問題に彼らの方法を適用するのは難しいかもしれない。

## 6.2 目的

本章の目的は、解答の状態遷移における閉路が、生徒の試行錯誤のプロセスをどのように示唆するかを示すことである。閉路とは、生徒の誤解ではなく、生徒の知識の手がかりを意味する。閉路を形成した生徒は、最初の状態が正しい解に到達するのに役立つと考え、それを分岐点としていたのである。

## 6.3 研究方法

この章では、ジグソー・コードのパズルを解くプロセス(6.3.2)における解の状態遷移と閉路を定義し(6.3.1)、そして、閉路がプレイヤーの試行錯誤のプロセス(6.3.3)の推定に役立つことを示す。その後、閉路の最初の状態がプレイヤーの最終解の部分列であることを確認することで、それを検証する(6.3.4)。本研究では、学部1年生向けのプログラミング入門コースで、230人の学部生がプレイした11のジグソー・コードのパズル問題のデータを分析する(6.3.5)。

### 6.3.1 ジグソー・コード

本研究ではジグソー・コードを開発して授業で使った。これは偽物を含む解答プログラムのコードブロック(ピース)をシャッフルしたものを生徒に与え、そこから取捨選択して正しい順番に並べ替えさせるプログラミング演習である。それに伴い、各生徒が行った操作が記録される[16]。

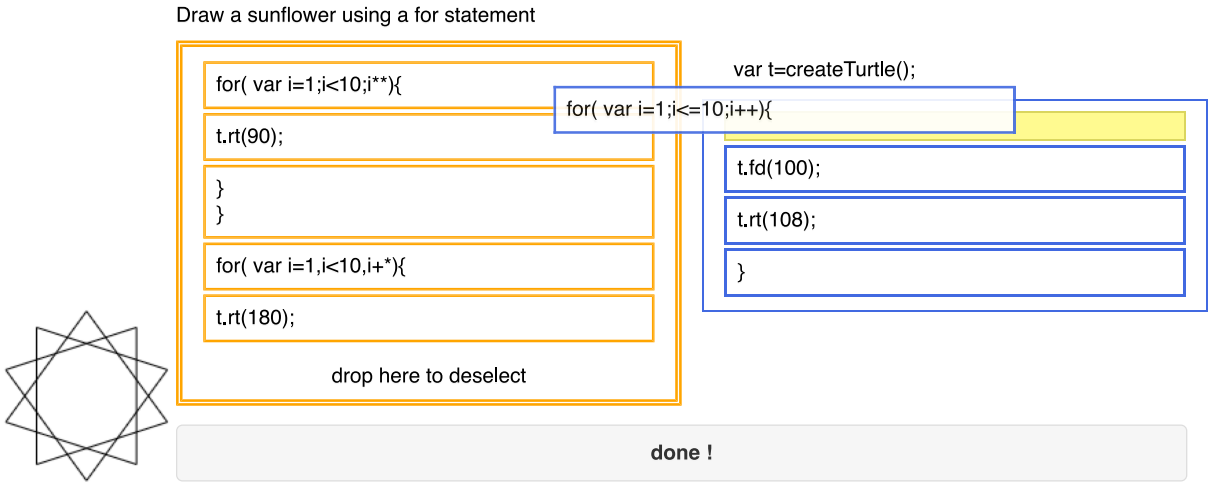



**図55**　ひまわりの絵(左)とプログラミング教材「ジグソーコード」のパズル(右)。ジグソー・コードでは、左側の枠からピースを選択して右側の枠に移し、右側でピースを並べ替えたり、選択を止めて左側に戻したりして、ひまわりが描かれるように右側に正しい順序のピースを完成させる。「var t=createTurtle(); 」はあらかじめ右側の枠の上に配置されており、動かすことはできない。

図55はジグソーパズルの画面である。ジグソー・パズルと同様に、ジグソー・コードはコード全体をコードの断片(ピース)に分解する。生徒(プレイヤー)がパズルを開くと、いくつかのピースをシャッフルし、左側の枠である選択肢ボックスに配置される。時には、最初から正解のピースが解答枠(右の枠)の上、中、下に置かれていることもある。オリジナルのジグソー・パズルとは異なり、偽のピースが混ざっていることもある。プレイヤーは正しいピースを選択肢の箱からドラッグし、解答の枠にドロップする。左から選んだピースが間違いであることが判明した場合、プレイヤーはそのピースを選択肢の枠にドラッグ＆ドロップで戻すことができる。ピースをドラッグ＆ドロップすることで、右の枠内でピースを並べ替えることができる。これらの操作を繰り返すことで、パズルが完成する。ジクソー法**[54]**と同様に、ジグソー・コードもジグソー・パズルにちなんで命名されている。すなわち、生徒が自分の解答つまり完成絵のピース(完成絵の一部)処理する様子にちなんでいる。

ジグソー・コードは解答プロセスにおけるプレイヤーの各操作ごとに、次の情報を収集している

- 操作の種別
- 操作対象のピースのID
- 解答枠内のピースの並び順
- タイムスタンプ

ジグソー・コードは、これらの操作を、そのプレイのためにユニークに生成されたドキュメントIDに関連付けて収集する。

### 6.3.2 解の並び順の状態遷移と閉路

**解の状態**

解答の状態とは、解答枠内のピースの並び順のことである。ジグソー・コードでは、解答のプロセスでプレイヤーが操作すると、そのときの状態が記録される。ピースにはIDがあり、通常プレイヤーには隠されている。ここではピースのIDの並び順で状態を表現する。**図56**は、パズル「2-2 Sunflower」を解いている途中の状態を示している。s1、s2、...はそれぞれピースのIDである。「for 文でひまわりを描け(Draw a sunflower using a for statement)」という指示も、やはりピースであり、同様に ID を持っている。解答枠内のID「s8, s3, s5, s6」の順序が、その時点での解の状態である。

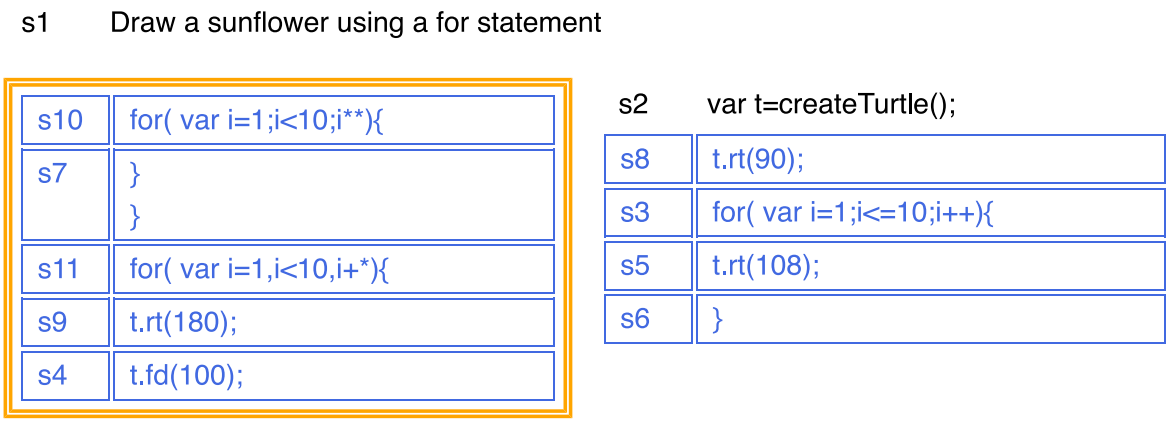



**図56** 解答途中の状態。ピースID「s8,s3,s5,s6」の並び順がこの時の状態であり、最終解としては正しくない。

**状態遷移と図**

解の状態は進行とともに変化する。その状態遷移を図で表す(**図57**)。図中、円は状態を表す。円の中のラベルは、状態を表すピースの並び順を表す。矢印は、ある状態から別の状態への遷移を表す。本稿では、状態図は現実の遷移を表し、ソフトウェア工学で使用される状態図は可能な遷移を表す。この図は、すべての生徒の遷移を集約しているのではなく、1つの遷移に対応している。図57に状態遷移図の3つの例を示す。矢印上の数字はプレイヤーが遷移した回数を数える。右の状態遷移図では、プレイヤーは1回のプレイ(解答プロセス)で状態「s2, s3」から「s2, s3, s10」に2回遷移した。これらの図は "d3-graphviz "によって描かれたものである。[*10] (and D3.js) on a web browser.

*10 GitHub - magjac/d3-graphviz: Graphviz DOT rendering and animated transitions using D3 https://github.com/magjac/d3-graphviz ↗

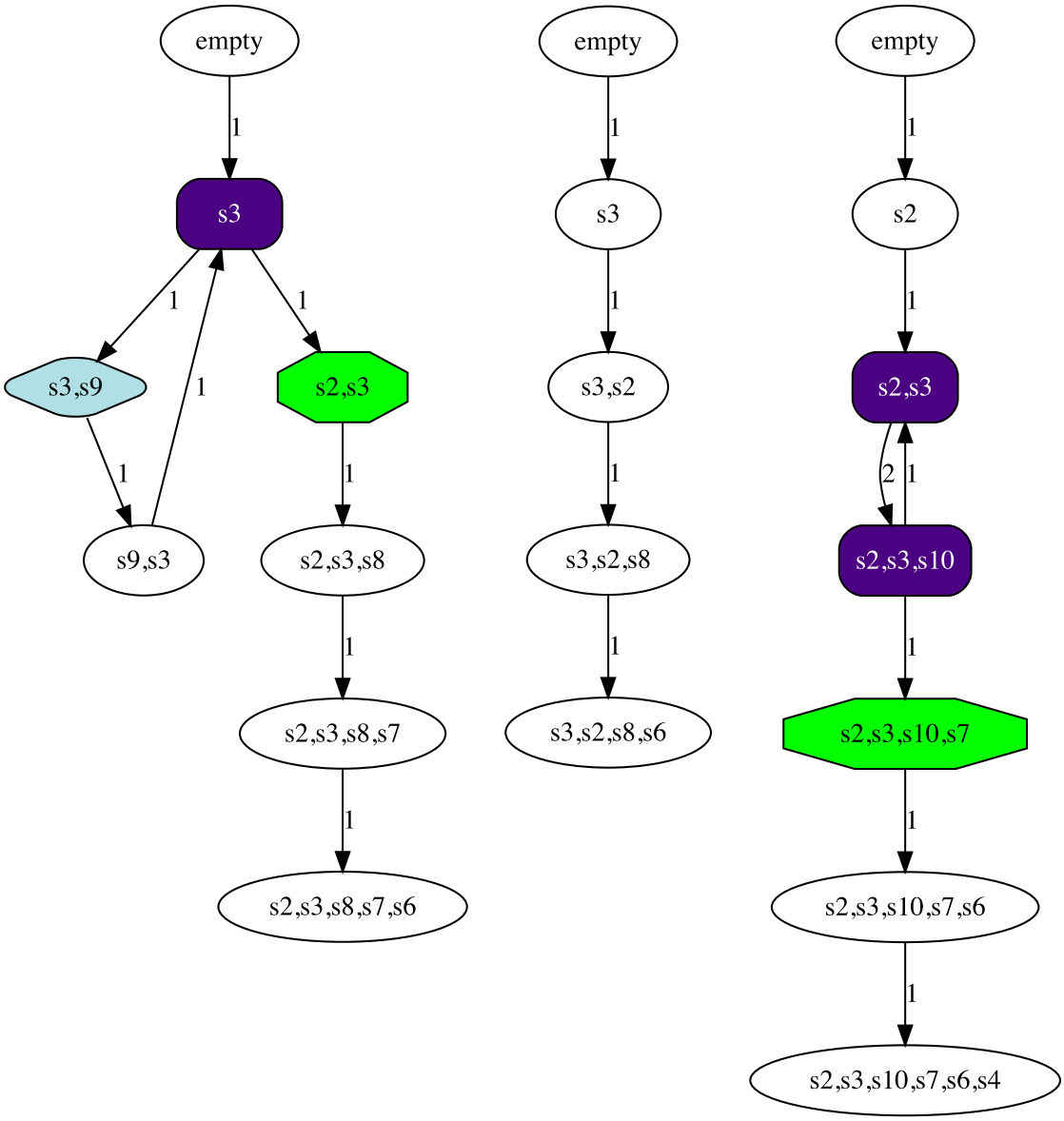



**図57**　状態遷移図の例。左は1つの閉路、中央は閉路なし、右は2つの閉路を含む。楕円は通常の状態、丸い箱は閉路の最初と最後の状態、菱形は閉路の最初の状態の次の状態、八角形は閉路の最後の状態の次の状態である。

**閉路**

状態遷移グラフにおける閉路とは、最初と最後の状態を除くすべての状態（頂点）が異なる、空でない閉じた状態列のことである。図57左の図には閉路が1つあり（Helminenらはこれを循環ループと呼んだ[52]）、中央にはない。右図には2つの閉路がある：（s2、s3」、「s2、s3、s10」、「s2、s3」）と（「s2、s3、s10」、「s2、s3」、「s2、s3、s10」）である（Helminenらはこれらをバックトラックと呼んだ[52]形は状態のタイプを示す。角の丸い箱は閉路の最初と最後の状態を表し、菱形は最初の状態の次の状態を表し、八角形は閉路の最後の状態の次の状態を表す。1つの状態は、これら3つのタイプ（丸い箱、ひし形、八角形）のうちの2つ以上に属することがあるため、正確に形状コード化されているわけではない。**図58**は図57から3つの閉路を取り出した

ものである。左の閉路は左の状態遷移から、残りの2つの閉路は右の状態遷移からのものである。

**閉路において最初に動かされたピース**

閉路で最初に動かされるピースは、最初の状態(丸い箱)と次の状態(ひし形)の差を表す。図58の左の閉路では、s9が最初に動かされる。右側の閉路では、最初の操作は削除である。s10は解答枠内のピースから削除され、この閉路で最初に動かされるピースとなる。

**閉路を抜けた後で最初に動かされるピース**

閉路からの抜け出した後に最初に動かされるピースは、最後の状態(最初の状態も丸い箱)と最後の状態の次の状態(八角形)との差を表している。図58の左の閉路では、脱出の直後にs2が移動している。最後の状態（丸い箱）がプレイヤーの最終解答の場合は、閉路からの脱出後に動かされるピースはない。

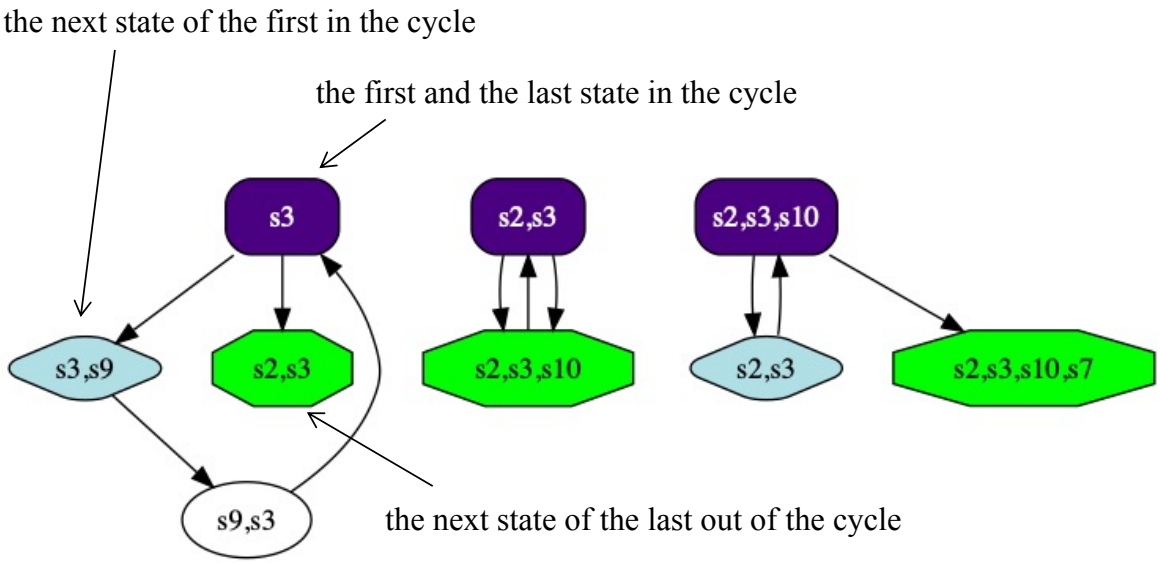



**図58**　閉路の開始状態(the first state)と終了状態(the last state)、開始後の最初の状態(the next state of the first state of the cycle)、閉路脱出後の最初の状態(the next state of the last state of the cycle)。

### 6.3.3　試行錯誤

状態遷移における閉路は、プレイヤーの試行錯誤を推測する手がかりとなる。閉路は、プレイヤーがその閉路の最初の状態に戻ったことを示す。最初の状態から移動して正解を探り(試行)、ある状態に到達してそれが間違った状態であることに気づき(エラー)、最初の状態(最後の状態)に戻る、この状態遷移が1つの閉路を形成している。そのとき、プレイヤーは閉路の最初の状態、つまり現在の状態が正しい最終解につながると考えたはずであり、だからこそ最初の状態に戻ったのである。

**図59**の左の状態遷移で、プレイヤーは「s3, s4」に到達した。そして、「s3, s4」から「s3, s4, s5」へ直進せず、「s3, s4, s8」へ回り道した。彼らは「s3, s4, s8」を試してみて、それが間違っていることに気づき、「s3, s4」に戻ったのである。

### 6.3.4 検証

閉路の最初の状態がプレイヤーの最終解の部分解である場合、プレイヤーは最初の状態を正しい最終解につながると考えた可能性が高い(図59)。プレイヤーの最終解が正しいかどうかは問題ではない。ここで、与えられた状態(ピースの並び順)の部分列とは、与えられた状態から、残りのピースの順序を変えることなく、いくつかのピースを削除するか、全く削除しないことによって導き出せるものである。

図59の左の遷移図において、プレイヤーが「s3, s4, s8」から「s4, s3」に戻らなかったのは、おそらく「s4, s3」が正しい解につながると考えていなかったからであろう。つまり、「s4, s3」が最終解「s3, s4, s5, s6」の部分列ではなかったので、「s4, s3 」に戻らなかったのは偶然ではない。

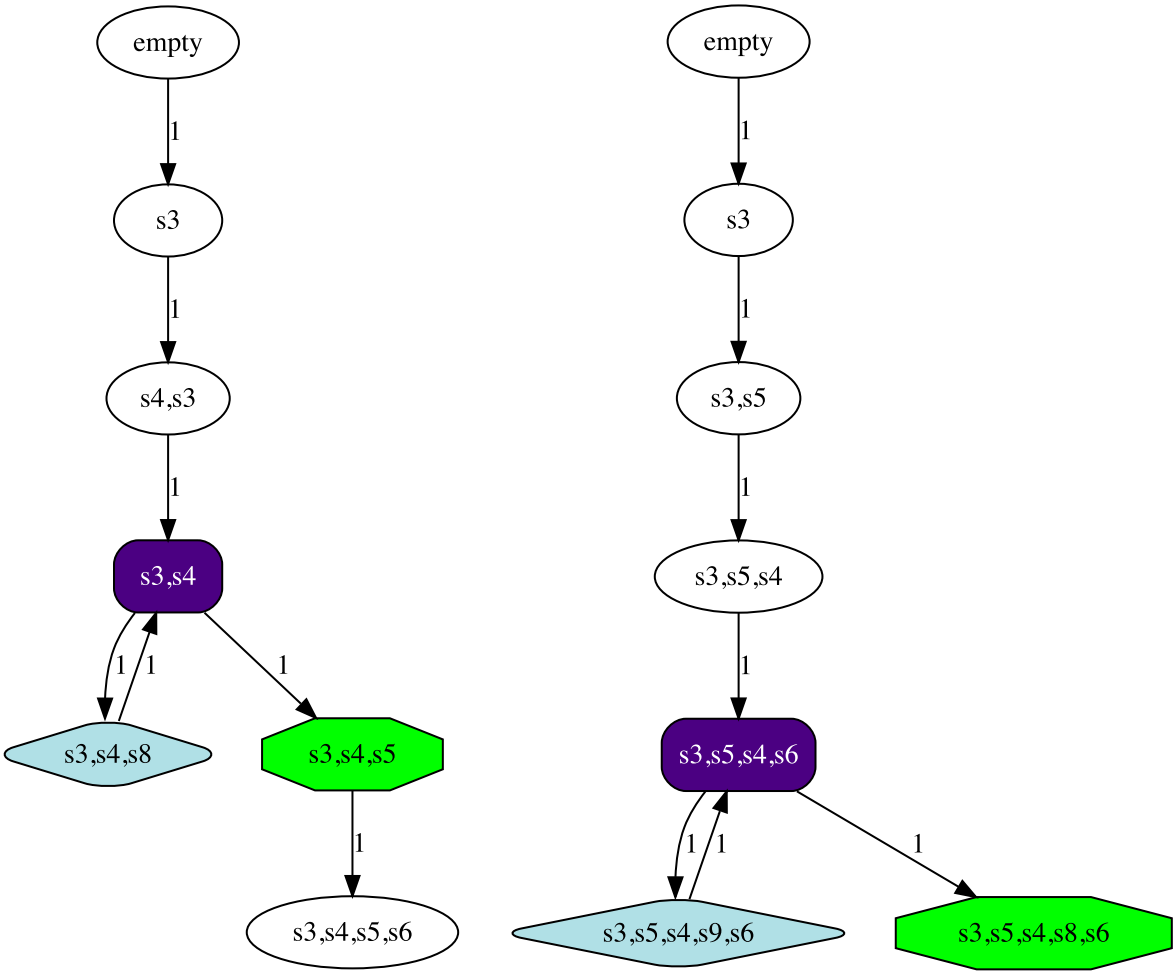



**図59**　状態遷移には、最初の状態（丸い箱）が最終解の部分列である閉路が含まれる。右図で「s3, s5, s4, s6」は「s3, s5, s4, s8, s6」の部分列である。

次に、ジグソー・コードによって収集されたデータを評価し、試行錯誤の推定方法を検証する。頻出する閉路を取り出し、その閉路の最初の状態が最終解の部分解であるかどうかを調べる。ここで、「頻繁に出現する」とは、その閉路が平均＋標準偏差2回以上出現することを意味する。ある閉路が頻繁に現れるなら、それは偶然ではなく、理由があるはずである。さらに、頻繁に現れる閉路の最初の状態が最終解の部分配列であることが多いなら、それも偶然ではない。

### 6.3.5　データ

学部1年生を対象としたプログラミング入門コースのショートテストで使用されたジグソー・コードによって収集されたデータを評価する。230名の学生が2020年にオンラインで受講し、合計11問のパズルを解いた。このコースでは、JavaScript、HTML、Turtle Graphics Libraryを使って、変数、条件分岐、ループ、関数について学ぶ。

生徒たちは授業の最初に短いテストを解いた。パズルの目的は、前回の授業の復習であった。教師は、生徒（プレイヤー）が正しいピースと偽のピースを注意深く選んだり外したりさせられるように、偽のピースを含むパズルをデザインした。パズルは、Turtle Graphics Library(図55)

を使ってひまわりを描く、砂時計を描く、旅行の割引率を計算するなど、さまざまなトピックを扱った。

### 6.4 結果

頻出する閉路を含む97%のプレイについて、その閉路の最初の状態は最終的な解の部分配列である(**表4**)。

**表4** 問題一覧
最後の列は、閉路の最初の状態が最終解の部分配列であるプレイの比率である。A: 頻出する閉路、B: 頻出する閉路(A)を含むプレイ(状態遷移)、C: 頻出する閉路(A)を含み、その閉路の最初の状態が最終解の部分状態であるプレイ。

| puzzle | No. of A | No. of B | No. of C | C/B (%) |
|---|---|---|---|---|
| 1-1 砂時計の容器を作るプログラム | 3 | 69 | 68 | 99 |
| 1-2 濃度判定プログラム | 2 | 20 | 20 | 100 |
| 1-3 整数入れ替えプログラム | 4 | 52 | 45 | 87 |
| 2-1 PCR 検査陽性率 | 4 | 30 | 29 | 97 |
| 2-2 ひまわりを描くプログラム | 2 | 8 | 8 | 100 |
| 3-2 二乗を計算するプログラム | 1 | 9 | 9 | 100 |
| 3-3 九九出力プログラム | 2 | 31 | 31 | 100 |
| 4-1 ドーナツを描くプログラム | 2 | 27 | 23 | 85 |
| 4-2 閏年判定プログラム | 1 | 11 | 11 | 100 |
| 5-1 GO TO travelの料金判定プログラム | 1 | 22 | 22 | 100 |
| 5-2 GO TO travelの料金判定プログラム2 | 4 | 22 | 22 | 100 |
| Average | | | | 97 |

### 6.5 考察

97%は高い。例えば、「s3, s5, s4, s6」は「s3, s5, s4, s8, s6」の部分列であり、4! - 1 = 23個の異なるアナグラムがあり、これらは部分列ではない。閉路の最初の状態が最終解の部分列であるのは偶然ではなく、意図的なものである。

図59の閉路によると、そのプロセスで何が起こったのだろうか？閉路で最初に動いたピース（菱形）と、閉路から脱出して最初に動いたピース（八角形）のペアを観察してみよう。**図60**は「2-2 Drawing Sunflower」の最終的な解答である。データから得られたペアの半分は(s8, s5)であった。図59の左の状態遷移図は(s8, s5)の例である。

ピースs5「t.rt(108);」はカメに108度右を向くように命令し、ピースs8「t.rt(90);」は90度右を向くように命令する。図59の左の図を見ると、選手たちは最初(試行)s8を選択したが、後で(間違いに気づいて)選択を解除し、「s3,s4」に戻り、今度はs8ではなくs5を選択した。試行錯誤

の結果、右を向く角度がわかった。教師は、生徒の理解度を試すために、偽のピース s8 を混ぜた。

彼らは「s3, s4」が正しいと信じていたので、「s4, s3」にまでは戻らなかった。「s3,s4」は、図60 に示す彼らの最終解「s3,s4,s5,s8」の部分解であることから、彼らの信念が推測される。

さらに、s5 と s8 が選手の選択肢だったように思える[34]。その検証は今後の研究に委ねたい。

解の状態遷移の閉路は、生徒が他の解を探索した後に閉路の最初の状態に戻ったことを示しており、おそらく閉路の最初の状態が正しい最終解につながると考えたからであろう。逆順に正確に閉路の最初の状態に戻る場合は、バックトラックであり、計算論的に考えている(think computationally)ようである。逆順に戻らなかった場合、計算的に考えたかどうかは不明である **[55]**。逆順に戻らなかった場合、計算的に考えたかどうかは不明である。例えば、閉路全体がバックトラックの1ステップかもしれない。これについては今後の研究に委ねたい。

s1 Draw a sunflower using a for statement

| | |
|---|---|
| s7 | }<br>} |
| s8 | t.rt(90); |
| s9 | t.rt(180); |
| s10 | for( var i=1;i<10;i**){ |
| s11 | for( var i=1,i<10,i+*){ |

s2 var t=createTurtle();

| | |
|---|---|
| s3 | for( var i=1;i<=10;i++){ |
| s4 | t.fd(100); |
| s5 | t.rt(108); |
| s6 | } |

**図60** 「2-2 Drawing Sunflower」の最終的な解答の例

試行錯誤は常に新しいピースを選ぶことから始まるわけではない。ピースを取り除くことから始まる閉路もある。**図61** では、おそらくプレイヤーは丸い箱の状態で s5 を不要と考え、取り除いて菱形の状態に遷移したが、後で取り戻して再び丸い箱の状態に遷移した

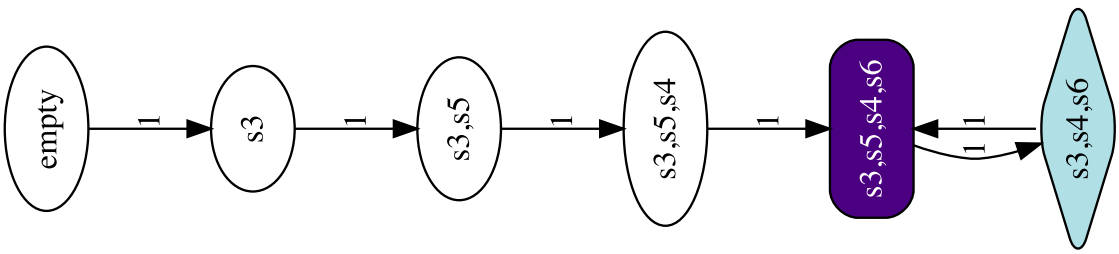



**図61** 「2-2 Drawing Sunflower.」の状態遷移図(反時計回り反転している)。ピース s5 を捨てることから始まる閉路が含まれている。

閉路を形成しない試行錯誤があるかもしれない。**図62**では、プレイヤーは s6 を「s10, s4, s5」の状態に加え、s10 を「s10, s4, s5, s6」の状態から取り除き、s3 を「s4, s5, s6」の状態に加えた。このプロセスは閉路を形成しない。「s10,s4,s5,s6」という状態はプレイヤーにとってはエラーかもしれないが、いつ試行が始まったのかは不明である。このような試行錯誤は今後の研究に委ねたい。

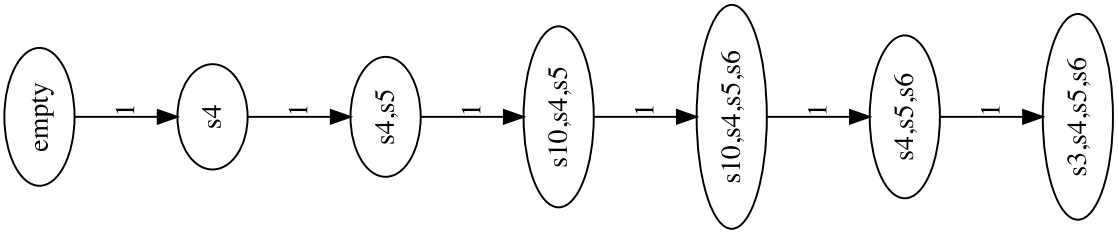



**図62**　「2-2 Drawing Sunflower」の状態遷移図（反時計回りに反転）。閉路は含まれていないが、ピース s10 を取って捨ててるため、試行錯誤を表している可能性がある。

### 6.6　結論

閉路は、プレイヤーの誤解というより、むしろプレイヤーの知識の手がかりとなる。バックトラックよりも多くの試行錯誤を機械的に指摘できる。この発見は、試行錯誤を好意的に評価する日本の教育の要求に応え、試行錯誤の戦略を評価するものである。

今後の研究では、閉路が、プレイヤーが試みた選択肢など、プレイヤーが何をしようとしているのかを知る手がかりになることを期待している。上記の結果や考察は、プレイヤーの最終的な解が正しいかどうかとは無関係であるため、解のない問題にこの方法を適用することを期待している。

この研究の限界は、試行錯誤のプロセスが常に閉路を形成するとは限らないことである。さらに、状態遷移の閉路が試行錯誤を示すことを明らかにしたのは、プログラミング演習に関してだけである。これが他の問題解決プロセスにも当てはまるかどうかは不明である。

# 7 操作間隔

ここではアプリケーション操作の時間間隔を分析する。ジグソー・テキストやジグソー・コードのピース操作の時間間隔にはプレイヤーの熟達度が現れる。ピースをソフトウェア開発のスクラムにおけるプロダクト・バックログ・アイテムとして、作る順番を決めるというパズル問題を解くとき、プロダクト・オーナー経験者の方が未経験者よりも、操作の時間間隔の分散が小さく平均値も小さいことが分かった。ピース操作の時間間隔にはプレイヤーの熟達度が現れるが、この熟達とはジグソー・テキストというアプリケーションの熟達ではなく、パズル問題の内容、この場合はプロダクト・オーナーの経験である。ただし、熟達度は試行錯誤の内容やプレイ戦略そのものではなく、本研究の狙いからは若干離れる。

スクラムでのプロダクト・バックログの順位付けにおいては、プロダクト・バックログはリファインメント(追加・変更・削除)があることが前提となる。このため、プロダクト・オーナー(Product Owner, PO)経験者は、プロダクト・バックログの順位付けにあたって、あまり時間をかけず、取り掛かるまでのスピードが速く、変更することへのハードルが低く、決断が速くて迷いが少ないという特徴があると予想した。これを確認するため、プロダクト・バックログ並べ替えの実験やワークショップを開催し、並べ替え操作を観察した。分析の結果、PO経験者は未経験者に比べて、操作間隔の平均が有意に小さく、個々の決断が速くて迷いが少ない傾向が認められた**[56]**。

## 7.1 はじめに

「現代は、先行きが見通せない予測困難な時代」であり「従来からの価値のみに縛られず、新たな課題の発見・解決を通じた『価値創造』に対応できる人材の重要性が増してきて」いる。「このような時代・社会からの要請に対して、『答えのない課題解決に挑む』人材をどのように育成していくのか」**[57]**は、現代に生きるわれわれの課題と言えるだろう。

答えのない課題への取り組み方の研究には、正解がないがために、解答の正誤といったアウトプットに基づく比較や評価ができない困難さがある。

大日本印刷株式会社では、新規事業創出にアジャイル開発の考え方を取り入れており、スクラムを採用している。そこでは、事業をリードするプロダクト・オーナー(Product Owner, 以下PO)の育成が課題となっている**[58]**。

POの仕事にプロダクト・バックログの管理・順位付けがある。プロダクト・バックログとはプロダクト・ゴールの実現に必要なものの一覧である。プロダクトへの要求は開発の途中であっても変化すると、スクラムでは認識する。スクラムは経験主義に立って、経験したこと、すなわち観察された物事に基づいて判断する。そこで、プロダクト・バックログは、追加・変更・削除といったリファインメントがあることが前提となる**[59]**。また、スクラムでは短い固定の期間に区切って作業するのが特徴である。「たとえスプリントの最終日に作業が残っていても、スプリントは終了し、期間は延長し」ない**[60]**。このようなスクラムにおいて責任を持ってきたPO経験者がプロダクト・バックログに順位をつける(並び替える)ときには、そのプロセスに、未経験者と比べて次の特徴があると予想される:

(1) あまり時間をかけずに並び替える。
(2) 取り掛かるまでのスピードが速い。
(3) 変更することへのハードルが少ない。
(4) 決断が速く、迷いが少ない。

プロダクト・バックログの管理は、広い意味で計画作業といえる。リファインメントが前提ということは、変更されることが前提の計画作業ということである。


**図63**　プロダクト・バックログの順位付け問題「雑貨のECサイトの制作」をドラッグ&ドロップで解く画面。「消費者は取り扱っている商品の一覧を見たい。なぜなら…」がドラッグ&ドロップによって移動中のピース

プロダクト・バックログは正解がないことが特徴である。プロジェクトの状況や、POによって重視することが異なるからである。

プロダクト・バックログの順位付けという計画作業は「答えのない課題解決」なので、順位付け結果(アウトプット)の正誤に基づいて人材を評価するのは困難である。しかし、(アウトプットではなく)プロセスにPO経験者と未経験者とで違いがあるならば、「答えのない課題解決に挑む」人材育成の手がかりとなるだろう。

本章はPOによるプロダクト・バックログの順位付けについて分析した既発表の文献**[61]**に、その後の分析結果を加えて再構成したものである。

## 7.2　目的

本研究の目的は、PO育成のために、PO経験者と未経験者との認知的な違い・考え方の違いを、作業プロセスに見つけることである。

プロダクト・バックログの順位に正解はないため、本研究は「答えのない課題の解決に挑む」**[62]**プロセスにおける経験者と未経験者との違いを見つけるものと言える。

## 7.3 先行研究と事例

本研究は、プロダクト・バックログの管理・順位付けという計画作業のプロセスに、経験者と未経験者の違いを見出そうとする。計画作業に関する熟達、あるいは能力とはどのようなものであろうか。そこで、スクラムに限らずに、計画作業に関する研究として、生活時間(time use)や時間管理(time management)の先行研究や事例を検討する。

### 7.3.1 生活時間

小学校学習指導要領の「8節 家庭」では生活時間を取り上げて「家庭には、家庭生活を支える仕事があり、互いに協力し分担する必要があることや生活時間の有効な使い方について理解すること。」としている**[63]**。「生活時間(time use)」は「何にどれくらい時間を使っているか」という一日の時間配分を指す用語で、役割分担と関連付けられることが多く、家庭教育での取り上げ方もその方向と考えられる**[64]**。

このような研究は、本研究とは方向が異なると考えられる。

### 7.3.2 時間管理

岡崎**[68]**によれば、時間管理研究には大きく2種類あり、1つは時間管理がもたらす効果の研究、1つは時間管理能力の研究であり、時間管理能力の研究が少ない。岡崎は、時間管理能力には、やり遂げるために必要な時間を見積もることが含まれるが、時間を正確に見積もることは難しいとしている。さらに、見積もりの誤差が小さくなれば適切に時間管理できるのかについては全く研究されていないとしている。この点が、本研究としては興味深い。本研究では、プロダクト・バックログが変わることを前提としているからである。ソフトウェア開発のウォータフォール・モデルでは前工程の正しさを前提とし、見積もりの正確さを問題にする。この意味で、従来の時間管理研究はウォーターフォール的と言える。

時間管理を時間アセスメント行動、プランニング行動、モニタリング行動の3つに分類したとき、プランニング行動には順位付けやto-doリスト作成が含まれる[68]。プロダクト・バックログの順位付けはプランニングに該当するので、プロダクト・バックログの順位付けという作業における時間の使い方とは、「計画の計画」に相当することになる。本研究の対象は、時間の使い方についてメタな構造を持つと言えよう。

### 7.3.3 試験時間の使い方

入試などで解答開始から終了までの試験時間／解答時間の使い方については、ノウハウがWebサイトなどで見つけられる。例えば、どのような問題から先に解くのがよいかといったノウハウである。しかし、それらノウハウの検証や評価を見つけられなかった。

#### 7.3.4 まとめ

従来の研究や取り組みはウォーターフォール的な計画・時間管理に関するものが多く、時間のスケールが数時間から数十日のものである。また、本研究の対象であるプロダクト・バックログの順位付けは、計画の計画というメタな作業と言える。

### 7.4 研究方法

後述のジグソー・コード(**図63**)やジグソー・テキスト**[69]**を使って、プロダクト・バックログ・アイテムをパズルのピースとして並べ替えることで、プロダクト・バックログを順位付けするワークショプや実験を行う。その後、ジグソー・コード等の操作ログから、並べ替え操作のプロセスを分析する。

冒頭で想定したPOの特徴から、並べ替え操作のプロセスには次の特徴が見られると予想する:

(1) あまり時間をかけずに並び替える
⇒ 検証1 解答に要する時間が短い
(2) 取り掛かるまでのスピードが速い
⇒検証2 最初のピースを動かすまでの時間が短い
(3) 変更することへのハードルが少ない
⇒ 検証3 操作回数が多い
(4) 決断が速く、迷いが少ない
⇒ 検証4 操作間隔が短い

分析ではこれらについて検証する。本章では以降で「検証1」などとして参照する。

#### 7.4.1 使用システム

ジグソー・コードやジグソー・テキストは、作業項目、文章の断片、プログラム・コードの断片、できごとなど、順序のあるものをピースとして、取捨選択・並べ替えて、適切と思う順序に完成させるパズル型のWebアプリケーションである[69]。図63にジグソー・コードのユーザー・インタフェースを示す。

#### 7.4.2 実験

架空のサービスをプロダクト・ゴールとしたプロダクト・バックログを想定し、そのアイテムをピースとして、ジグソー・コードのパズル問題を開発する。そして、そのプロダクト・バックログの順位付けを行う「予備実験」や「追加実験」などの実験、および「PO育成支援ワークショップ」[58]の「事前課題」および「事後課題」を開催する。実験やワークショップの事前・事

後課題などを、ここでは「セッション」と総称する。被験者は、勤務時間中に自由に時間を取って解いたり、ワークショップの場に(リアル and/or オンラインで)集まって解いたりする。

研究の、より上位の目的はPO育成である。パズル問題の狙いもそこにある。そのため、セッション後の被験者や参加者の感想などを踏まえて、次のセッションではパズル問題を改善することがある。結果的に全セッションを通して共通に解かれる問題はなくなっていく。

### 7.4.3 分析

検証1については、解答に要した時間をPO経験者と未経験者とで比較する。

検証2については、プレイ開始から最初のピースを動かすまでの時間をPO経験者と未経験者とで比較する。

検証3については、解答に要した操作回数をPO経験者と未経験者とで比較する。

検証4については、操作間隔(**図64**)の平均、中央値、標準偏差(バラツキ)をPO経験者と未経験者とで比較する。

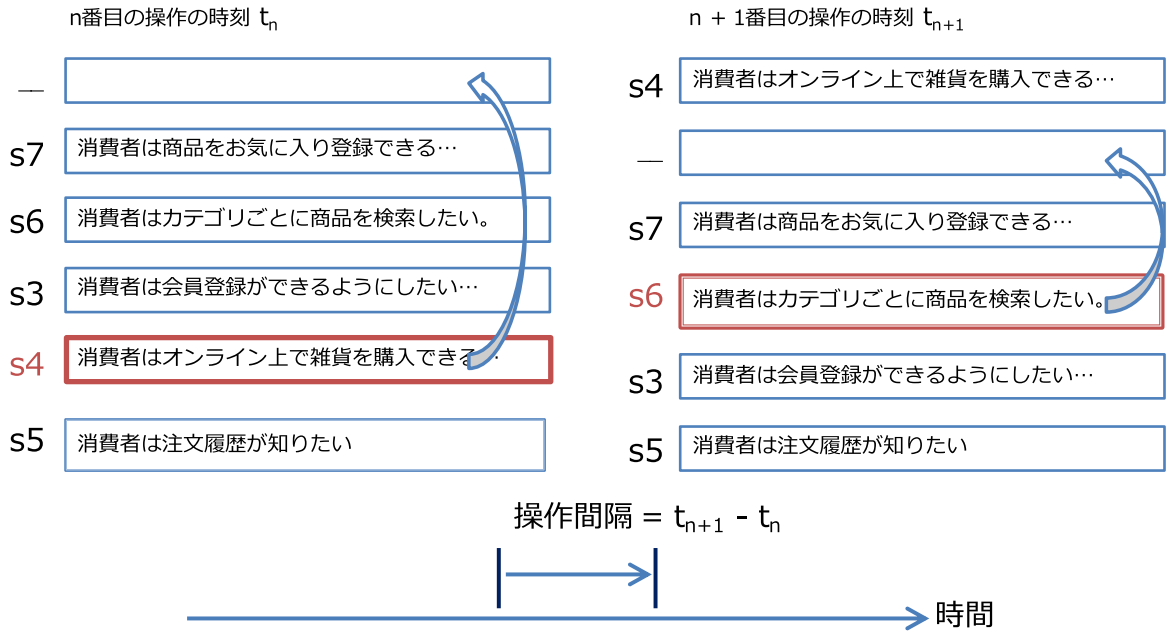



**図64** 順位付け操作の操作間隔

## 7.5 結果

まず、実施したセッションや解かれたパズル問題(プロダクト・バックログ)や被験者の内訳、およびデータ処理について述べる。その後で、検証1から検証4について結果を示す。

### 7.5.1 パズル問題とセッション

開発したパズル問題を表4に示す。前述の通りパズル問題を改善し続けた結果、全部で14個の問題を開発した。問題タイトルが同じでも内容は異なる問題である。図63は、表で11_ecと略記された「雑貨のECサイトの制作」を解いている画面であった。

**表5** 問題一覧

| 問題略記 | 問題タイトル |
|---|---|
| 01_fre_ma | オンラインフリーマーケット1 |
| 02_hotel | ホテルのオンライン対応 |
| 03_ec | 雑貨のECサイトの制作 |
| 04_fre_ma | オンラインフリーマーケット2 |
| 05_1_fre_ma | 1. オンラインフリーマーケット1 |
| 06_2_hotel | 2. ホテルのオンライン対応 |
| 07_3_ec | 3.雑貨のECサイトの制作 |
| 08_4_fre_ma | 4. オンラインフリーマーケット2 |
| 09_5_fre_ma | 5. オンラインフリーマーケット3 |
| 10_6_ec | 6.雑貨のECサイトの制作 |
| 11_ec | 雑貨のECサイトの制作 |
| 12_kyoto_3step | 旅行の準備 複数ステップ |
| 13_ec_3step | 雑貨のECサイトの制作 複数ステップ |
| 14_ec | 雑貨のECサイトの制作 |

実施したセッション(実験とワークショップ)を**表6**に示す。全部で10回のセッション、すなわち問題を解く機会を用意した。

**表6**　実験/ワークショップ一覧

| セッション略記 | 実験/ワークショップ | 実施月 |
|---|---|---|
| 010_preYobi | プレ予備実験 | 2022年6月 |
| 020_yobi | 予備実験 | 2022年7月 |
| 030_add_sep | 9月の追加実験 | 2022年9月 |
| 041_ws1pre | 第1回ワークショップ 事前課題 | 2022年10月 |
| 042_ws1after | 第1回ワークショップ 事後課題 | 2022年10月 |
| 051_ws2pre | 第2回ワークショップ 事前課題 | 2023年1月 |
| 052_ws2after | 第2回ワークショップ 事後課題 | 2023年1月 |
| 060_add_feb | 2月の追加実験 | 2023年2月 |
| 070_ws3 | 第3回ワークショップ | 2023年3月 |
| 070_ws3after | 第3回ワークショップ 事後課題 | 2023年3月 |
| 080_po_ws | PO育成支援ワークショップ | 2023年11月 |

**表7**にセッションとパズル問題の対応関係を示す。セッションのたびに問題を改善していったので、問題とセッションとがほぼ対応してる。

全セッション共通の参加者は少なかった。また全セッション共通の問題というものはない。このあたりは、全体を通した分析の難しさにつながった。

**表7**　セッションと問題の対応

| セッション略記 | 実験/ワークショップ | 問題 |
|---|---|---|
| 010_preYobi | プレ予備実験 | 01_fre_ma |
| | | 02_hotel |
| | | 03_ec |
| | | 04_fre_ma |
| 020_yobi | 予備実験 | 05_1_fre_ma |
| | | 06_2_hotel |
| | | 07_3_ec |
| | | 08_4_fre_ma |
| | | 09_5_fre_ma |
| | | 10_6_ec |
| 030_add_sep | 9月の追加実験 | 06_2_hotel |
| 041_ws1pre | 第1回ワークショップ 事前課題 | 11_ec |
| 042_ws1after | 第1回ワークショップ 事後課題 | 11_ec |
| 051_ws2pre | 第2回ワークショップ 事前課題 | 11_ec |
| 052_ws2after | 第2回ワークショップ 事後課題 | 11_ec |
| 060_add_feb | 2月の追加実験 | 12_kyoto_3step |
| | | 13_ec_3step |
| 070_ws3 | 第3回ワークショップ | 14_ec |
| 070_ws3after | 第3回ワークショップ 事後課題 | 14_ec |
| 080_po_ws | PO育成支援ワークショップ | 14_ec |

**表8**に被験者におけるPO経験者および未経験者(図でNP)のユニークな人数と、プレイした(解いた)数の総数を示す。研究期間中にPO経験した被験者がいたので、経験者の人数+未経験者の人数=総人数とはならない。また、**表9**にセッションごとのPO経験者の内訳を、**表10**に問題ごとの内訳を、それぞれ示す。いずれもPO経験者の人数は少なく、1人というものもあった。

**表8**　被験者の人数とPO経験の内訳、およびプレイの数。PO経験なしをNP、経験ありをPで示す

| PO経験 | 被験者のユニークな人数 | プレイの数 | プレイの数の割合 |
|---|---|---|---|
| NP | 50 | 120 | 77.7% |
| P | 9 | 38 | 22.3% |
| 総計 | 58 | 158 | 100.0% |

**表9**　セッションごとのPO経験者の人数

| セッション | NP | P |
|---|---|---|
| 010_preYobi | 4 | 1 |
| 020_yobi | 5 | 3 |
| 030_add_sep | 31 | 3 |
| 041_ws1pre | 3 | 1 |
| 042_ws1after | 3 | 1 |
| 051_ws2pre | 3 | 1 |
| 052_ws2after | 3 | 1 |
| 060_add_feb | 11 | 3 |
| 070_ws3 | 3 | 1 |
| 070_ws3after | 3 | 1 |
| 080_po_ws | 3 | 1 |
| 総計 | 50 | 9 |

**表10**　問題ごとのPO経験者の人数

| 問題 | NP | P |
|---|---|---|
| 01_fre_ma | 4 | 1 |
| 02_hotel | 4 | 1 |
| 03_ec | 4 | 1 |
| 04_fre_ma | 4 | 1 |
| 05_1_fre_ma | 5 | 3 |
| 06_2_hotel | 36 | 6 |
| 07_3_ec | 5 | 3 |
| 08_4_fre_ma | 5 | 3 |
| 09_5_fre_ma | 5 | 3 |
| 10_6_ec | 5 | 3 |
| 11_ec | 6 | 2 |
| 12_kyoto_3step | 11 | 3 |
| 13_ec_3step | 11 | 3 |
| 14_ec | 6 | 2 |

### 7.5.2　データ処理

0.5秒未満の操作間隔は除いて分析した。0.5秒未満を除くのは、システムの制限で、恐らくマウス等操作の仕方によって極めて短い時間間隔が記録されることがあるからである。

### 7.5.3 検証 1 解答に要した時間

解答に要した時間に、PO 経験者と未経験者とで有意な差はみとめられなかった。**図 65** は解答に要した時間の平均値、中央値および標準偏差を示す。PO 経験者(P)の方が平均も中央値も短く、ばらつきも小さいように見える。**図 66** は解答に要した時間(ミリ秒)の自然対数値のヒストグラムである。図より、対数正規分布に従うと考えられる。PO 経験者と未経験者の解答に要した時間の自然対数値についてウィルコクソンの順位和検定すると 有意な差は認められなかった(p 値=0.094 > 0.05)。

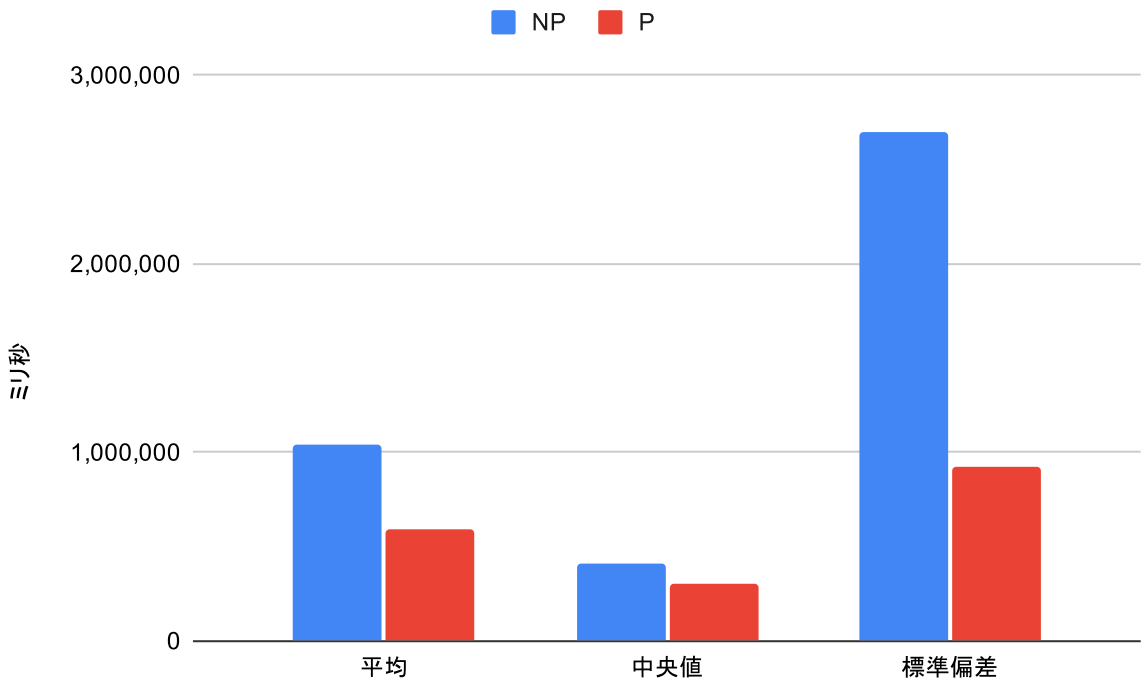



**図 65** PO 経験者(P)と未経験者(NP)の解答に要した時間の平均値、中央値、標準偏差

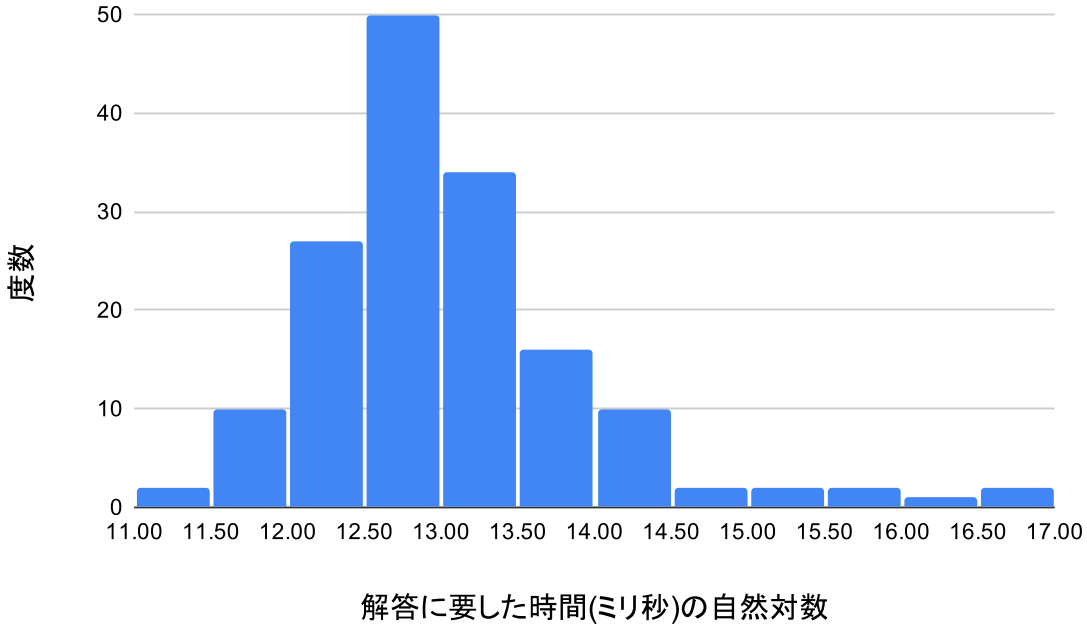



**図66**　解答に要した時間(ミリ秒)の自然対数値のヒストグラム

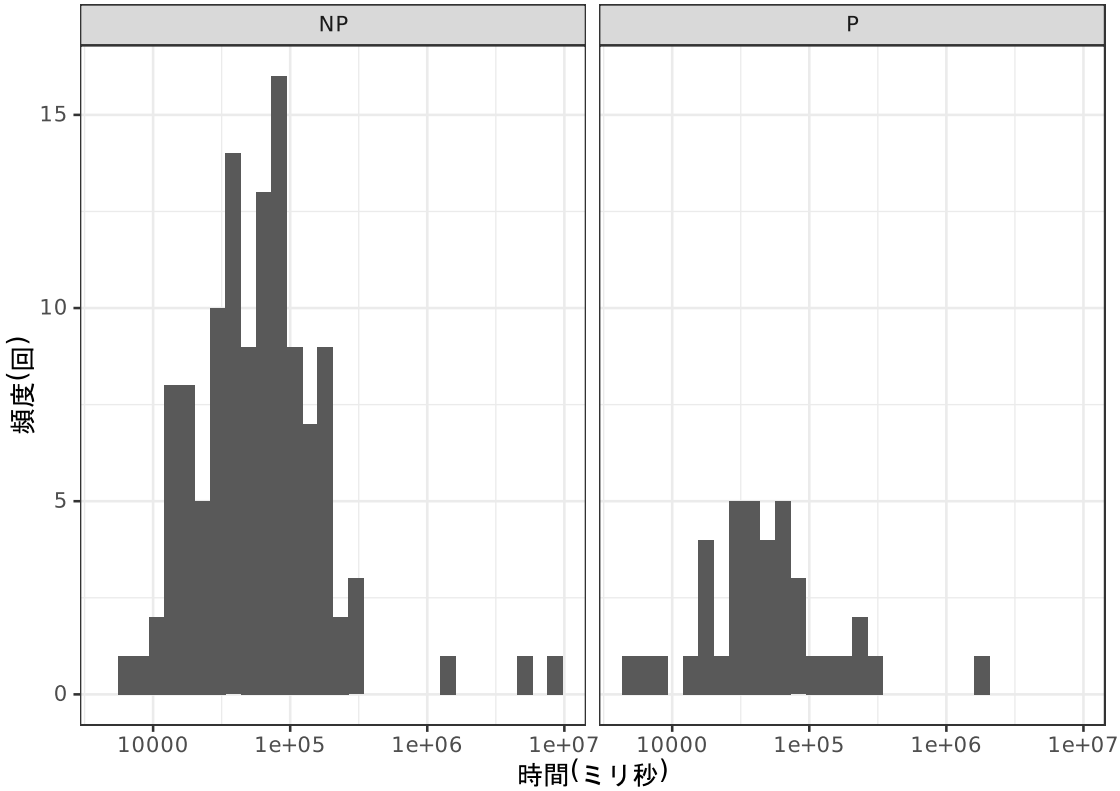



**図67**　PO経験者(左)と未経験者(右)の、最初のピースを動かすまでの時間の分布、横軸は自然対数

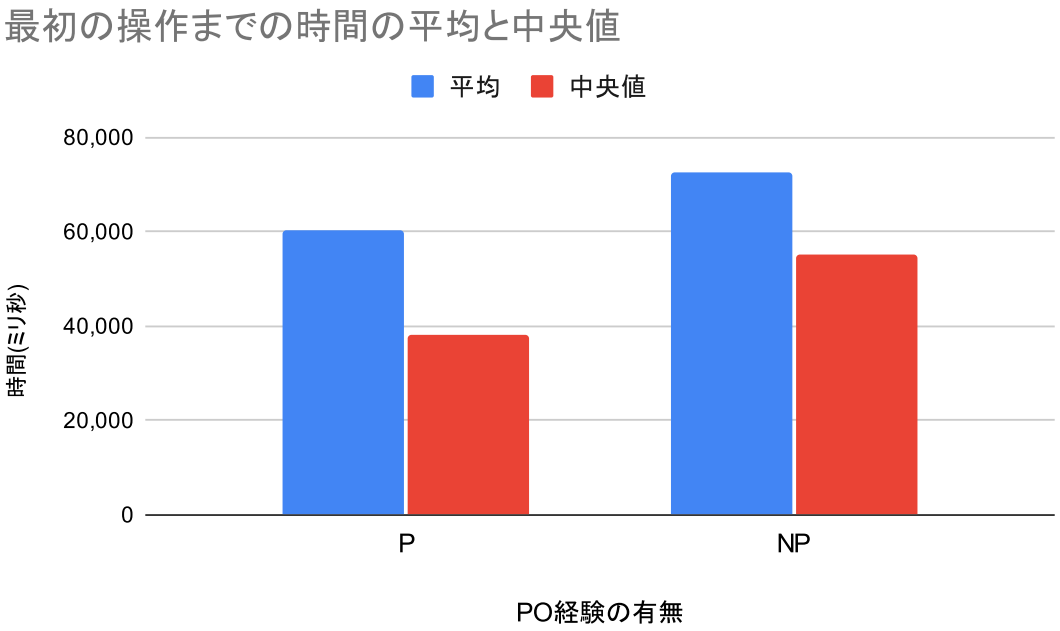



**図68**　PO経験者(左)と未経験者(右)の、最初のピースを動かすまでの時間の平均と中央値

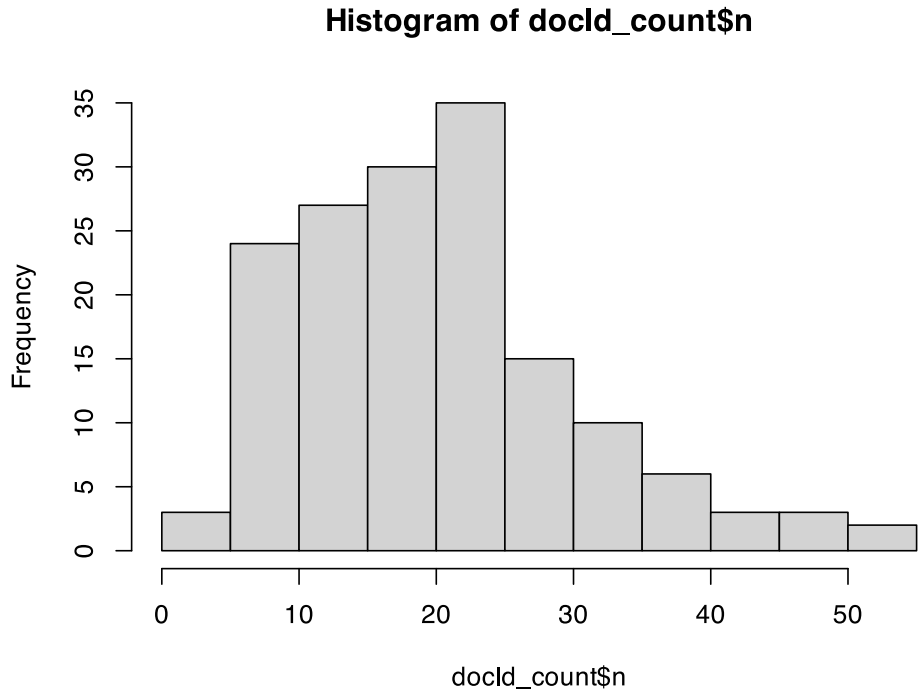



**図69**　PO経験者および未経験者の操作回数のヒストグラム、横軸は真数

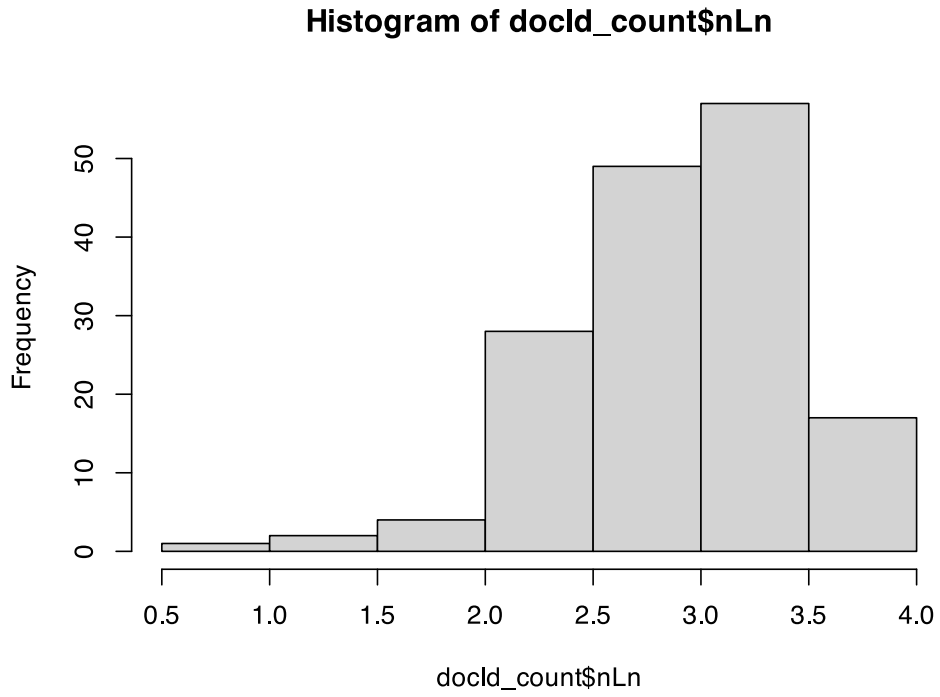



**図70**　PO経験者および未経験者の操作回数のヒストグラム、横軸は自然対数

#### 7.5.4　検証2 最初のピースを動かすまでの時間

最初のピースを動かすまでの時間に、PO経験者と未経験者とで有意な差は認められなかった。最初のピースを動かすまでの時間も対数正規分布に従うと考えられた(**図67**)。**図68**を見ると、PO経験者(左)の方が平均も中央値も小さいように見える。しかし、自然対数値についてウィルコクソンの順位和検定すると有意な差は認められなかった(p値=0.2238 > 0.05)。

#### 7.5.5　検証3 操作回数

**図69**および**図70**は、PO経験者および未経験者の操作回数の分布を示す。図69の横軸は真数、図70の横軸は自然対数である。真数および対数の両方とも正規分布に近いとは思えない。ウィルコクソンの順位和検定すると有意な差は認められず(p値=0.2999 > 0.05)、解くまでの操作回数に、PO経験者と未経験者とで有意な差は認められなかった。

#### 7.5.6　検証4 操作間隔

解くときの操作間隔に、PO経験者と未経験者とで有意な差が認められた。まず操作間隔のデータを概観した後で差を検証する。

**操作間隔の分布の概観**

**図71**は全体の操作間隔のヒストグラムで、縦軸は度数、横軸は操作間隔のミリ秒で対数目盛りである。操作間隔は対数正規分布に従うことが分かる。

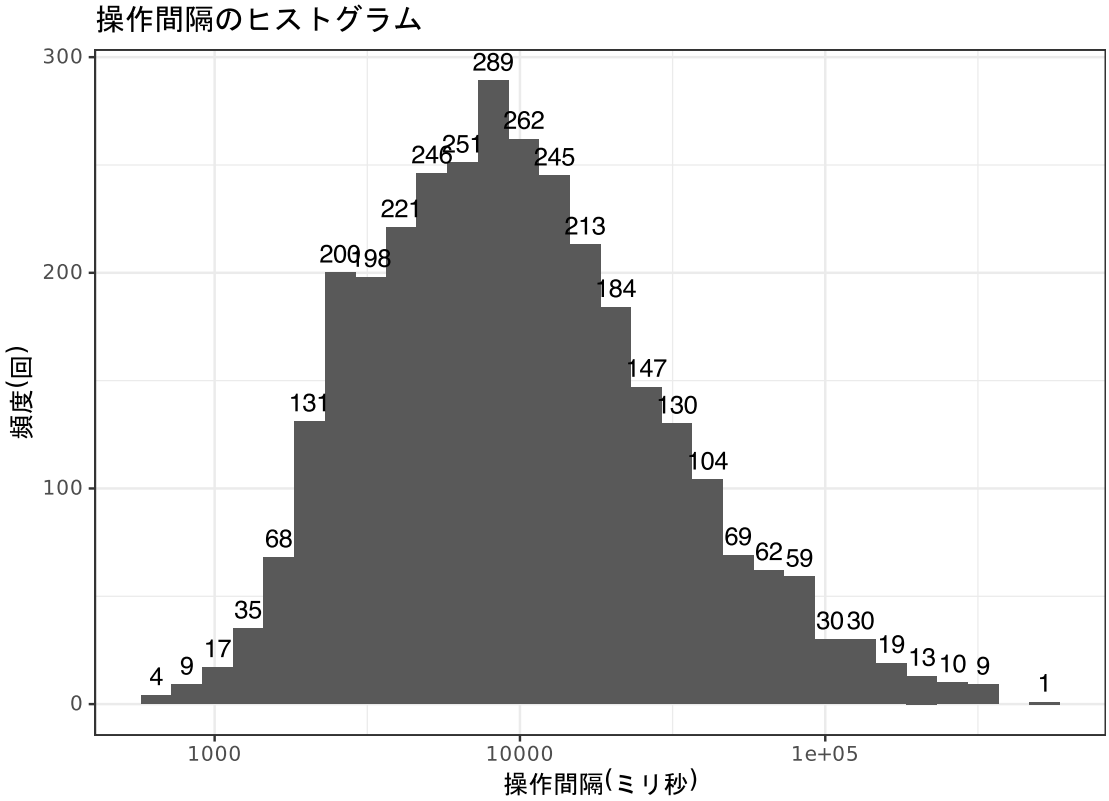



**図71**　全問題に対する操作間隔のヒストグラム

**図72**は予備実験(020_yobi)の6問(表7)における操作間隔の、被験者ごとのヒストグラムである。縦軸が度数、横軸は操作間隔の時間をミリ秒で示す。横軸は対数である。AからHはそれぞれ被験者で、(P)がPO経験者、(NP)が未経験者である。このセッションでは、AからCのPO経験者は未経験者に比べて分布の幅が狭く、操作間隔のバラツキが小さい、言い換えると並べ替え操作が安定していることが分かる。

以下で、全セッションの全問題について結果を示す。

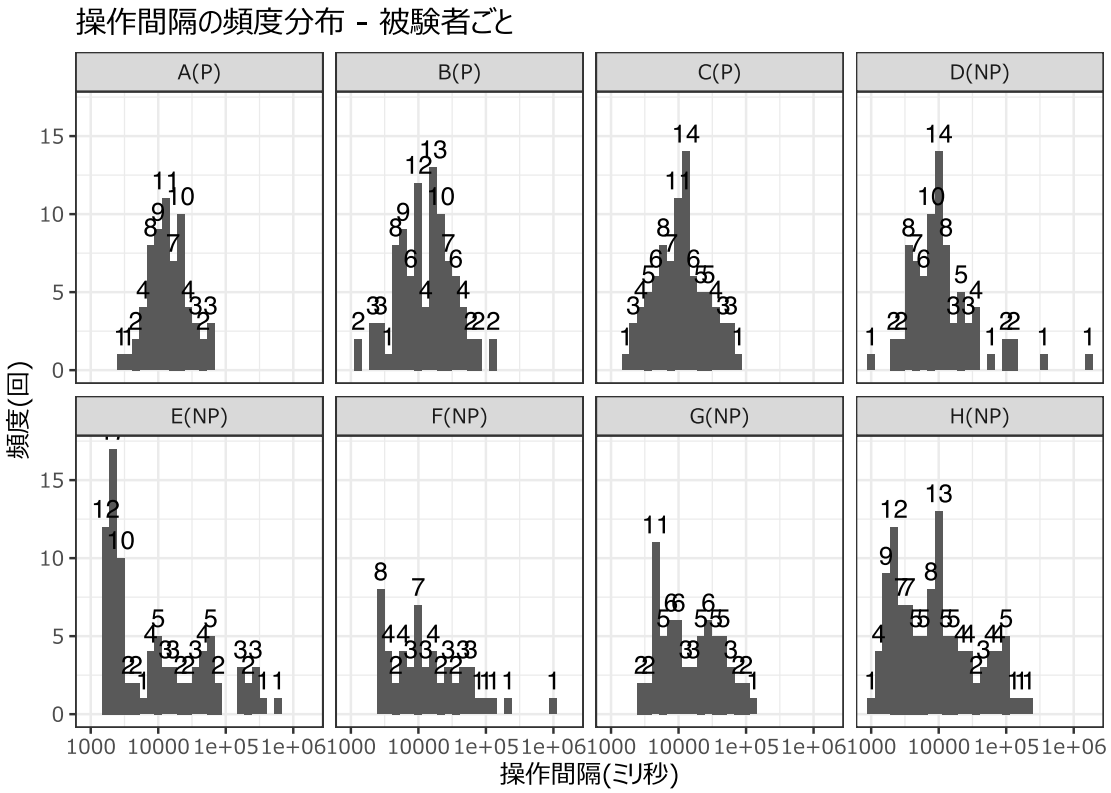



**図72**　予備実験(020_yobi)の6問に対する操作間隔の、被験者ごとのヒストグラム

**PO経験の有無による操作間隔の差**

**図73**にPO経験の有無による操作間隔の平均値と中央値の違いを示す。**図74**には操作間隔の標準偏差の違いを示す。

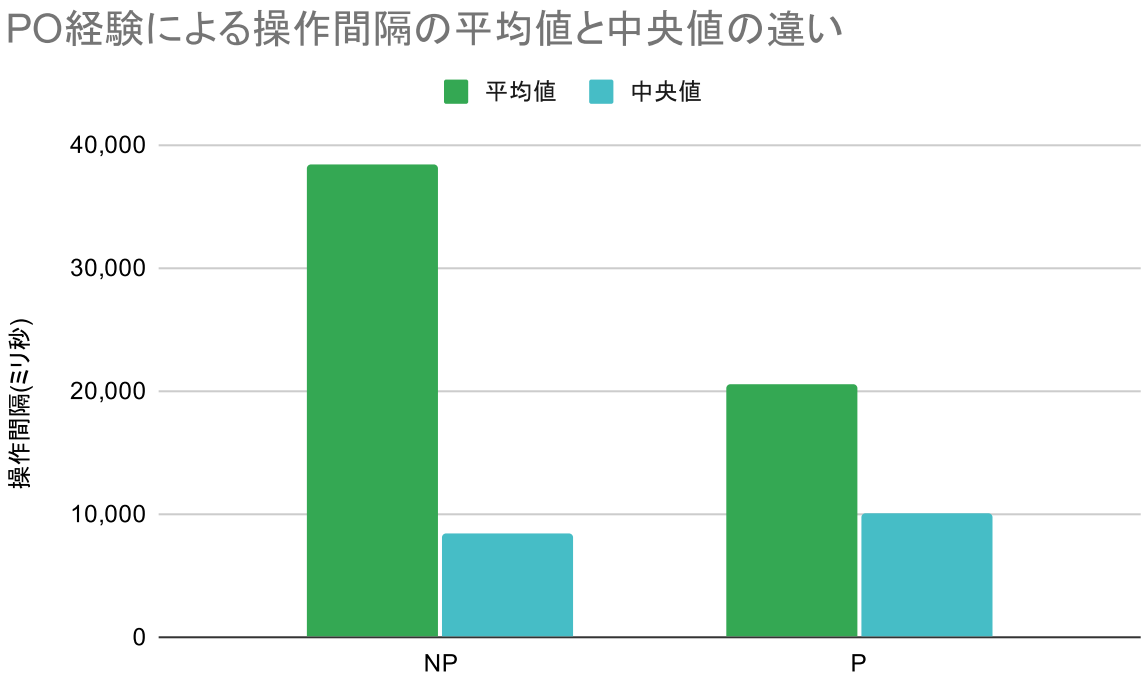



**図73**　PO経験の有無による操作間隔の平均値と中央値の違い。PがPO経験者、NPが未経験者

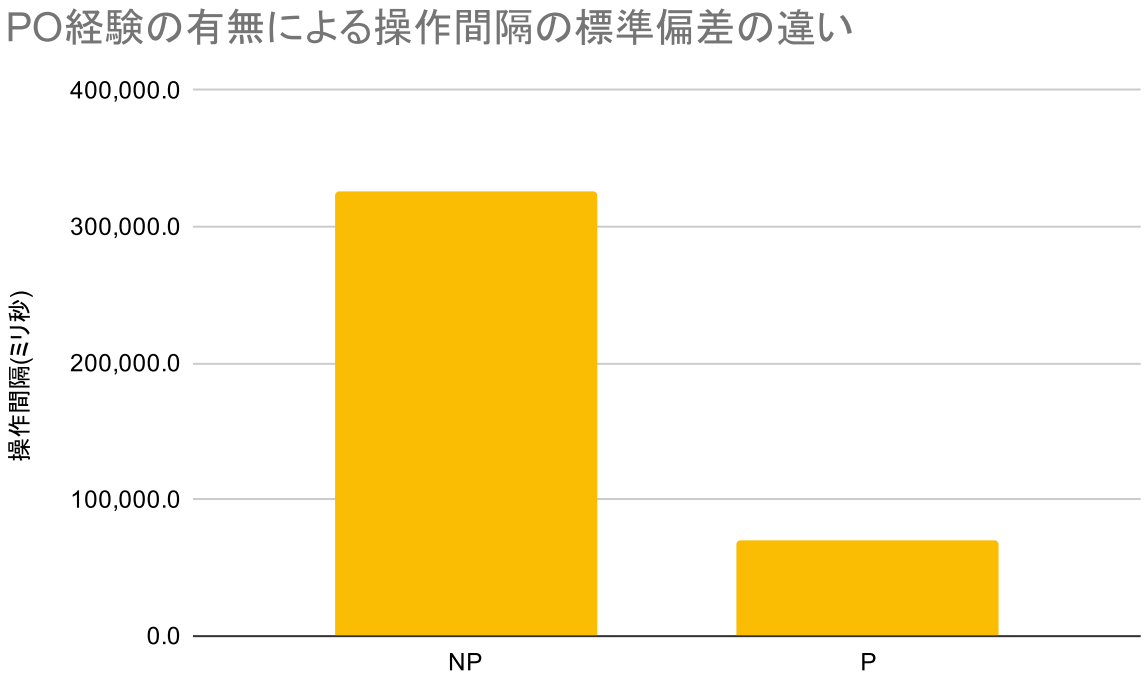



**図74**　PO経験の有無による操作間隔の標準偏差の違い

図73および図74では、中央値よりも平均が大きい傾向が、PO未経験者(NP)で顕著である。さらに、PO未経験者の方が標準偏差が大きく、操作間隔がばらついていると考えられる。

PO経験の有無で差があるか、操作間隔(ミリ秒)の自然対数値についてウィルコクソンの順位和検定したところ、有意な差が認められた(p値 0.00072 < 0.05)。

## 7.6 考察

以上の結果を詳細に検討する。

### 7.6.1 10分を超える操作間隔

「7.5.2で0.5秒未満の操作間隔は除いて分析したと述べた。その一方で、10分より大きいような操作間隔があったが、除かずにそのまま分析した。図71のヒストグラムを見ると、10分程度の操作間隔を超えて滑らかに分布するように見えて、これらを除外する理由が見当たらないからである。そこで、現段階では、これらを含めて分析する。

とはいえ、ジグソー・テキストの「オレオレ詐欺」のようなパズル問題を解くプロセスでは、操作間隔が10分を超えることは考えづらい。勤務時間中に自由に時間を取って解いたため、その間に他のことを行った可能性があり、実際に、途中で他の業務を行った場合があったという報告がある。他のことを行った場合を除外するならば、では、何分以上を除外するのか？そもそも、途中で他のことに気を取られる場合があることも含めて、普通の人の問題解決プロセスではないのか？このように考えても、やはり、10分を超えるデータを除外する根拠とならない。

### 7.6.2 問題ごとの操作間隔

検証4について、操作間隔の傾向を別角度から概観する。

全体に、ある程度考えながら解いているといえるだろうか？**図75**に操作間隔のパズル問題ごとの平均値を、**図76**に問題ごとの中央値を、**図77**に標準偏差を示す。

縦軸はミリ秒で、一番下の目盛りは10,000ミリ秒=10秒である。グラフから、一定の時間を考えながら操作していたことが分かる。

また、これらが対数正規分布に従うことを示したが、グラフからも、中央値よりも平均がかなり大きいことが分かる。これは、ときどき考え込むことで操作間隔が長くなるためと考えられる。

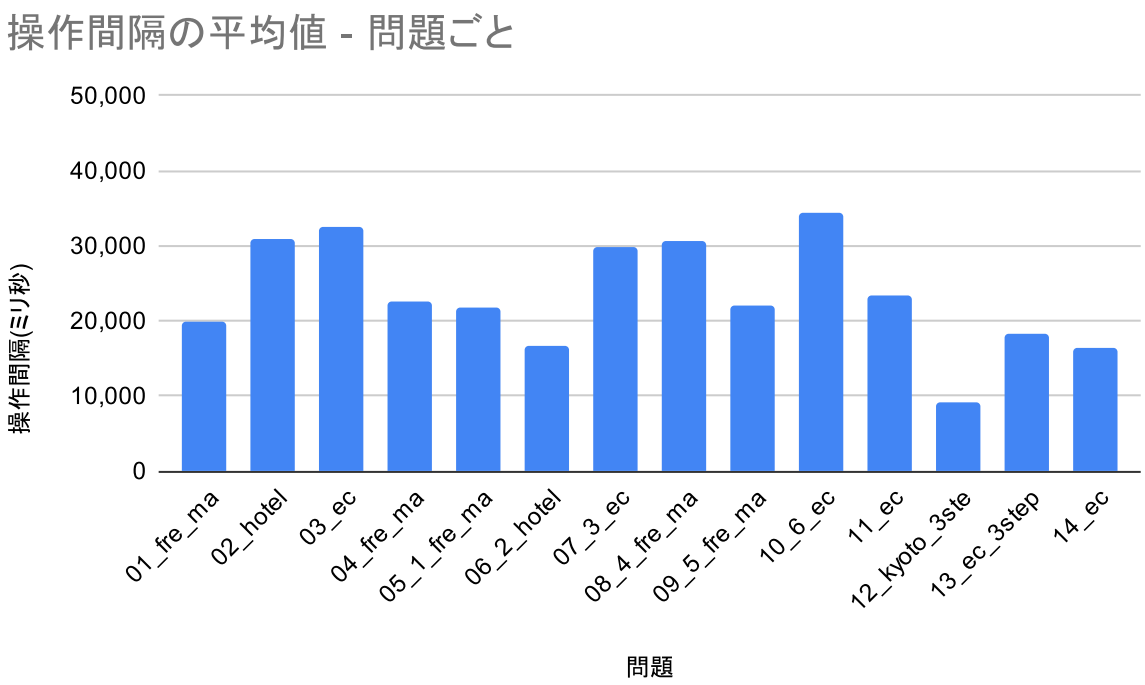



**図 75**　問題ごとの操作間隔の平均値

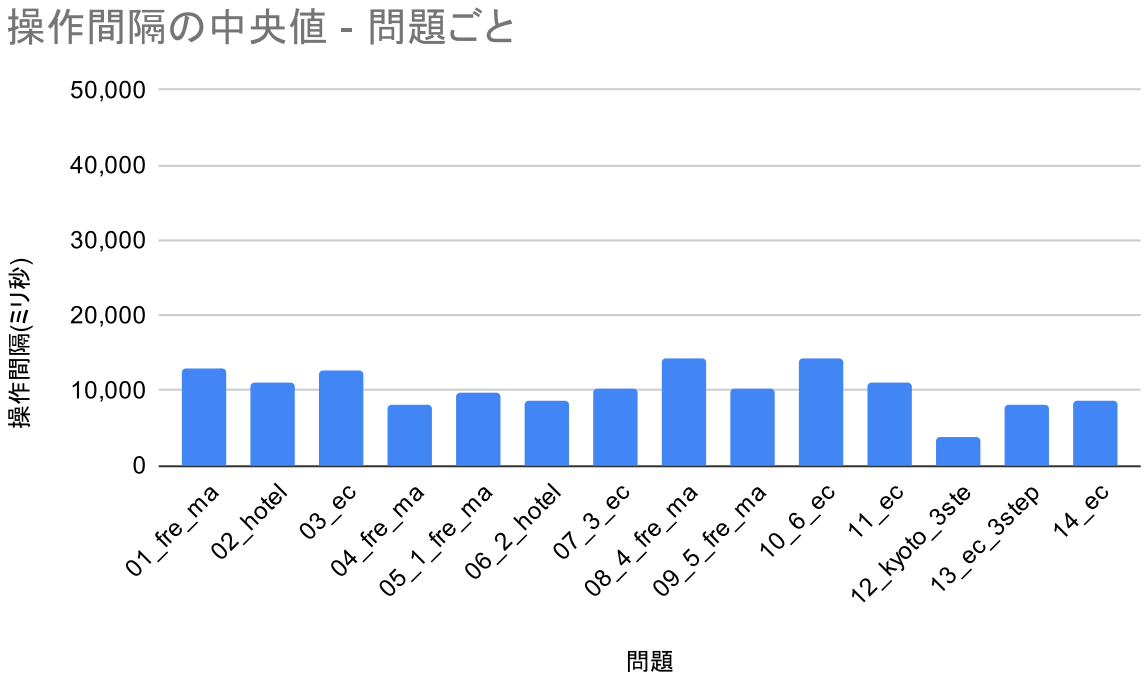



**図 76**　問題ごとの操作間隔の中央値

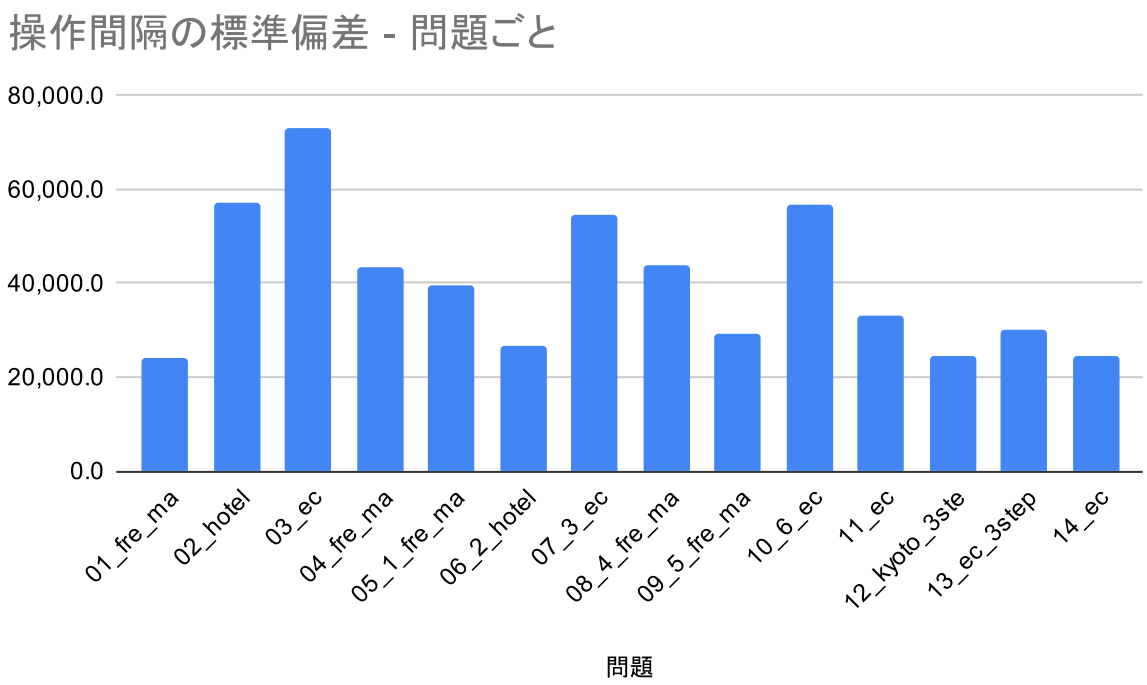



**図77**　問題ごとの操作間隔の標準偏差

### 7.6.3　難易度の小さな問題 - 旅行の準備 複数ステップ

考えながら解いているか、別の角度から検討する。

「旅行の準備 複数ステップ」(kyoto_3step)は平均値と中央値が最も小さい(図75、図76)。小さいといっても、中央値で3.9秒あり、闇雲に操作したのではなく、ピースの内容は読んでいたと考えられる。この問題は「アプリの操作になれるため」と告げて解いてもらったもので、ピースの内容は「清水寺」や「産寧坂」といった京都の観光地で、これらをまわる順位を問うものであった。操作練習として提示したが、平均値・中央値が最小なことから、その通りに扱ってもらえたといえる。深く考えずに解いた場合の操作間隔の目安になると考えられる。また、問題の難易度を考えた場合、「旅行の準備 複数ステップ」は最も優しい問題といってよいのではないか。セッション共通の問題がないと述べたが、「操作に慣れるため」の操作練習として、最初にこのような問題を出すことは可能であろう。今後の課題である。

### 7.6.4　特定の被験者が平均を引き上げていないか

操作間隔の平均値を、PO経験なしの特定の少数の人が引き上げているのではないか？

**表11**に、操作間隔が3分を越える操作の回数を、人ごとに集計した。全55人のうち12人に3分を超える操作があり、特定の少数の人が平均を引き上げてるとは言えない。PO経験者でも操作間隔が長いときがある。一方で、操作間隔の中央値、平均値は数秒から50秒未満なので、これらの操作では特に考え込んでいると推測される。

**表11**　操作間隔が3分より大きい操作の回数、被験者ごと。NP: PO未経験者、P: PO経験者

| 被験者 | セッション | NP | P |
|---|---|---|---|
| D065 | 010_preYobi | 4 | |
| | 020_yobi | 10 | |
| D064 | 010_preYobi | 4 | |
| | 020_yobi | 3 | |
| | 060_add_feb | 2 | |
| D068 | 010_preYobi | 4 | |
| | 020_yobi | 3 | |
| D066 | 020_yobi | 4 | |
| D071 | 060_add_feb | 3 | |
| D072 | 060_add_feb | | 2 |
| D010 | 060_add_feb | | 2 |
| D080 | 070_ws3after | 1 | |
| D042 | 052_ws2after | | 1 |
| D041 | 051_ws2pre | 1 | |
| D033 | 030_add_sep | 1 | |
| D004 | 030_add_sep | 1 | |

ちなみに、10分を超える操作間隔は除いて分析したとのべた。**表12**に操作間隔が10分を超えた操作の回数を示す。

**表12**　操作間隔が10分より大きい操作の回数、被験者ごと。NP: PO未経験者、P: PO経験者

| 被験者 | セッション | NP | P |
|---|---|---|---|
| D064 | 010_preYobi | 2 | |
| | 020_yobi | 1 | |
| | 060_add_feb | 1 | |
| D068 | 010_preYobi | 3 | |
| D071 | 060_add_feb | 2 | |
| D066 | 020_yobi | 2 | |
| D065 | 020_yobi | 1 | |
| D010 | 060_add_feb | | 1 |

### 7.6.5 10分を超える操作間隔

10分より大きいような操作間隔があったが、除かずにそのまま分析した。図71のヒストグラムを見ると、10分程度の操作間隔を超えて滑らかに分布するように見えて、これらを除外する理由が見当たらないからである。そこで、現段階では、これらを含めて分析する。

とはいえ、ジグソー・テキストの「オレオレ詐欺」のようなパズル問題を解くプロセスでは、操作間隔が10分を超えることは考えづらい。勤務時間中に自由に時間を取って解いたため、その間に他のことを行った可能性があり、実際に、途中で他の業務を行った場合があったという報告がある。他のことを行った場合を除外するならば、では、何分以上を除外するのか？そもそも、途中で他のことに気を取られる場合があることも含めて、普通の人の問題解決プロセスではないのか？このように考えても、やはり、10分を超えるデータを除外する根拠とならない。

### 7.6.6 答えのない課題

ワークショップでは、PO経験者から「プロダクト・バックログの順位に正解はない」といった発言があった。冒頭で述べた「プロダクト・バックログに正解はない」という認識が、被験者と共有されていたことを示すと考えられる。

## 7.7 結論

PO経験者と未経験者では、プロダクト・バックログの並べ替え操作について、1 解答に要した時間、2 最初のピースを動かすまでの時間、3 解くまでの操作回数に有意な差は認められなかった。その一方で、4 操作間隔に有意な差が認められ、PO経験者の方が操作間隔の平均値が小さくバラツキが小さかった。

これら結果から、プロダクト・バックログの順位付けにおいて、PO経験者は未経験者に比べて、1 あまり時間をかけずに並び替える、2 取り掛かるまでのスピードが速い、3 変更することへのハードルが少ないとは認められないが、4 決断が速くて迷いが少ないという認知的な傾向があると考えられる。

分析の過程で、解答に要した時間、最初のピースを動かすまでの時間および操作間隔の時間は対数正規分布に従うことが分かった。

また、この結論から、並べ替え操作の操作間隔には操作者(パズルのプレイヤー)の認知的な傾向が現れることが分かった。

本研究が明らかにした「並べ替えに時間をかけない」というPO経験者の特徴は、プロダクト・バックログの並べ替えに関するものである。他の題材において同様とは限らず、「すべての判断において時間をかけない」と示すものでもない。また、そのような傾向の良し悪しは、本研究の範囲外である。

本章の成果は試行錯誤といった思考プロセスの内容を明らかにするものではない。しかし、操作間隔におけるPO経験者と未経験者の違いは納得できるものである。提案アプリケーションの

測定データが思考プロセスの特徴を捉えるものであることを、別の側面から確かめたといえよう。

## 7.8 おわりに

スクラムでのプロダクト・バックログの順位付けにおいては、プロダクト・バックログはリファインメント(追加・変更・削除)があることが前提となる。このため、PO経験者は、プロダクト・バックログの順位付けにあたって、あまり時間をかけず、取り掛かるまでのスピードが速く、変更することへのハードルが低く、決断が速くて迷いが少ないという特徴があると予想した。これを確認するため、プロダクト・バックログ並べ替えの実験やワークショップを開催し、並べ替え操作を観察した。分析の結果、PO経験者は未経験者に比べて、操作間隔の平均が有意に小さく、個々の決断が速くて迷いが少ない傾向が認められた。

### 7.8.1 今後の課題など

本章では順位付けプロセスの操作間隔について、PO経験の有無による違いを分析した。他の観点の分析は今後の課題とする。例を挙げると、順位付けプロセスで最初に着目するピースの傾向、ワークショップ前後でのPO未経験者の変化、PO経験者に絞って経験したプロジェクト数などによる傾向、順位に正解はないが順位付け結果の傾向、などが考えられる。

本研究は人材育成に位置づけて行われ、教育研究の分野で報告した。本研究はメンバー／従業員の判断の仕方に関するものなので、経営や品質管理への認知的アプローチとしての発展も考えられる**[70]**。

### 7.8.2 本研究の意義

本研究は時間管理(time management)に関するものである。従来の時間管理や生活時間(time use)が数時間から数十日のスケールであった。これに対し、本研究は、パズル問題を解く全体で数分から十数分の期間における数秒単位の時間の使い方に認知的な特徴があることを明らかにした。

本研究は人材育成に位置づけられるものである。企業においてアジャイル的な仕事の進め方ができる人材へのニーズがあることを、本章の冒頭で述べた。本研究が明らかにしたPO経験者の特徴は、その意味で優れた人材の特徴と言えるだろう(ただし、この特徴そのものの良し悪しは本研究の範囲外である)。その一方で、従来の教育や研究は、ウォーターフォール的な作業の時間管理を求めるものであった。今後の学校教育でアジャイル的な進め方を教えるべきか？という課題を、本研究は提示するものである。

# 8 距離による分析

ここではプロセスの分析に正解との距離を導入することを論じる。正解との間の何らかの距離を適用して、解くプロセスの時間経過に伴う距離の変化を分析し、距離が大きくなるときにはプレイヤーが迷っていると判定する研究がある。このような判定が妥当かどうか、本研究では検証した。クイックソートなどのアルゴリズムでコンピューターに数を整列させ、途中の数の並び順を記録して、レーベンシュタイン距離など様々な距離の変化を分析した。その結果、整列のプロセスで、必ずしも距離が単調非増加せず、距離が広がることがあるし、正解に達したにもかかわらず再び誤った順序に変わることすらあることが分かった。整列アルゴリズムは、同じルールの判定と並べ替えを繰り返しているだけであり、途中で悩んでいるみなすのは妥当ではない。ただし、検証した範囲では、単調非増加する距離も存在した。

学習者の演習など知的プロセスを測定・分析・評価する研究では、「適切なプロセスでは、プロセスの進展に伴って正解との距離が単調に減少する」ことを前提とする場合がある。この前提は妥当だろうか?われわれは、コンピューターによる整列を題材に、これを確かめた。1から10の10個の数をクイックソートなどの整列アルゴリズムでコンピューターに整列させ、途中の数列を記録した。そして、途中の数列と正解の数列との距離を計算して変化を調べた。距離としてはレーベンシュタイン距離やケイリー距離などを採用した。コンピューターによる整列では、プロセスの途中で正解との距離が遠ざかることがあるし、距離0すなわち正解に達しても、さらにプロセスが続いて遠ざかることがあると分かった。単調減少の前提は、無条件に妥当ではないと考えられる[40]。

## 8.1 背景

特にプログラミング教育において、学習者の演習プロセスを測定・分析して、学習者のつまづきや理解を測ろうという研究が行われるようになってきた[48]**[71]**[49][37]**[72]**。

人の知的プロセスを測定・分析・評価するこのような研究では、「適切なプロセスでは、プロセスの進展に伴って正解との距離が単調に減少する」ことを前提とする場合がある。ここで「単調に減少する」とは広い意味で、$x < y$のときに$f(x) \leqq f(y)$であること、言い換えると単調非増加することを指して、あるステップで距離が変わらない場合も含むこととする。

プロセス(過程)を対象とする学習分析では、プロセスをデータ化/測定する手法や測定データを分析する手法などが提案される。人の考えを直接見ることはできないが、分析に基づく推定と正答率との相関を調べたり、発話プロトコルと照合したりすることで提案手法が評価される(**図79**)。編集距離を使う研究では、この測定・分析で、プロセスの各ステップでの正解との距離を計算し、それら距離の変化を分析して学習者の状態を推定する。

## 8.2 目的

人が知的な処理を行うとき、「適切なプロセスでは、プロセスの進展に伴って正解との編集距離が単調に減少する」という前提は妥当だろうか?。本研究の目的は、コンピューターによる整列を題材に、人に対するこの前提の妥当性を検証することである。図79での学習者、すなわち人をコンピューターにおきかえて、コンピューターが数を整列するプロセス(過程)を測定し、正解すなわち整列済み数列(数の並び)との距離を計算して単調に減少するかを検証する(**図78**)。

整列を題材にするのは、プログラミングよりも単純なプロセスだからである。また、コンピューターによる整列を調べるのは、中で何をやっているかアルゴリズムが明白で、かつ、デバッガを使ったりコードを書き換えることでプロセスの途中の状態を詳細に調べられるからである。

人の知的プロセスに関する前提を検証するため、コンピューターによる処理を測定・分析するにあたって、条件を設定する。これについては8.4節で述べる。

人による整列とコンピューターによる整列が同じなのか？については検証が必要であるが、それは別稿に譲り、本稿の範囲外とする。

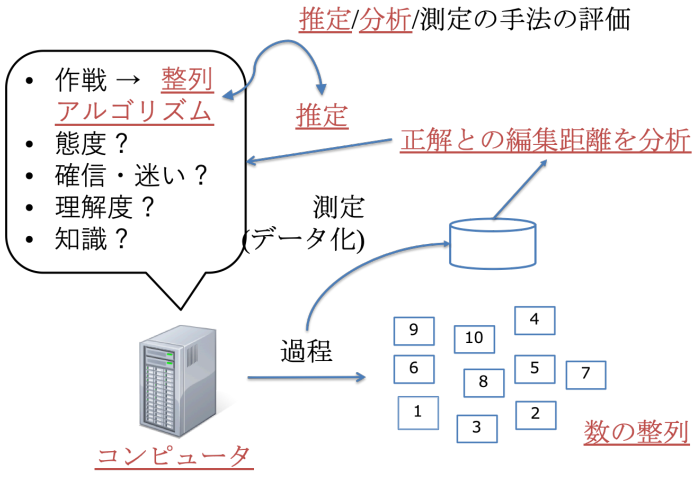



**図78**　コンピューターによる整列を題材に、正解との編集距離の変化を調査

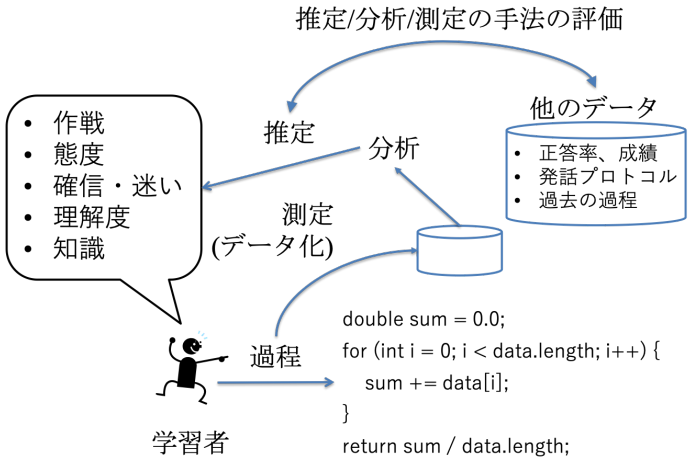



**図79**　学習分析の研究の構造

## 8.3　関連する研究

### 8.3.1　正解との距離に基づく分析

図79の測定・分析で編集距離を使う研究では、プロセスの各ステップでの正解との距離を計算し、それら距離の変化を調べて学習者の状態を推定する。「適切なプロセスでは成果との距離が単調に減少する」ことを前提にすると、途中で距離が増加する/広がる場合、学習者に何らかの問題が起きていると推定できる。例えば、被験者/学習者はTinkerしてる(下手にいじくり回してる)と推定することがある(**図80**)。

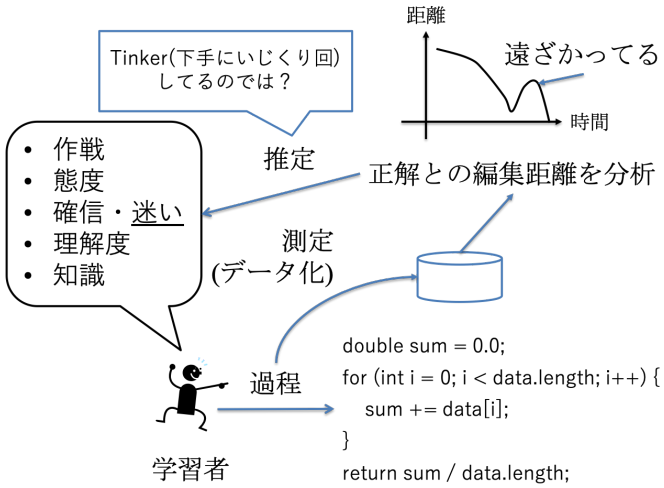



**図80**　正解から遠ざかることに基づく推定

### 8.3.2　コンピューターによる整列

データを一定のルールに従って並べ替えることを整列、またはソート(sort)といい**[73]**、単純だが知的なプロセスである。その研究は古く、コンピューターによる整列はコンピューターの歴史と共にあると言ってよい。

コンピューターによる整列は、次のような論点で研究されてきた:

- 結果が正しいのは当たり前
- 性能、ステップ数、処理時間
- メモリ所要量
- 安定性。同じ順序だった複数のデータの順序が、整列の前後で変わるかどうか。
- データ操作の手法: データの位置の替え方、交換、挿入、選択、マージ、など
- データ操作の作業領域: 数列の内部か、外部か

本稿あるいは学習分析の観点からは、コンピューターによる整列は次の特徴を持つ:

- 必ず正解する。プロセスだけが異なる。正答率の変化で評価するとか正解を教師として機械学習するといった、学習分析でよく採用される「結果に基づく分析」手法を使えない。
- アルゴリズムが明らかである。本人にインタビューして「何考えてましたか？」と聞くまでもない。(何より自分で組んだアルゴリズムで動いているのだから。)
- 再現できる。人間による整列よりもコンピューターによる整列の方が再現性が高い。
- 網羅性が高い。(人間に並べ替え可能な)10個程度の数であれば、パソコンで全順列を網羅できる。

### 8.3.3 人による並べ替え操作の測定・分析

我々はこれまで、並べ替えプログラミング・パズルのジグソー・コードを開発して、学習者が並べ替えプログラミングするとき、並べ替え操作を測定した。操作の傾向を分析して、学習者へアドバイスすべき内容を明らかにした[48][49]。

ジグソー・コードは、PCやスマホのUIで広く採用されているドラッグ&ドロップ操作でプログラムの断片(ピース)を並べ替えて完成させる。開始、ドラッグ開始、ドロップ、完成ボタンというタイミングで、ピース全体の並び順や時刻などを記録する。ドラッグやドロップでは、操作対象のピースも記録する。

## 8.4 アプローチ

単調減少の前提の妥当性を、コンピューターによる数の整列を題材に検証する。このアプローチは、コンピューターに整列させる問題の作問と正解、検証の対象とする整列アルゴリズム、プロセスの途中で正解との距離を測る測定タイミング、採用する距離、距離の変化の評価方法で構成される。

### 8.4.1 作問と正解

1から10の10個の数を昇順に並べ替えることを問題とする。全順列${}_{10}P_{10} = 3,628,800$個の問題をコンピューターに解かせる。

これら問題の正解は(1,2,3,4,5,6,7,8,9,10)である。問題の中には、最初から正解で並べ替え不要のものも含まれる。

### 8.4.2 整列アルゴリズム

整列アルゴリズムとして、バブルソート、クイックソート、ヒープソート、マージソート、選択ソート、挿入ソートの6個を検証する。

挿入ソートはドラッグ&ドロップと同じ操作でデータを操作する。ジグソー・コードの並べ替え操作と同じなので、将来の参考のために検証対象とする。

コンピューターに実行させる整列プログラムは、文献[73]のアルゴリズムをPythonで実装する。

### 8.4.3 測定タイミング

並べ替えの進展に伴う正解との距離の変化を調べるために、タイミングを決めて、その瞬間の数列のスナップショットを記録する。そのタイミングでは、数列に欠けている数があってはならないし、数が重複していてもならない。これには、8.3.2で述べた、整列アルゴリズムのデータ操作の手法やデータ操作の作業領域が関係する。

バブルソート、ヒープソート、クイックソート、選択ソートは、処理の途中で数を交換(入れ替え、swap)するが、その交換後に数列を記録する。

マージソートは、整列済の数列を1つの数列にマージするが、このマージの直後に数列を記録する。

挿入ソートは、数を挿入した直後に数列を記録する。

### 8.4.4 距離

学習分析の研究では、学習アプリケーションで可能な操作に応じて、編集距離を定義する場合がある[71]。整列問題はデータ操作を限定しない。そのうえで、前節で述べたとおり、整列アルゴリズムは、置換、マージ、挿入といった様々な独自の規則でデータを操作する。人の知的プロセスに関する研究では、現状、これらアルゴリズムに相当する考え方を突き止めることが目標の1つであり、事前にアルゴリズムを知って距離を決めることはできない。本章では、アルゴリズムごとに距離を限定することをせず、いくつかの距離を採用して全てのアルゴリズムに適用して分析する。

距離としてはレーベンシュタイン距離(Levenshtein distance)、ハミング距離(Hamming distance)、ケイリー距離(Cayley distance)およびウラム距離(Ulam distance)の4個を採用する。レーベンシュタイン距離以外は、同じ文字数の文字などに適用される距離である。

レーベンシュタイン距離は、1要素の挿入・削除・置換(その要素を任意の別の要素に置き換える)操作を何回繰り返すと、一方の並びから他方の並びへ変わるかの操作回数である。断りなく編集距離というとレーベンシュタイン距離を指すことが多い。

ハミング距離は、2つの並びの同じ位置にあって異なる要素の個数である。(2,5,3,1,4)と(1,2,3,4,5)の距離は、3だけが同じで他は異なるので、4である。

**表13** 正解との距離の変化

| | Levenshtein距離 | | Hamming距離 | | Caley距離 | | Ulam距離 | |
|---|---|---|---|---|---|---|---|---|
| | 増加した問題数 | 割合 | 増加した問題数 | 割合 | 増加した問題数 | 割合 | 増加した問題数 | 割合 |
| バブルソート | 2,743,264 | 75.6% | 3,512,825 | 96.8% | 3,624,619 | 99.9% | 0 | 0.0% |
| ヒープソート | 3,628,800 | 100.0% | 3,628,800 | 100.0% | 3,628,800 | 100.0% | 3,628,800 | 100.0% |
| 挿入ソート | 1,339,176 | 36.9% | 1,964,215 | 54.1% | 3,443,878 | 94.9% | 0 | 0.0% |
| マージソート | 1,852,755 | 51.1% | 2,432,789 | 67.0% | 3,426,453 | 94.4% | 0 | 0.0% |
| クイックソート | 1,034,478 | 28.5% | 0 | 0.0% | 1,089,872 | 30.0% | 872,908 | 24.1% |
| 選択ソート | 0 | 0.0% | 0 | 0.0% | 0 | 0.0% | 0 | 0.0% |

ケイリー距離は「交換」操作によってのみ並びを変換して、一方の並びから他方の並びにいたるまでの変換の最小回数を距離とするものである**[74]**。この操作は、クイックソートなどが行う

交換に該当する。**図81**はケイリー距離の例である。destinationとstartとの距離を測っている。swap(1, 4)は交換操作を表し、1や4は各数の添字である。前の行に交換操作を行った結果がswap()の行に表示されている。swap(1, 4)は、添字1と添字4を交換、すなわち添字1=1、添字4=4を交換することを表す。交換操作で交換された数に下線が引かれている。startから3回の交換でdestinationと同じになったので、ケイリー距離は3である。

| | |
|---|---|
| destination | 2, 5, 3, 1, 4 |
| start | 1, 2, 3, 4, 5 |
| swap(1, 4) * | 4, 2, 3, 1, 5 |
| swap(1, 5) | 5, 2, 3, 1, 4 |
| swap(1, 2) | 2, 5, 3, 1, 4 |
| number of cycles | 3 |
| | (* swapの引数は添字) |

**図81**　ケイリー距離の例

ケイリー距離は、PythonならばSymPyライブラリのPermutations**[75]**を使って、Rであればper Mallowsパッケージ**[76]**を使って計算できる。SymPyは代数計算のライブラリである。

ウラム距離の操作は、スマホのUIなどのドラッグ&ドロップに該当する。ウラム距離は、目的の並びになるまでの最小の操作数である[74]。これは、挿入ソートが行う操作に該当する。ウラム距離は、並びの長さから最長共通部分(Longest Common Subsequence)を引いた値に等しい[74]ので、これをPythonで計算した。RであればPerMallowsパッケージ[76]を使って計算できる。

### 8.4.5　実験と距離の変化の評価

8.4.2の整列アルゴリズムを実装するプログラムに、8.4.3のタイミングで数列を出力するコードを挿入する。これらのプログラムで8.4.1の問題を整列させて、各タイミングでの数列を記録する。これら記録しておいた整列中の数列と正解との距離を、後で8.4.4の距離で測る。こうすることで、後から別の距離で分析しなおすこともできる。

こうして得られた距離の列の中で、前の距離よりも大きくなる場合があるかどうかを調べる。

## 8.5　結果

全ての整列アルゴリズムで、全ての距離について、正解との距離が遠ざかる場合があることが分かった(**表13**)。

ヒープソートは、全ての問題について、すべての距離で、処理のどこかで正解との距離が増加する。

選択ソートは、全ての問題について、すべての距離で、処理の途中で正解との距離が単調減少した。

### 8.6 考察

ヒープソートが不適切なアルゴリズムだという理由はない。正解との距離が単調減少しないからといって、ヒープソートが不適切だとは言えない。

選択ソートが他のアルゴリズムと異なって優秀だという理由はない。正解との距離が必ず単調減少するからといって、選択ソートが適切だとは言えない。

距離が遠ざかるとき、コンピューターは試行錯誤してるのか？あるいはアルゴリズムは何か特別な処理をしてるのか？正解に近づく場合と遠ざかる場合とで、実行されるコードは同じである。仮に、それぞれの場合で実行されるコードが異なれば、近づく場合は順調、遠ざかる場合は不調といった解釈への道が拓けるかもしれない。しかし、そうではない。

```
swap-vl                              levenshtein    hamming        cayley         ulam
start    1, 2, 3, 4, 5, 7, 8,10, 6, 9   4 ####         5 #####        4 ####         2 ##
( 9,10)  1, 2, 3, 4, 5, 7, 8, 9, 6,10   2 ##           4 ####         3 ###          1 #
( 6, 8)  1, 2, 3, 4, 5, 7, 6, 9, 8,10   3 ###          4 ####         2 ##           2 ##
( 6, 7)  1, 2, 3, 4, 5, 6, 7, 9, 8,10   2 ##           2 ##           1 #            1 #
( 8, 9)  1, 2, 3, 4, 5, 6, 7, 8, 9,10   0              0              0              0
answer   1, 2, 3, 4, 5, 6, 7, 8, 9,10
```

**図82**　クイックソートの距離の変化

**図82**はクイックソートの途中で正解との距離が広がった例である。一番左の列の( 9,10)などは数列に対する置換操作で、その上の行の数列で9と10を入れ替えた結果、その行の数列になったことが分かる。( 9,10)の置換では距離が縮まったが、次の( 6, 8)の置換ではレーベンシュタイン距離が広がった。( 9,10)の置換にいたるまで処理(for文、while文といった繰り返しやif文の判定)も、(6, 8)の置換にいたる処理も同じプログラム・コードによるものである。

正解との距離が単調減少することを、良い整列アルゴリズムの条件とするならば、では距離を確かめながら処理することは妥当か？例えば、各整列アルゴリズムにおいて、置換などの並べ替えを行う前に「並べ替え後の正解との距離を調べて、遠ざかる場合は並べ替えしない」という処理をすれば、単調減少するだろうか。これは上手くいかない。なぜならば、正解が分からないからこそ整列するからである。正解を知っていて、それとの距離を確かめられるなら、そもそも整列不要である。従って、正解を知らずに正解との距離を知るアルゴリズムが必要となる。それを実装したとき、その整列アルゴリズムはもはや、元の整列アルゴリズムとは別物であろう。

```
swap-vl                                levenshtein    hamming        cayley         ulam
start    1, 2, 3, 4, 9, 7,10, 5, 6, 8  6 ######       6 ######       5 #####        3 ###
( 2, 1)  2, 1, 3, 4, 9, 7,10, 5, 6, 8  8 ########     8 ########     6 ######       4 ####
( 3, 2)  3, 1, 2, 4, 9, 7,10, 5, 6, 8  8 ########     9 #########    7 #######      4 ####
( 4, 1)  3, 4, 2, 1, 9, 7,10, 5, 6, 8  9 #########   10 ##########   8 ########     5 #####
( 4, 3)  4, 3, 2, 1, 9, 7,10, 5, 6, 8  9 #########   10 ##########   7 #######      6 ######
( 9, 3)  4, 9, 2, 1, 3, 7,10, 5, 6, 8  8 ########    10 ##########   8 ########     5 #####
( 9, 4)  9, 4, 2, 1, 3, 7,10, 5, 6, 8  8 ########    10 ##########   9 #########    5 #####
( 7, 2)  9, 4, 7, 1, 3, 2,10, 5, 6, 8  9 #########   10 ##########   8 ########     5 #####
(10, 7)  9, 4,10, 1, 3, 2, 7, 5, 6, 8  9 #########    9 #########    7 #######      5 #####
(10, 9) 10, 4, 9, 1, 3, 2, 7, 5, 6, 8  9 #########    9 #########    8 ########     5 #####
( 5, 1) 10, 4, 9, 5, 3, 2, 7, 1, 6, 8  9 #########    9 #########    7 #######      6 ######
( 5, 4) 10, 5, 9, 4, 3, 2, 7, 1, 6, 8  8 ########     8 ########     6 ######       7 #######
( 6, 4) 10, 5, 9, 6, 3, 2, 7, 1, 4, 8  9 #########    9 #########    7 #######      6 ######
( 6, 5) 10, 6, 9, 5, 3, 2, 7, 1, 4, 8  9 #########    9 #########    6 ######       7 #######
( 8, 3) 10, 6, 9, 5, 8, 2, 7, 1, 4, 3  9 #########    9 #########    7 #######      8 ########
( 8, 6) 10, 8, 9, 5, 6, 2, 7, 1, 4, 3  8 ########     9 #########    8 ########     7 #######
(10, 3)  3, 8, 9, 5, 6, 2, 7, 1, 4,10  7 #######      8 ########     7 #######      5 #####
( 9, 3)  9, 8, 3, 5, 6, 2, 7, 1, 4,10  6 ######       7 #######      6 ######       5 #####
( 7, 3)  9, 8, 7, 5, 6, 2, 3, 1, 4,10  8 ########     9 #########    7 #######      6 ######
( 9, 4)  4, 8, 7, 5, 6, 2, 3, 1, 9,10  7 #######      8 ########     6 ######       5 #####
( 8, 4)  8, 4, 7, 5, 6, 2, 3, 1, 9,10  7 #######      8 ########     5 #####        5 #####
( 6, 4)  8, 6, 7, 5, 4, 2, 3, 1, 9,10  8 ########     8 ########     4 ####         6 ######
( 8, 1)  1, 6, 7, 5, 4, 2, 3, 8, 9,10  6 ######       6 ######       3 ###          4 ####
( 7, 1)  7, 6, 1, 5, 4, 2, 3, 8, 9,10  7 #######      7 #######      4 ####         4 ####
( 3, 1)  7, 6, 3, 5, 4, 2, 1, 8, 9,10  6 ######       6 ######       3 ###          5 #####
( 7, 1)  1, 6, 3, 5, 4, 2, 7, 8, 9,10  4 ####         4 ####         2 ##           3 ###
( 6, 1)  6, 1, 3, 5, 4, 2, 7, 8, 9,10  5 #####        5 #####        3 ###          3 ###
( 5, 1)  6, 5, 3, 1, 4, 2, 7, 8, 9,10  5 #####        5 #####        4 ####         4 ####
( 6, 2)  2, 5, 3, 1, 4, 6, 7, 8, 9,10  4 ####         4 ####         3 ###          2 ##
( 5, 2)  5, 2, 3, 1, 4, 6, 7, 8, 9,10  3 ###          3 ###          2 ##           2 ##
( 4, 2)  5, 4, 3, 1, 2, 6, 7, 8, 9,10  4 ####         4 ####         3 ###          3 ###
( 5, 2)  2, 4, 3, 1, 5, 6, 7, 8, 9,10  3 ###          3 ###          2 ##           2 ##
( 4, 2)  4, 2, 3, 1, 5, 6, 7, 8, 9,10  2 ##           2 ##           1 #            2 ##
( 4, 1)  1, 2, 3, 4, 5, 6, 7, 8, 9,10  0              0              0              0
( 3, 1)  3, 2, 1, 4, 5, 6, 7, 8, 9,10  2 ##           2 ##           1 #            2 ##
( 3, 1)  1, 2, 3, 4, 5, 6, 7, 8, 9,10  0              0              0              0
( 2, 1)  2, 1, 3, 4, 5, 6, 7, 8, 9,10  2 ##           2 ##           1 #            1 #
( 2, 1)  1, 2, 3, 4, 5, 6, 7, 8, 9,10  0              0              0              0
answer   1, 2, 3, 4, 5, 6, 7, 8, 9,10
```

**図83**　ヒープソートの距離の変化

距離0、すなわち正解になった後で遠ざかっている例

**図83**では、距離が0、すなわち正解に達した後でも処理を続けている。ここでも、正解に達したかを確かめながら処理を進めれば、このような事態を避けられるが、それはできないのは同様である。

その他の距離の変化の例を示す。**図84**から**図89**で、一番左の列は置換等の操作を表す。例えば図88では、start行の数列に対し、3と8を交換した(3,8)結果がstartの1つ下の行である。

| 操作 | | levenshtein | hamming | cayley | ulam |
|---|---|---|---|---|---|
| start | 1, 2, 3, 4, 5, 7, 8,10, 6, 9 | 4 #### | 5 ##### | 4 #### | 2 ## |
| ( 9,10) | 1, 2, 3, 4, 5, 7, 8, 9, 6,10 | 2 ## | 4 #### | 3 ### | 1 # |
| ( 6, 8) | 1, 2, 3, 4, 5, 7, 6, 9, 8,10 | 3 ### | 4 #### | 2 ## | 2 ## |
| ( 6, 7) | 1, 2, 3, 4, 5, 6, 7, 9, 8,10 | 2 ## | 2 ## | 1 # | 1 # |
| ( 8, 9) | 1, 2, 3, 4, 5, 6, 7, 8, 9,10 | 0 | 0 | 0 | 0 |
| answer | 1, 2, 3, 4, 5, 6, 7, 8, 9,10 | | | | |

**図84**　クイックソートでの正解との距離の変化。途中で距離が増加している

| 操作 | | levenshtein | hamming | cayley | ulam |
|---|---|---|---|---|---|
| start | 1, 2, 3, 4, 5, 7, 9, 6,10, 8 | 4 #### | 5 ##### | 4 #### | 2 ## |
| ( 6, 7) | 1, 2, 3, 4, 5, 6, 9, 7,10, 8 | 4 #### | 4 #### | 3 ### | 2 ## |
| ( 7, 9) | 1, 2, 3, 4, 5, 6, 7, 9,10, 8 | 2 ## | 3 ### | 2 ## | 1 # |
| ( 8,10) | 1, 2, 3, 4, 5, 6, 7, 9, 8,10 | 2 ## | 2 ## | 1 # | 1 # |
| ( 8, 9) | 1, 2, 3, 4, 5, 6, 7, 8, 9,10 | 0 | 0 | 0 | 0 |
| answer | 1, 2, 3, 4, 5, 6, 7, 8, 9,10 | | | | |

**図85**　クイックソートでの正解との距離の変化。距離が単調に減少している

| 操作 | | levenshtein | hamming | cayley | ulam |
|---|---|---|---|---|---|
| start | 10, 1, 5, 8, 9, 2, 3, 4, 6, 7 | 8 ######## | 10 ########## | 8 ######## | 4 #### |
| ( 1,10) | 1,10, 5, 8, 9, 2, 3, 4, 6, 7 | 8 ######## | 9 ######### | 7 ####### | 4 #### |
| ( 2,10) | 1, 2, 5, 8, 9,10, 3, 4, 6, 7 | 8 ######## | 8 ######## | 6 ###### | 4 #### |
| ( 3, 5) | 1, 2, 3, 8, 9,10, 5, 4, 6, 7 | 7 ####### | 7 ####### | 5 ##### | 4 #### |
| ( 4, 8) | 1, 2, 3, 4, 9,10, 5, 8, 6, 7 | 5 ##### | 5 ##### | 4 #### | 3 ### |
| ( 5, 9) | 1, 2, 3, 4, 5,10, 9, 8, 6, 7 | 4 #### | 4 #### | 3 ### | 3 ### |
| ( 6,10) | 1, 2, 3, 4, 5, 6, 9, 8,10, 7 | 3 ### | 3 ### | 2 ## | 2 ## |
| ( 7, 9) | 1, 2, 3, 4, 5, 6, 7, 8,10, 9 | 2 ## | 2 ## | 1 # | 1 # |
| ( 8, 8) | 1, 2, 3, 4, 5, 6, 7, 8,10, 9 | 2 ## | 2 ## | 1 # | 1 # |
| ( 9,10) | 1, 2, 3, 4, 5, 6, 7, 8, 9,10 | 0 | 0 | 0 | 0 |
| answer | 1, 2, 3, 4, 5, 6, 7, 8, 9,10 | | | | |

**図86**　選択ソートでの正解との距離の変化。距離が単調に減少している

| 操作 | | levenshtein | hamming | cayley | ulam |
|---|---|---|---|---|---|
| start | 10, 1, 5, 8, 9, 2, 3, 4, 6, 7 | 8 ######## | 10 ########## | 8 ######## | 4 #### |
| ( 1, ) | 1,10, 5, 8, 9, 2, 3, 4, 6, 7 | 8 ######## | 9 ######### | 7 ####### | 4 #### |
| ( 5, ) | 1, 5,10, 8, 9, 2, 3, 4, 6, 7 | 8 ######## | 9 ######### | 6 ###### | 4 #### |
| ( 8, ) | 1, 5, 8,10, 9, 2, 3, 4, 6, 7 | 8 ######## | 9 ######### | 7 ####### | 4 #### |
| ( 9, ) | 1, 5, 8, 9,10, 2, 3, 4, 6, 7 | 8 ######## | 9 ######### | 8 ######## | 4 #### |
| ( 2, ) | 1, 2, 5, 8, 9,10, 3, 4, 6, 7 | 8 ######## | 8 ######## | 6 ###### | 4 #### |
| ( 3, ) | 1, 2, 3, 5, 8, 9,10, 4, 6, 7 | 6 ###### | 7 ####### | 4 #### | 3 ### |
| ( 4, ) | 1, 2, 3, 4, 5, 8, 9,10, 6, 7 | 4 #### | 5 ##### | 4 #### | 2 ## |
| ( 6, ) | 1, 2, 3, 4, 5, 6, 8, 9,10, 7 | 2 ## | 4 #### | 3 ### | 1 # |
| ( 7, ) | 1, 2, 3, 4, 5, 6, 7, 8, 9,10 | 0 | 0 | 0 | 0 |
| answer | 1, 2, 3, 4, 5, 6, 7, 8, 9,10 | | | | |

**図87**　挿入ソートでの正解との距離の変化。途中で距離が増加している

| 操作 | | levenshtein | hamming | cayley | ulam |
|---|---|---|---|---|---|
| start | 1, 2, 4,10, 9, 5, 7, 6, 8, 3 | 7 ####### | 7 ####### | 5 ##### | 4 #### |
| ( 3, 8) | 1, 2, 4,10, 9, 5, 7, 6, 3, 8 | 7 ####### | 7 ####### | 6 ###### | 4 #### |
| ( 3, 6) | 1, 2, 4,10, 9, 5, 7, 3, 6, 8 | 7 ####### | 7 ####### | 5 ##### | 4 #### |
| ( 3, 7) | 1, 2, 4,10, 9, 5, 3, 7, 6, 8 | 7 ####### | 8 ######## | 6 ###### | 4 #### |
| ( 3, 5) | 1, 2, 4,10, 9, 3, 5, 7, 6, 8 | 8 ######## | 8 ######## | 7 ####### | 4 #### |
| ( 3, 9) | 1, 2, 4,10, 3, 9, 5, 7, 6, 8 | 7 ####### | 8 ######## | 6 ###### | 4 #### |
| ( 3,10) | 1, 2, 4, 3,10, 9, 5, 7, 6, 8 | 7 ####### | 8 ######## | 5 ##### | 4 #### |
| ( 3, 4) | 1, 2, 3, 4,10, 9, 5, 7, 6, 8 | 6 ###### | 6 ###### | 4 #### | 3 ### |
| ( 6, 7) | 1, 2, 3, 4,10, 9, 5, 6, 7, 8 | 4 #### | 6 ###### | 5 ##### | 2 ## |
| ( 5, 9) | 1, 2, 3, 4,10, 5, 9, 6, 7, 8 | 4 #### | 6 ###### | 4 #### | 2 ## |
| ( 5,10) | 1, 2, 3, 4, 5,10, 9, 6, 7, 8 | 4 #### | 5 ##### | 3 ### | 2 ## |
| ( 6, 9) | 1, 2, 3, 4, 5,10, 6, 9, 7, 8 | 4 #### | 5 ##### | 4 #### | 2 ## |
| ( 6,10) | 1, 2, 3, 4, 5, 6,10, 9, 7, 8 | 4 #### | 4 #### | 3 ### | 2 ## |
| ( 7, 9) | 1, 2, 3, 4, 5, 6,10, 7, 9, 8 | 3 ### | 3 ### | 2 ## | 2 ## |
| ( 7,10) | 1, 2, 3, 4, 5, 6, 7,10, 9, 8 | 2 ## | 2 ## | 1 # | 2 ## |
| ( 8, 9) | 1, 2, 3, 4, 5, 6, 7,10, 8, 9 | 2 ## | 3 ### | 2 ## | 1 # |
| ( 8,10) | 1, 2, 3, 4, 5, 6, 7, 8,10, 9 | 2 ## | 2 ## | 1 # | 1 # |
| ( 9,10) | 1, 2, 3, 4, 5, 6, 7, 8, 9,10 | 0 | 0 | 0 | 0 |
| answer | 1, 2, 3, 4, 5, 6, 7, 8, 9,10 | | | | |

**図88**　バブルソートでの正解との距離の変化。途中で距離が増加している

| 操作 | | levenshtein | hamming | cayley | ulam |
|---|---|---|---|---|---|
| start | 10, 1, 5, 8, 9, 2, 3, 4, 6, 7 | 8 ######## | 10 ########## | 8 ######## | 4 #### |
| ( 8, 1) | 10, 8, 5, 1, 9, 2, 3, 4, 6, 7 | 8 ######## | 10 ########## | 9 ######### | 4 #### |
| ( 9, 8) | 10, 9, 5, 1, 8, 2, 3, 4, 6, 7 | 8 ######## | 10 ########## | 8 ######## | 4 #### |
| ( 4, 1) | 10, 9, 5, 4, 8, 2, 3, 1, 6, 7 | 9 ######### | 9 ######### | 7 ####### | 6 ###### |
| ( 6, 4) | 10, 9, 5, 6, 8, 2, 3, 1, 4, 7 | 10 ########## | 10 ########## | 8 ######## | 6 ###### |
| (10, 7) | 7, 9, 5, 6, 8, 2, 3, 1, 4,10 | 9 ######### | 9 ######### | 7 ####### | 6 ###### |
| ( 9, 7) | 9, 7, 5, 6, 8, 2, 3, 1, 4,10 | 9 ######### | 9 ######### | 8 ######## | 6 ###### |
| ( 8, 7) | 9, 8, 5, 6, 7, 2, 3, 1, 4,10 | 8 ######## | 9 ######### | 7 ####### | 6 ###### |
| ( 9, 4) | 4, 8, 5, 6, 7, 2, 3, 1, 9,10 | 7 ####### | 8 ######## | 6 ###### | 4 #### |
| ( 8, 4) | 8, 4, 5, 6, 7, 2, 3, 1, 9,10 | 6 ###### | 8 ######## | 5 ##### | 4 #### |
| ( 7, 4) | 8, 7, 5, 6, 4, 2, 3, 1, 9,10 | 8 ######## | 8 ######## | 6 ###### | 6 ###### |
| ( 8, 1) | 1, 7, 5, 6, 4, 2, 3, 8, 9,10 | 6 ###### | 6 ###### | 5 ##### | 4 #### |
| ( 7, 1) | 7, 1, 5, 6, 4, 2, 3, 8, 9,10 | 6 ###### | 7 ####### | 6 ###### | 4 #### |
| ( 6, 1) | 7, 6, 5, 1, 4, 2, 3, 8, 9,10 | 7 ####### | 7 ####### | 5 ##### | 4 #### |
| ( 7, 3) | 3, 6, 5, 1, 4, 2, 7, 8, 9,10 | 6 ###### | 6 ###### | 4 #### | 4 #### |
| ( 6, 3) | 6, 3, 5, 1, 4, 2, 7, 8, 9,10 | 6 ###### | 6 ###### | 5 ##### | 4 #### |
| ( 4, 3) | 6, 4, 5, 1, 3, 2, 7, 8, 9,10 | 6 ###### | 6 ###### | 4 #### | 4 #### |
| ( 6, 2) | 2, 4, 5, 1, 3, 6, 7, 8, 9,10 | 4 #### | 5 ##### | 3 ### | 2 ## |
| ( 5, 2) | 5, 4, 2, 1, 3, 6, 7, 8, 9,10 | 5 ##### | 5 ##### | 4 #### | 3 ### |
| ( 5, 3) | 3, 4, 2, 1, 5, 6, 7, 8, 9,10 | 4 #### | 4 #### | 3 ### | 2 ## |
| ( 4, 3) | 4, 3, 2, 1, 5, 6, 7, 8, 9,10 | 4 #### | 4 #### | 2 ## | 3 ### |
| ( 4, 1) | 1, 3, 2, 4, 5, 6, 7, 8, 9,10 | 2 ## | 2 ## | 1 # | 1 # |
| ( 3, 1) | 3, 1, 2, 4, 5, 6, 7, 8, 9,10 | 2 ## | 3 ### | 2 ## | 1 # |
| ( 3, 2) | 2, 1, 3, 4, 5, 6, 7, 8, 9,10 | 2 ## | 2 ## | 1 # | 1 # |
| ( 2, 1) | 1, 2, 3, 4, 5, 6, 7, 8, 9,10 | 0 | 0 | 0 | 0 |
| answer | 1, 2, 3, 4, 5, 6, 7, 8, 9,10 | | | | |

**図89**　ヒープソートでの正解との距離の変化。途中で距離が増加している

### 8.7　結論

以上の考察を踏まえると、コンピューターによる整列において「適切なプロセスでは正解との距離が単調に減少する」とは言えない。単調減少を前提とするためには、何らかの条件が必要だろう。本章であれば、例えば、距離としてUlam距離を採用することが妥当と言える条件である。

本稿の主旨に即して言えば、「正解との距離が単調に**減少しなかった**からといって、試行錯誤を捉えているとは限らない」ということになる。

## 8.8 残された課題

本稿は「人の知的プロセスの分析に正解との距離は役に立たない」と主張しているわけではない。整列アルゴリズムと距離の組み合わせについて、なぜ表13の結果になるのか？、特に100%や0%のセルの存在は興味深い。なぜ、全ての問題について正解との距離が増えることがあるのか。なぜ、全ての問題について正解との距離が増えないのか。これについては、今後の研究に委ねる。

本章を踏まえて、人による並べ替えについても検証する。表13の中の51%を「処理の途中で距離が増加する確率」と読み替えられるのは、全ての問題が同じ確率で起きる場合である。現実の、人による並べ替えでは異なるかもしれない。8.3.3で述べた研究[48][49]のデータについて、正解との距離が単調に減少するか検証する。

# 3部

# 9 対話型アプリケーションの性能設計とアクセス解析

ここでは提案手法のコンピューター・システムとしての品質・性能を論じる。以上のようなアプリケーションや分析手法が、実験室ではなく実際の運用で、コンピューター・システムとして成立するかどうかは検証が必要である。従来、教育工学研究では、このような検証の報告が少ない。本研究では、サーバーへ送信される測定データの欠損を、提案手法による分析結果への影響の観点から評価して、問題ないレベルであることを確認した。また、本研究で開発したシステムは、200人規模など、いくつかの学校の授業で実際に使われてきた。これら事例をあげることで、提案アプリケーションや分析手法が実際に使えるものであることを示す。

学習・教育で情報システムを活用する研究や、現場への情報システムの導入が進んでいる。このような研究や事例には、学習者の演習プロセスを測定・分析するものも増えている。しかし、このような事例の報告で、学習システムへのアクセス解析や、分析にあたってのデータ欠損の評価が少ない。われわれが開発・運用する学習システムで性能上のトラブルが発生したときには、問題ページへのアクセスの秒間最大リクエスト数が33件/秒に達し、測定しているユーザー操作ログのデータの一部が失われた。このとき、同じユーザーから同じ問題に0.2秒おきにアクセスがあったことが分かった。対策の結果、その後、同様の性能上のトラブルは起きていない。ユーザー操作ログについては、データベースへの送信にリトライを導入した。その後、これまでの2年半の間に約66万件の操作ログが発生し、そのうち0.06%でリトライが発生していた。リトライが多い授業ではデータ欠損も発生していたが、一方、最近1ヶ月では6万2千件の操作ログが発生しているが、リトライは5件、0.008%であり、欠損は0件である。これらの成果は、学習・教育システムの設計・運用にとって有益であろう。

## 9.1 はじめに

プログラミング教育において、学習者の演習プロセスを測定・分析して、学習者のつまづきを検出したり理解度を測ったりする研究が行われるようになってきた**[77][78][79][80]**。これらの中には事後の分析だけでなく、リアルタイムでも受講生の様子をモニターするものがある。このようなシステムが実際の授業でも使われる事例は今後増えるだろう。しかし、これらにおけるアクセス解析や、分析にあたってのデータ欠損などに関する報告は少ない。

本章では、われわれが研究・開発・運用している並べ替えプログラミング・パズル・システムでのユーザー・アクセスと操作ログ取得の状況について、既発表の文献**[81]**を要約し、特に受講生からのアクセス頻度について改めて検討する。また、ユーザー操作ログのサーバ送信のリトライについて、本学会にて既発表の文献**[82]**に加えて、直近1ヶ月の結果を合わせて報告・検討する。

## 9.2 先行研究

学習アプリケーションを開発して、ユーザー操作を測定・分析する学習分析の研究は数多く報告されるようになった。しかし、学習に直接関係しないためか、システムへの同時アクセス数やレイテンシ(laytency、待ち時間、「待たせる時間」)を分析して設計を評価する報告、あるいは分析対象における欠損データの有無の報告は少ない。

文献**[83]**は、大学入試を想定したCBT (Computer Based Testing)システムのプロトタイプを開発して、試験問題を作成してCBTを実施した研究の報告である。システムはWebアプリケーションとして実装され、受験生はWebブラウザからサーバにアクセスした。この研究の主な対象は試験問題である。1,406名の高校生が受験したとの報告がある(p.50)が、サーバへのリクエスト数やレイテンシに関する定量的な報告はない。

また、文献[83]の「別添資料6 大規模CBTシステム構築への課題とその解決策」は、大規模CBTを検討したものである。性能上の懸念から、全受験生が同時に受験するのではなく、いくつかの日程に分けて受験することを想定している。このために項目応答理論によって、どの機会でも同じ難易度の試験問題となるようにする。また、入試用の専用端末を導入し、入試用アプリケーションまたはWebブラウザを使い(p.4)、試験問題は事前に端末に配付し(p.3)、受験者の解答も会場内ネットワークの分散ハッシュテーブル(Distributed Hash Table、DHT)に時々刻々と保存する(p.6)ことを提案している。

文献[83]の別添資料6には、この他にも参考になる記述がある:「ゲームのプラットフォームの実例…プレイ中の利用者も250万人居る。」「ゲームは有料なので、…遅延に対しては、厳しいと予想される」(p.12)、「(仮想サーバより物理サーバが無難であるが)年1回の試験では、物理サーバを確保するのはコスト的に難しいのではないか」(p.13)。「合格発表にWebを利用している大学では、Webサーバ1台あたり700同時アクセスを設計上の上限としている」(p.14)。

そもそも、システムの安定性や性能に言及する文献が少ない。教育工学の研究のガイドブックとしては、教育工学会が監修した教育工学選書[*11]がある。そのシリーズのうち「教育工学研究の

方法」**[84]**の「第7章 教育システム・ツールの開発研究の方法」では、実践・評価のフェーズで「機能や操作性、インタフェースなど、技術的な側面を診断的に評価する。」としている。また改善フェーズでは「テクノロジー・プッシュの開発では、機能や操作性、インタフェースなど技術的な問題を解決する。」としている。いずれも、システムの安定性や性能に言及していない。また、同選書でシステム開発の研究に絞った「教育工学とシステム開発」**[85]**でも、システムの安定性や性能に関する設計や評価の言及がない。

## 9.3 これまでの取り組み

ここでは、文献[81]から要約して、本稿の対象システムである並べ替えアプリケーション「ジグソー・コード」システム、そこで発生したトラブル、およびトラブルへの対策の概要を述べ、最後に文献[81]で「残された課題」とした懸案を述べる。この懸案が、本研究で取り組んだ課題である。

### 9.3.1 ジグソー・コードと操作の測定

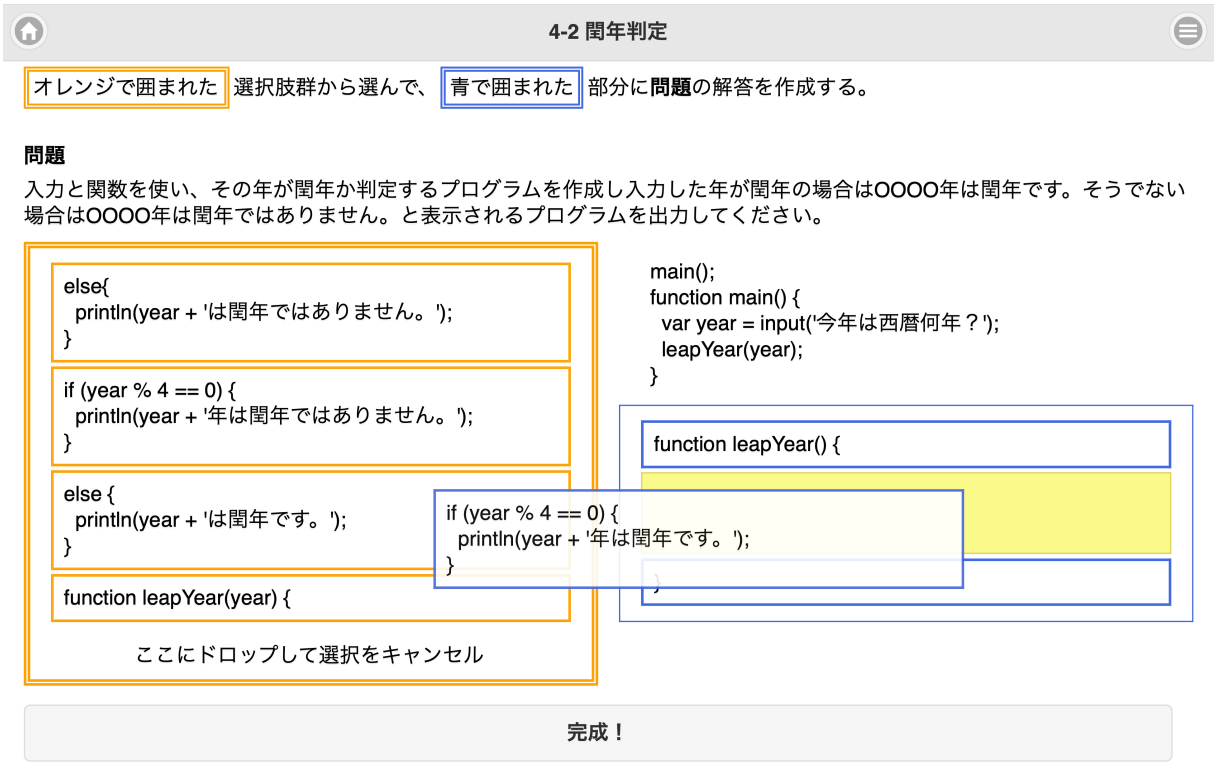



**図90** 並べ替えプログラミング・パズルのジグソー・コード

*11 出版物のご案内｜日本教育工学会（JSET）
https://www.jset.gr.jp/publications/ ↗

ジグソー・コードは並べ替えプログラミングのアプリケーション(**図90**)で、プログラミングの演習などで使われている。これらアプリケーションは、ユーザーの並べ替え操作を測定している(**図91**)。われわれは、これらアプリケーションの機能/ユーザー・インタフェース(UI)、操作ログのデータ、データの分析手法、分析を学習・指導に活用する手法を研究してきた[48][49]。これらの研究は「ユーザーが問題を解くプロセス(過程)」に着目するものである。このプロセスを「プレイ」と呼ぶ。

操作ログには種類があって、「プレイ開始(start)」、「プレイ完了(completed)」、「行をドラッグ開始(drag)」、「行をドロップ(drop)」、「行を(左右どちらかの)枠から取り出す(remove)」、「行を(左右どちらかの)枠に入れる(receive)」の6通りがある。removeやreceiveは枠をまたがる移動で発生し、同じ枠内で行を移動した場合は発生しない。そこで「ユーザー操作の数 ≠ 操作ログのレコード数」となる。後の章で操作ログ数や操作ログ送信のリクエスト数を示すが、それらがそのままユーザー操作数ではないことに留意しなくてはならない。例えば、図90のようなUIの場合、左のオレンジ色の選択肢枠から取り出して、右の青色の解答枠に移動する1つの操作では、「行をドラッグ開始」、「行をドロップ」、「行を(左右どちらかの)枠から取り出す」、「行を(左右どちらかの)枠に入れる」の4つの操作ログが発生する。青色の解答枠内でピースを移動する1つの操作では、「行をドラッグ開始」、「行をドロップ」の2つの操作ログが発生する。

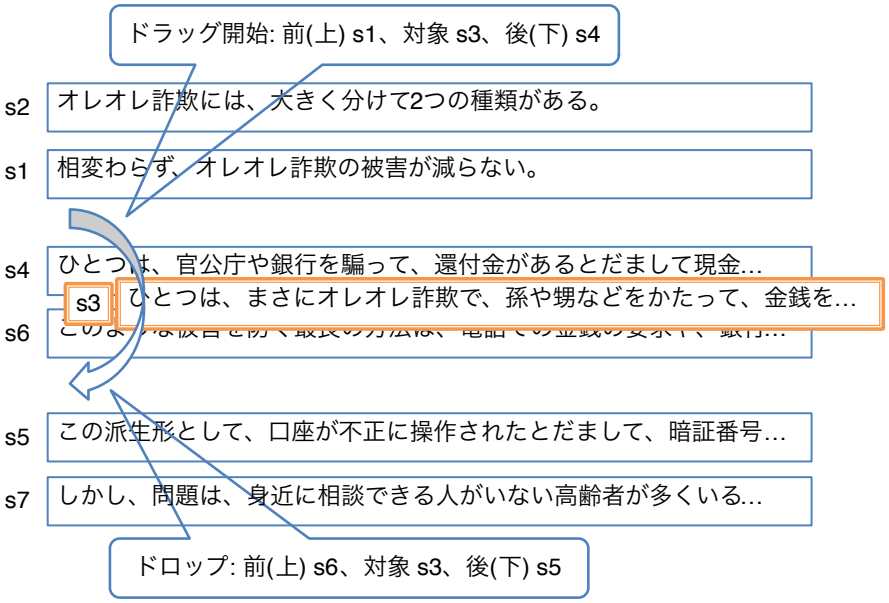



**図91**　並べ替え作文のジグソー・テキストにおける、並べ替え操作の測定

### 9.3.2　システム構成

本稿のシステムは、Google社のクラウド・プラットフォームのGoogle App Engineを使っている。

**図92**にジグソー・コードの(トラブル発生時ではなく)現在のシステム構成を示す。

図90の画面は、HTMLとCSSとJavaScriptで記述されている。ブラウザが図90のような問題ページをサーバにリクエストすると、ページを記述したHTMLがサーバから返される。そのHTMLを解釈して、ブラウザはCSSやJavaScriptのスクリプト・ファイルをサーバにリクエストする。読み込まれたJavaScriptは、並べ替え操作を実現するとともに、問題データをサーバに非同期でリクエストする。問題データはGoogleスプレッドシートで記述されている。サーバはGoogleスプレッドシートのAPI (Sheets API)を使って問題データを取り出し、変換してブラウザに返す。この問題データのリクエストに対する処理はPythonで記述されている。ブラウザ側のJavaScriptが、このデータを受け取って解釈して問題がユーザーに示される。

図91のユーザー操作の測定処理はJavaScriptで記述されている。ユーザーがプログラム行をドロップなどすると、ドロップされた行のIDや操作時刻などを、JavaScriptが非同期にサーバに送信する。サーバは、操作ログ・データを受け取って、データベースに登録し、登録処理の成否をブラウザに返す。

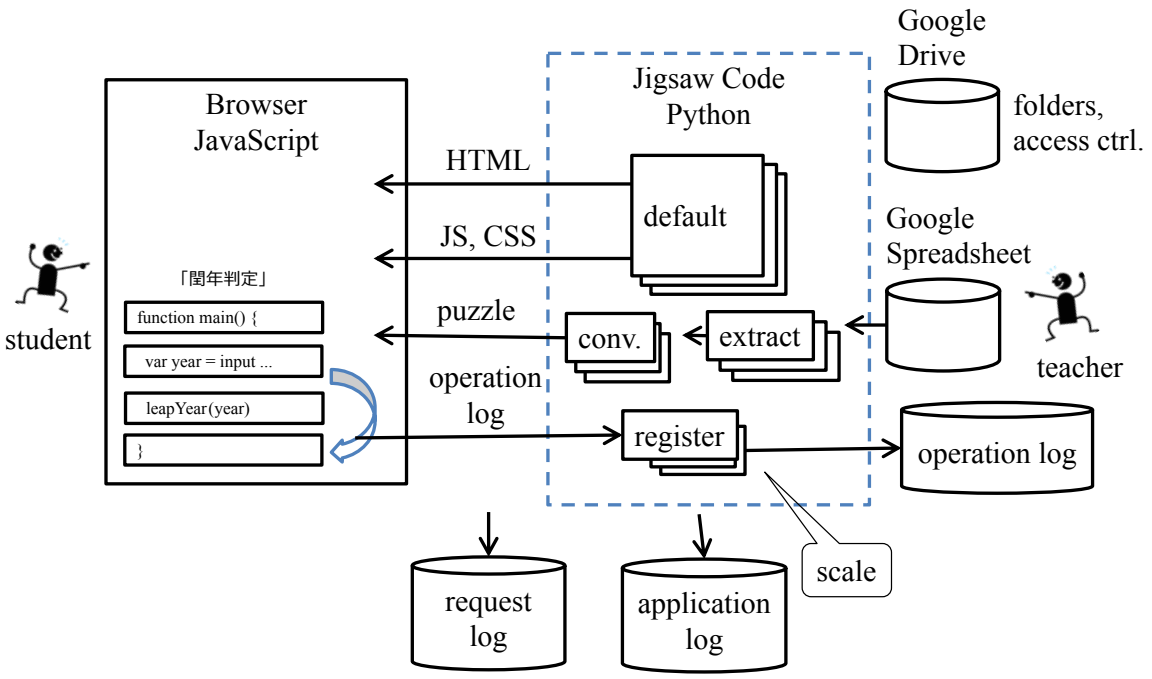



**図92　ジグソー・コードのシステム構成**

### 9.3.3　性能上のトラブル発生と対策

2020年度後期の授業でジグソー・コードを使ったとき、トラブルが発生した。

授業は大学の文理融合型の学部1年生向け必須授業のコンピューティング実習であった。内容は、JavaScript、HTMLを使って、変数、条件分岐、繰り返し、関数などプログラミングの初歩を学ぶものであった。受講人数は約230人であった。授業はオンライン形式であった。

授業の最初に前回の復習として小テストを3問、「始め」の合図で一斉に解き始めた。すると、複数回の授業の小テストで、問題を解き始められるようになるまで2分以上待たされるというトラブルが起きた。

クラウド・プラットフォームが出力したサーバへのリクエスト・ログ、開発したアプリケーションが出力したアプリケーション・ログ、およびデータベース登録に成功したユーザー操作のログ(操作ログ)から、分かる範囲で現象を調査した。サーバに対するリクエストのログを調べたところ、操作ログのデータの一部が、データベースへの登録に失敗し、データが失われていたことが分かった。

以下に具体的に述べる。

**リクエスト数**

**図93**は、トラブル発生時の1秒間のリクエスト数の推移で、縦軸はリクエスト件数、横軸が経過時刻で1,200秒 = 20分間にわたる。問題ページへのアクセスが急激に立ち上がり、アクセス開始から52秒かけて秒間最大リクエスト数が33件/秒に達していた。この時点までの累計リクエスト数は775件であった。2分ほどでピークアウトしている。

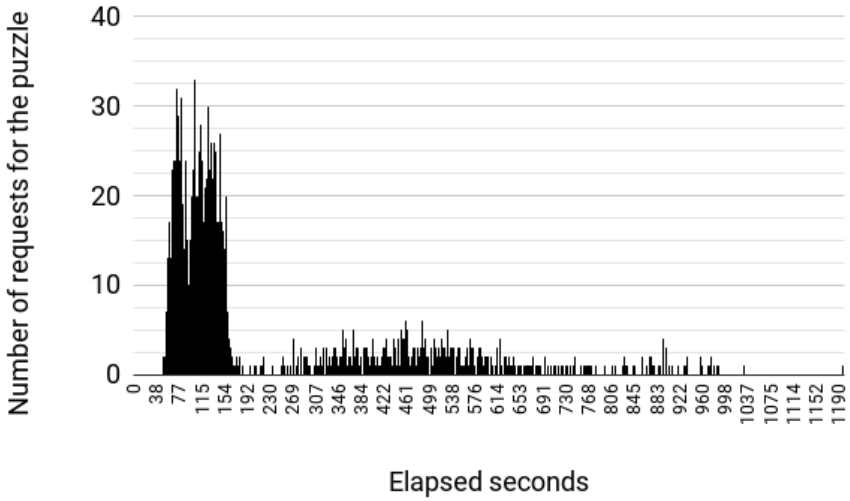



**図93**　サーバへのリクエスト数の推移

**リクエスト間隔と再読み込み**

**表14**は、あるユーザーからの、リクエストの経過である。表の上から下へ時間が経過し、各行が各リクエストに対応する。前回のリクエストから0.2秒でリクエストしている場合は、ブラウザの応答を待って操作していないと考えられる。以前の授業で待たされた経験から、この回の授業ではこのような操作になっているか、あるいは、これが「デジタル・ネイティブ世代におけるアプリ操作の作法」である可能性もある。

**表14**　あるユーザーからのリクエストの間隔(秒)と、各リクエストごとのサーバ側レイテンシ

| puzzle name | request interval(sec.) | latency |
|---|---|---|
| ドーナツプログラム | - | 33.9 |
| ドーナツプログラム | 4.5 | 23.4 |
| ドーナツプログラム | 6.4 | 27.7 |
| ドーナツプログラム | 0.7 | 27.1 |
| ドーナツプログラム | 0.2 | 5.9 |
| ドーナツプログラム | 0.2 | 26.7 |
| ドーナツプログラム | 0.5 | 26.1 |
| ドーナツプログラム | 0.2 | 26.0 |
| ドーナツプログラム | 0.2 | 25.7 |
| ドーナツプログラム | 0.3 | 13.9 |
| ドーナツプログラム | 9.4 | 10.1 |
| ドーナツプログラム | 4.7 | 30.9 |
| ドーナツプログラム | 7.6 | 30.6 |
| ドーナツプログラム | 43.0 | 3.6 |
| ドーナツプログラム | 2.3 | 11.3 |
| ドーナツプログラム | 3.4 | 3.4 |
| ドーナツプログラム | 0.9 | 4.9 |
| ドーナツプログラム | 0.1 | 1.8 |
| 閏年判定 | 373.5 | 0.0 |
| 手裏剣を描くプログラム | 207.8 | 0.0 |

表14で分かるように、ユーザーがブラウザで再読み込みすることで、ブラウザがサーバに新たなリクエストをしても、サーバは前のリクエストの処理を続けている。あるリクエストの0.2秒後に再読み込みしても、サーバは5.9秒かけて再読み込み前のリクエストを処理して応答している。その間にサーバ資源を消費しているので、サーバ側内で、ある処理が別の処理の応答をどれくらい待ち続けるべきかが、設計上のポイントの1つとなることが分かる。

**操作ログの全件数と失敗数**

クラウド・プラットフォーム側のログによると、授業時間の午前9時から12時までの間に、データベースへの登録をリクエストされた操作ログの件数は22,104件であった。このうち15件のリクエストが、アプリケーションに渡されず中断されたことが分かった。3問で22,104件のうちの15件で、0.068%の操作ログデータが失われたことになる。

### 9.3.4　取り組んだ課題

トラブルは3つに整理できた:

1. 問題データをGoogleスプレッドシートから取り出す処理が重い
2. さらにこの処理の重さによって他の処理が実行されず中断してしまう
3. 研究の観点からは、操作ログデータが失われること

本章では課題として、操作ログが失われることにフォーカスする。続く節で、施した対策と残る懸案を述べる。

### 9.3.5 対策: 操作ログ送信のリトライとシリアル番号の付与

操作ログが失われるリスクを小さくするために、送信に失敗したとき、ブラウザ側(JavaScript)で操作ログの送信をリトライするように修正した。リトライの実態が分かるように、操作ログにはリトライ回数を設定した。回数が0なら一番最初の送信、1より大きい場合は何度目かの再送信によって登録された操作ログデータである。また、リクエスト・ログを見なくても操作ログの欠損を検出できるように、送信に操作ログのシリアル番号を付与した。操作ログには6種類あるが、JavaScriptアプリケーションが操作ログデータを作成した順序に1から番号を振った。クラウド・プラットフォームのリクエスト・ログはシステム管理者でないと見られないが、シリアル番号はデータ分析者やデータ分析アプリケーションが見られる。

**表15** リトライの分布

| puzzle # | # of plays | # of op. logs retried | puzzle # | # of plays | # of op. logs retried |
|---|---|---|---|---|---|
| p#1 | 2 | 44 | p#22 | 1 | 2 |
| p#2 | 1 | 35 | p#23 | 1 | 2 |
| p#3 | 1 | 33 | p#24 | 2 | 2 |
| p#4 | 3 | 31 | p#25 | 2 | 2 |
| p#5 | 1 | 23 | p#26 | 1 | 1 |
| p#6 | 2 | 19 | p#27 | 1 | 1 |
| p#7 | 4 | 18 | p#28 | 1 | 1 |
| p#8 | 1 | 17 | p#29 | 1 | 1 |
| p#9 | 1 | 17 | p#30 | 1 | 1 |
| p#10 | 2 | 17 | p#31 | 1 | 1 |
| p#11 | 1 | 15 | p#32 | 1 | 1 |
| p#12 | 1 | 14 | p#33 | 1 | 1 |
| p#13 | 1 | 11 | p#34 | 1 | 1 |
| p#14 | 1 | 11 | p#35 | 1 | 1 |
| p#15 | 1 | 10 | p#36 | 1 | 1 |
| p#16 | 1 | 8 | p#37 | 1 | 1 |
| p#17 | 1 | 7 | p#38 | 1 | 1 |
| p#18 | 2 | 4 | p#39 | 1 | 1 |
| p#19 | 1 | 4 | p#40 | 1 | 1 |
| p#20 | 4 | 4 | p#41 | 1 | 1 |
| p#21 | 2 | 3 | p#42 | 1 | 1 |
| | | | total | 57 | 370 |

#### 9.3.6 懸案: 操作ログのリトライ、2重登録および欠損の実態

対策によってトラブルは発生しにくくなるが、操作ログ送信のリトライや欠損の実態は確認する必要がある。

クラウド・プラットフォームのスケールによって操作ログは複数のプロセス(クラウド・プラットフォームではインスタンスと呼ばれる)で処理されるため、DB登録順序 ≠ 操作順序となり、以前から要注意であった。今後はさらに、リトライの導入によって、ブラウザにエラーが返っても実際は登録されてる2重登録の可能性にも注意が必要となった。

### 9.4 研究方法

まず、リトライ発生の有無を、リトライ実装後の全操作ログ・データについて調べる。欠損については、操作ログ送信のリトライ回数が多いプレイで発生する可能性が大きいと考えられる。そこで、次の2つの方法で2重登録やデータ欠損の有無を調べる:

1. リトライ回数の多いプレイにおける欠損: リトライ回数の多いプレイを見つけ出し、シリアル番号を手がかりに欠損を調べる。最悪のケースを調べるものである。
2. 直近1ヶ月のプレイにおける欠損: 直近1ヶ月の全操作ログについて、欠損を調べる。

## 9.5 結果

### 9.5.1 リトライされた操作ログ

操作ログの送信のリトライを実装の後、現在までの操作ログは663,088件あった。このうち、リトライされた操作ログ、すなわちリトライ回数が0より大きいものは370件であった。これらは42個の問題、57件のプレイ(解くプロセス)にわたり、最も大きいもので1個の問題に4件のプレイでリトライが発生していた。1個の問題に対してリトライされた操作ログが最も多いのは、2個のプレイにわたって合計44件の操作ログにリトライがあった。最もプレイ数が多い問題と、最もリトライされた操作ログ数が多い問題は異なる(**表15**)。

全操作ログに対してリトライされた操作ログの割合は

$$370 \div 663,088 \fallingdotseq 0.056\%$$

となった。

リトライした操作ログの数が最も多いp#1の問題では、2つのプレイでリトライがあった。そのうち1つのプレイではリトライが1回、もう1つのプレイで43回あった。問題p#1のプレイ全体では5,898件の操作ログがあったので、これに対するリトライされた操作ログの割合は

$$43 \div 5,898 \fallingdotseq 0.73\%$$

となった。他の問題を含めた全体よりも割合が1桁多いので、リトライが発生するには問題、あるいは問題が解かれる場面が関係すると考えられる。

### 9.5.2 リトライ回数

リトライがあった操作ログのうち、操作ログの数が多いものについて、リトライ回数の内訳をプレイごとに**表16**に示す。例えば、問題p#3について、プレイd#4では、3回のリトライで登録された操作ログが5件、6回のリトライで登録された操作ログが9件あった。操作ログの総計が多いことと、操作ログのリトライ回数が多いこととは別であった。

**表16**　リトライした操作ログのリトライ回数

| | | number of retries | | | | | | |
|---|---|---|---|---|---|---|---|---|
| puzzle # | play # | onece | twice | 3 times | 4 | 5 | 6 | total |
| p#1 | d#1 | 13 | 10 | 8 | 4 | 8 | | 43 |
| | d#2 | 1 | | | | | | 1 |
| p#2 | d#3 | 18 | 13 | 4 | | | | 35 |
| p#3 | d#4 | | | 5 | 8 | 11 | 9 | 33 |
| p#4 | d#5 | 4 | 1 | 3 | 6 | 3 | | 17 |
| | d#6 | | 4 | 6 | | | | 10 |
| | d#7 | 4 | | | | | | 4 |
| p#5 | d#8 | 16 | 7 | | | | | 23 |
| p#6 | d#9 | 10 | 6 | 2 | | | | 18 |
| | d#10 | 1 | | | | | | 1 |
| p#7 | d#11 | 1 | 7 | 7 | | | | 15 |
| | d#12 | 1 | | | | | | 1 |
| | d#13 | 1 | | | | | | 1 |
| | d#14 | 1 | | | | | | 1 |
| p#8 | d#15 | | 4 | 4 | 4 | | 5 | 17 |
| p#9 | d#16 | | 1 | 4 | 7 | 4 | 1 | 17 |
| p#10 | d#17 | 8 | 8 | | | | | 16 |
| | d#18 | 1 | | | | | | 1 |
| p#11 | d#19 | 1 | | 2 | 2 | 2 | 8 | 15 |
| p#12 | d#20 | 10 | 4 | | | | | 14 |
| p#13 | d#21 | 11 | | | | | | 11 |
| p#14 | d#22 | 9 | 2 | | | | | 11 |
| p#15 | d#23 | 8 | | 2 | | | | 10 |
| p#16 | d#24 | | | | 3 | 4 | 1 | 8 |
| p#17 | d#25 | 4 | 3 | | | | | 7 |
| p#18 | d#26 | 3 | | | | | | 3 |
| | d#27 | 1 | | | | | | 1 |
| p#19 | d#28 | 4 | | | | | | 4 |

### 9.5.3　データ欠損と２重登録

**リトライ回数の多いプレイにおける欠損**

リトライされた370件の操作ログには重複したものが含まれている可能性がある。表16でリトライ回数が多かった問題p#3のプレイd#4について、登録された40件の操作ログの内訳を**表17**に示す。

**表17**　リトライ回数の多いプレイの操作ログ

| type of operation | number of retries | serial number |
|---|---|---|
| start | 0 | 1 |
| drag | 0 | 2 |
| remove | 0 | 3 |
| receive | 0 | 4 |
| drop | 0 | 5 |
| drag | 0 | 6 |
| drop | 0 | 7 |
| ? |  | (**? 8**) |
| remove | 6 | 9 |
| receive | 6 | 10 |
| drop | 6 | 11 |
| drag | 6 | 12 |
| drop | 6 | 13 |
| drag | 6 | 14 |
| drop | 5 | **15** |
| drop | 6 | **15** |
| drag | 6 | 16 |
| drop | 6 | 17 |
| drag | 5 | 18 |
| drop | 5 | 19 |
| drag | 5 | 20 |
| remove | 5 | 21 |
| receive | 5 | 22 |
| drop | 5 | 23 |
| drag | 4 | **24** |
| drag | 5 | **24** |
| remove | 4 | **25** |
| remove | 5 | **25** |
| receive | 4 | **26** |
| receive | 5 | **26** |
| drop | 4 | **27** |
| drop | 5 | **27** |
| drag | 3 | **28** |
| drag | 4 | **28** |
| remove | 3 | **29** |
| remove | 4 | **29** |
| receive | 3 | **30** |
| receive | 4 | **30** |
| drop | 3 | **31** |
| drop | 4 | **31** |
| completed | 3 | 32 |

表の40件(「type of operation」列が「?」の行を除く40行)の操作ログのシリアル番号(serial number列)を見ると、「24」など9個の数が重複している。すなわち、実際の操作ログは $40 - 9 = 31$件であったことになる。しかし、シリアル番号の最後は32で、32件の操作ログがあったことを示している。実際、シリアル番号は8が欠けている。順番から考えると「行をドラッグ開始(drag)」が失われたと考えられる。シリアル番号の最初の1が「プレイ開始(start)」なので、この前には失われた操作ログはない。また、最後の32が「プレイ完了(completed)」なので、この後にも失われた操作ログはない。

同じ日にこの問題p#3を解いた操作ログは2,010件あったので、1問について $1 \div 2,010 = 0.050\%$の操作ログデータが失われたことになる。これは最も悪いケースに関する数値だが、それでも、以前のトラブル発生時の0.068%より改善した。

**直近1ヶ月のプレイにおける欠損**

2023年11月1日から2023年11月29日までの操作ログの数は62,533件。これらを出力したプレイは3,185件あった。

操作ログ送信のシリアル番号を手がかりに、番号が途切れたときログが欠けたと判定すると、この間の欠損は0件だった。送信リトライは2つのプレイで発生し、リトライ回数は最大1回であった。

## 9.6 考察

### 9.6.1 欠損の有無と負荷

直近1ヶ月の送信ログには欠損が見られなかった。またリトライ回数も少なかった。この期間を含む、大学の夏休み明けにあたる10月から11月のリクエスト数の推移を、クラウドシステムの機能を使って**図94**に図示する。1週間ごとにピークが現れていて、これは毎週金曜日の授業に該当する。これらリクエスト数のピークは数個/秒で一定していて、冒頭で述べた性能上のトラブル発生時と比べると1桁小さく、当時と比べて負荷が小さいことが分かる。リクエスト数は授業の進め方や問題の解き方に依存する。例えば、一斉に問題を解き始めれば操作ログのリクエストが集中する。この期間に操作ログを安定して送信できた理由には、このような事情もあるだろう。

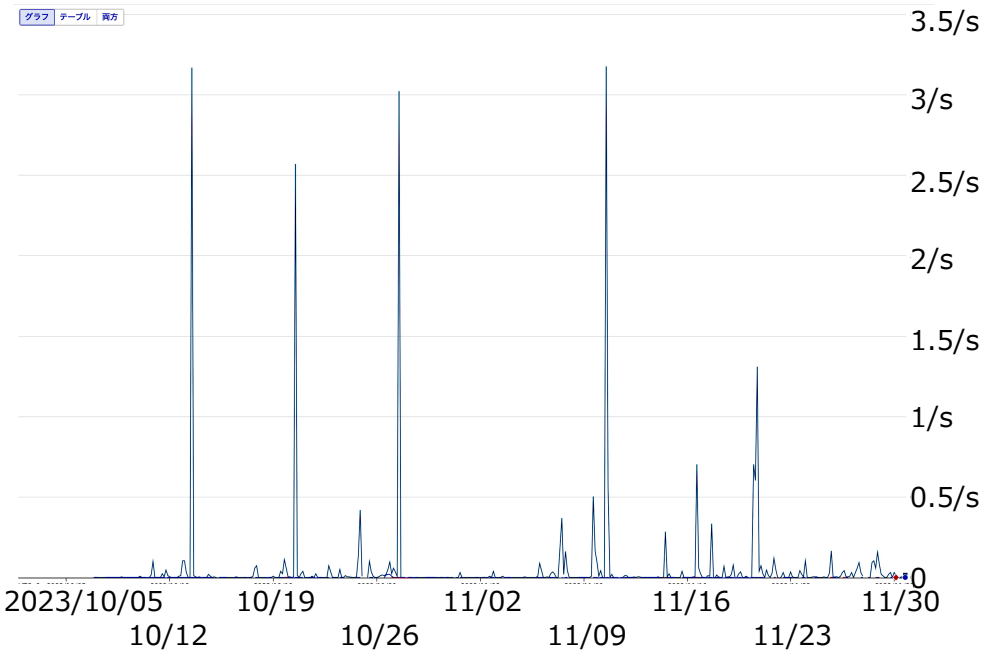



**図94**　10月から11月のリクエスト数の推移

### 9.6.2　欠損の評価

ここで既発表の文献[81]から、データ欠損の評価を再掲する。

ジグソー・コードの操作ログの分析では、操作過程のパターンを見つけるために、操作対象となった行(パズルのピース)の共起行列を使うことが多い。共起行列は、ある行を動かした次にどの行を動かしたかの回数を集計する。「次に動かされた」という時間的な近さを見ることから、従来のテキスト分析の共起と区別して時間的な共起(temporal co-occurrence)と呼ぶ。**図95**は、ある問題を解くプロセスの操作ログを集計した時間的な共起行列である。行や列の見出しにあるs4やs5は問題に含まれる行(ピース)のIDである。図では、s4行の次にs5行を動かした回数が全部で76あったことが分かる。

図95のように時間的な共起頻度には偏りがあり、操作の傾向を表していると考える。「s4の次にs5」という共起の頻度の評価として、共起頻度の平均値と標準偏差を使う。図では、セルの値の合計が1,608、平均値が25.1、標準偏差が16.9である。**平均値** + **標準偏差** = 42よりも大きいセルを黄色の背景(薄い背景)、**平均値** + **標準偏差** × 2 = 58.9より大きいセルを赤い背景(濃い背景)で示す。

| n \ n+1 | s3 | s4 | s5 | s6 | s8 | s9 | s10 | s11 |
|---|---|---|---|---|---|---|---|---|
| s3 | 29 | 32 | 14 | 35 | 13 | 27 | 15 | 18 |
| s4 | 18 | 47 | 76 | 21 | 5 | 66 | 32 | 36 |
| s5 | 31 | 27 | 34 | 77 | 15 | 17 | 22 | 25 |
| s6 | 39 | 15 | 11 | 39 | 25 | 14 | 10 | 10 |
| s8 | 15 | 12 | 6 | 21 | 26 | 6 | 3 | 5 |
| s9 | 10 | 77 | 32 | 13 | 5 | 37 | 38 | 36 |
| s10 | 17 | 14 | 24 | 28 | 9 | 17 | 39 | 23 |
| s11 | 25 | 14 | 42 | 35 | 10 | 9 | 16 | 48 |

**図95**　並べ替え操作の共起行列

仮に、冒頭のトラブル発生時のように、失われた操作ログが15件あって、それらが全てs4の次にs4を動かすものだったとすると、「s4の次にs4」という共起の頻度は$48 + 15 = 63$となり、赤い背景に該当することになる。しかし、そのような確率は$48 \div 1,608 = 3\%$であることを共起行列は示している。さらに、操作ログには種類があって、「プレイ開始」、「プレイ完了」、「行をドラッグ開始」、「行をドロップ」、「行を(左右どちらかの)枠から取り出す」、「行を(左右どちらかの)枠に入れる」の6通りである。共起行列は「行をドロップ」だけを集計する。「行をドラッグ開始」と「行をドロップ」は必ず対で発生するので、失われた操作ログの半分が「行をドロップ」であると見積もると、3%よりもさらに小さくなる。

トラブル発生時については、失われた操作ログが共起行列に基づく分析に影響を与えた可能性は小さいと考えられる。また、対策後についても、このような分析にあたって欠損が影響している可能性は小さいと考えられる。

以上は操作ログの欠損をデータ分析の観点から評価したものである。先行研究の章で検討した大規模CBTのように送信データが受験生の解答であれば、全く異なる評価となり、技術的な対策も異なるものとなる。

### 9.6.3　使用事例

最後に、その他の事例も合わせて、開発したアプリケーションの使用事例を、性能の観点から整理する:

大学Oでは、2015年度から2019年度にかけて、テクニカルライティングの演習で、Topic Writerとジグソー・テキストを使った。50名ほどのクラスであった。

大学Tでは、2018年度から2019年度にかけて、ロジカルシンキングの演習でジグソー・テキストを使った。40名から50名ほどのクラスであった。

大学Hでは、2018年度から2022年度にかけて、テクニカルライティングの演習でTopic Writerやジグソー・テキスト、マネジメント論の演習ではTopic Writerを使った。テクニカルライティングの演習は10名弱、マネジメント論の演習は300名弱の受講生であった。

大学Aでは、2020年度から2023年度まで、プログラミングの演習などでジグソー・コードを使った。受講生は200名程度で、本章の事例がこれである。

文章を書くTopic Writerのようなアプリケーションでは、この章で検討したような性能上の問題は起きていない。Topic WriterやV字エディタでは、ワークシート(ロジック・ツリー)データへのアクセスがネックとなる可能性があるが、アクセスの「同時」が、ジグソー・テキストなどとは異なり緩やかだと考えられる。

### 9.6.4 本研究の意義

本章のようなアクセス解析の報告は、システム設計やテストケース設計の参考となる。図93は、立ち上がりから52秒後にリクエストが33件/秒に達することを示しているので、テスト環境をそのように設定することになる。このように「テスト環境の設定ができる」ことなどが、本章のような稼働統計の意義である。

様々な学習・教育システムの研究成果が発表されているが、このような稼働統計の発表が増えることを願っている。

## 9.7 結論

提案アプリケーションについて、データ欠損を検出できること、および、実際に起きている欠損が提案する分析手法に与える影響が十分に小さいことを示せた。

# 10 スクリーン・リーダーによる複雑な情報処理における考え方の分析

ここでは本研究をアクセシビリティの観点から論じる。障害者差別解消法の改正など、我が国は障害のある人もない人も共に暮らせる社会を目指している。新しい技術を開発し導入を目指す研究は、障害がある人でも、それによって利益を得られるよう努力する必要があるだろう。本研究では、ジグソー・テキストにスクリーン・リーダーで操作できるUIを用意し、スクリーン・リーダーによるプレイを測定・分析した。これによって、画面を見なくても並べ替え問題を解ける。このUIは前記のアプリケーションの条件を満たすものである。予備実験を実施し、このUIで解くプロセスを分析したところ、スクリーンリーダーで解く方が目で見て読んで解くよりも、最初の操作にかける時間が長い傾向が見られた。UIによって解くプロセスが異なることは、異なる思考が起きていることを示唆し、ひいては同じ問題を解いてるといえるかどうかの疑問に達する。今後、さらなる研究が必要である。

人の複雑な思考を支援する従来の図表やコンピューターシステムの多くは、視覚を前提に発明・開発されてきた。では、人が複雑な情報処理を音声インターフェース(Voice User Interface, VUI)で行う場合、視覚で処理する場合と同じ考え方をするだろうか。本研究の目的は、人とコンピューターとのインタラクションの測定・分析によって「考え方」の違いを明らかにすることである。研究の第一段階として、視覚(Graphic User Interface, GUI)による操作とスクリーン・リーダーによる操作とを測定・分析し、テキストを理解するときの考え方の違いを明らかにする。このために、視覚およびiPhoneのVoiceOverで操作できるテキストのジグソー・パズル・アプリケーション「ジグソー・テキスト」を開発した。予備実験として、7問のパズルを2人の被験者がVoiceOverで解き、並べ替え操作を記録して分析した。同じ問題を延べ220名が視覚で解いた場合と、今回VoiceOverの読み上げで解いた場合とを比べると、視覚の場合の方がVoiceOverよりも最初のピースを動かすまでの時間が長い傾向があった。視覚では正解順序の見通しを立ててからピースを動かし始めたのに対して、VoiceOverではピースを並べながら正解を探り、完成に近づくとピースを動かすたびに全文を読み上げさせて確認していた可能性がある。今後は被験者を増やして本実験を行う。また、スマートスピーカーを使って同様の実験を行う。最後に、インタラクティブな音声インターフェースによる図表の理解への展望を述べる。

## 10.1 はじめに

近年、世界的にスマートスピーカー(Google Home、Amazon Echo、など)やオーディオ・ブック(Audibleなど)といったVUIを利用するシステムが普及してきた。これらVUIは「今日の天気」といった単純な応答を前提に設計・評価されている。VUIデザイン・ガイドには、例えば「なるべくユーザーにコマンドを暗記させない」とある。

人の複雑な情報処理を支援する従来システムの多くは視覚前提のGUIで実装されている。GUIではアイコンやショートカットが多用され、これらを暗記することで、ユーザーは複雑な情報処理に没頭できる。

国連の開発目標SDGs (Sustainable Development Goals)では、教育・雇用・社会参加・経済参加などですべての人に等しい機会を保証するという目標を掲げている。視覚だけでなく様々なインタフェースで高度な仕事を行える環境が求められる。

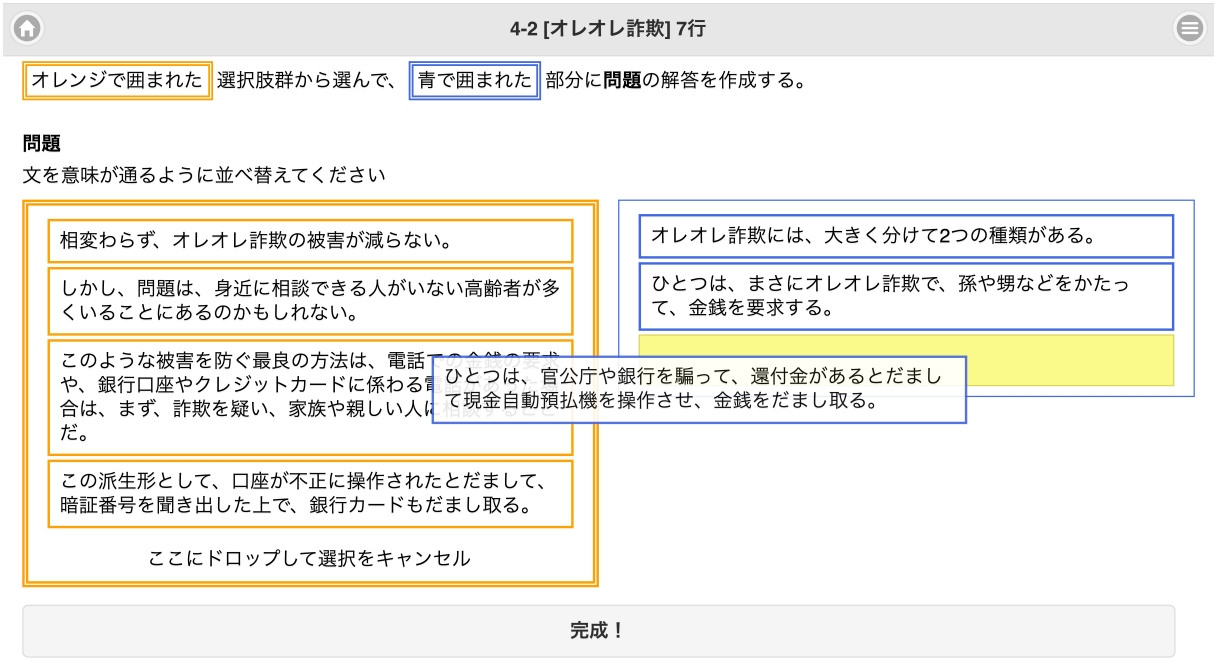



**図96** 問題「4-2 [オレオレ詐欺]」を目で見てドラッグ&ドロップで解くジグソー・テキスト2の画面

VoiceOver**[86]**といったスクリーン・リーダーなどのVUIは、GUIのショートカット同様に、すぐに使いこなせるものではなく、実験等の敷居が高い。だからこそVUIアプリが「単純な応答を前提に設計・評価されている」とも言えて、本研究は「鶏と卵」の様相を呈している。

人の複雑な情報処理を支援するVUIをGUIと比較して評価できる、晴眼者にとっても敷居の低い仕組みはどのようなものか？

学習管理システムLMSや教材アプリをスクリーンリーダーで操作できるようにした研究はあるが**[87][88]**、それらにおける思考プロセスを明らかにしたものはない。

## 10.2 目的

本章は、人がVUIでコンピューター・システムと対話しながら複雑な問題を解決するときの考え方の特徴を、GUIとの比較で明らかにする。このために、敷居の低いアプリを工夫して開発する。それを通して、本稿のような新しいアプリケーションを提案する研究が、障害のある人々にも利益をもたらす道筋を示す。

## 10.3 研究方法

人がVUIでシステムと対話しながら複雑な情報処理を行うプロセスを測定・分析するために、敷居の低い仕組みを用意して実験する。具体的には、文章の並べ替えパズルを使う。この章では、使うシステム、実験の手順および分析の方針を示す。

### 10.3.1 システム

「敷居の低い仕組み」として、核心部分以外はGUIで操作できるインタフェースを、まず用意して、小さなステップで段階的に進められるようにする。複雑な情報処理を求めるGUIアプリを、まず、スクリーン・リーダーで操作できるようにする。次に、スクリーン・リーダーでの操作を求める部分を限定する。例えば、問題／パズルそのものに至るまではGUIで操作できるようにし、核心の部分だけスクリーン・リーダーを使わざるをえないようにする。

**図96**は、文章の断片(ピース)を並べ替えて完成させる文章のジグソー・パズル「ジグソー・テキスト2」の画面である。図は「ひとつは、官公庁や銀行を騙って…」というピースを左側の枠から選んで、右側の枠に移動しているところである。**図97**は、ジグソー・テキスト2のスクリーンリーダー向けUI「ジグソー・テキスト3」である**[89][90]**。図97の左側の画面でユーザーが移動するピースを選択すると、画面は図の右側に遷移する。そこで移動先を選択すると、ピースがそこへ移動して左側の画面に遷移する。ドラッグ&ドロップによる並べ替えよりも手堅く操作できる。

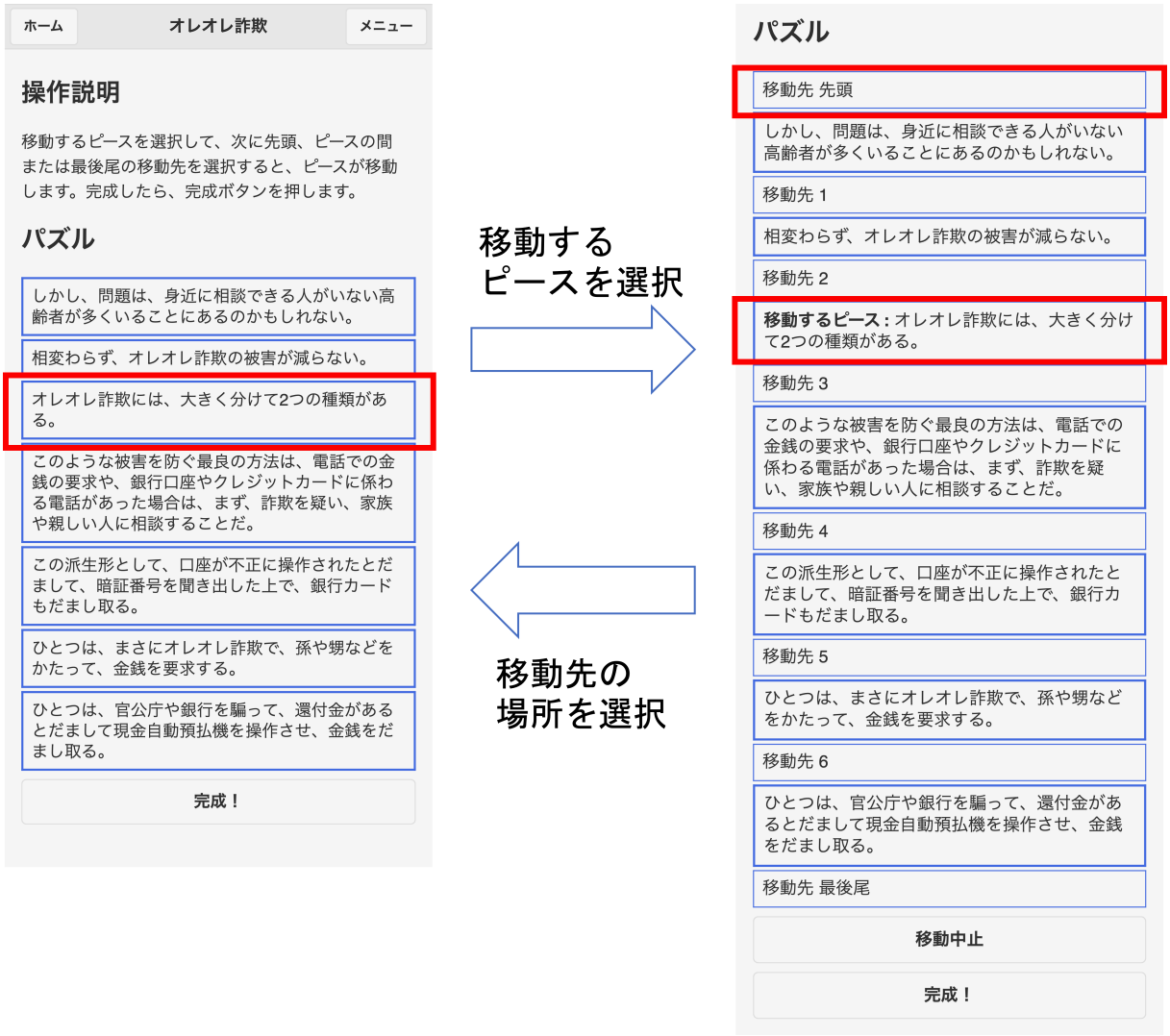



**図97**　タッチ操作でピースを並べ替えるWebアプリ、スクリーン・リーダーで操作できる「ジグソー・テキスト3」

ジグソー・テキストはユーザー操作を記録している。いつ・どのピースをドラッグ／移動開始した、いつ・どのピースをドロップ／移動完了した、完成ボタンを押した、などが記録される。本研究では、このデータを分析する。

**図98**は、スクリーンリーダーでの操作を簡単にするために、画面を真っ黒にする図1の目隠しモードで、「ブラックアウト」と呼ぶ。パズルそのものに至るまでの画面は文字を表示して、パズル画面だけ文字が視覚で読めない。それでも、自分が画面のどこをタッチしているかは分かり、操作の敷居が低くなると期待できる。

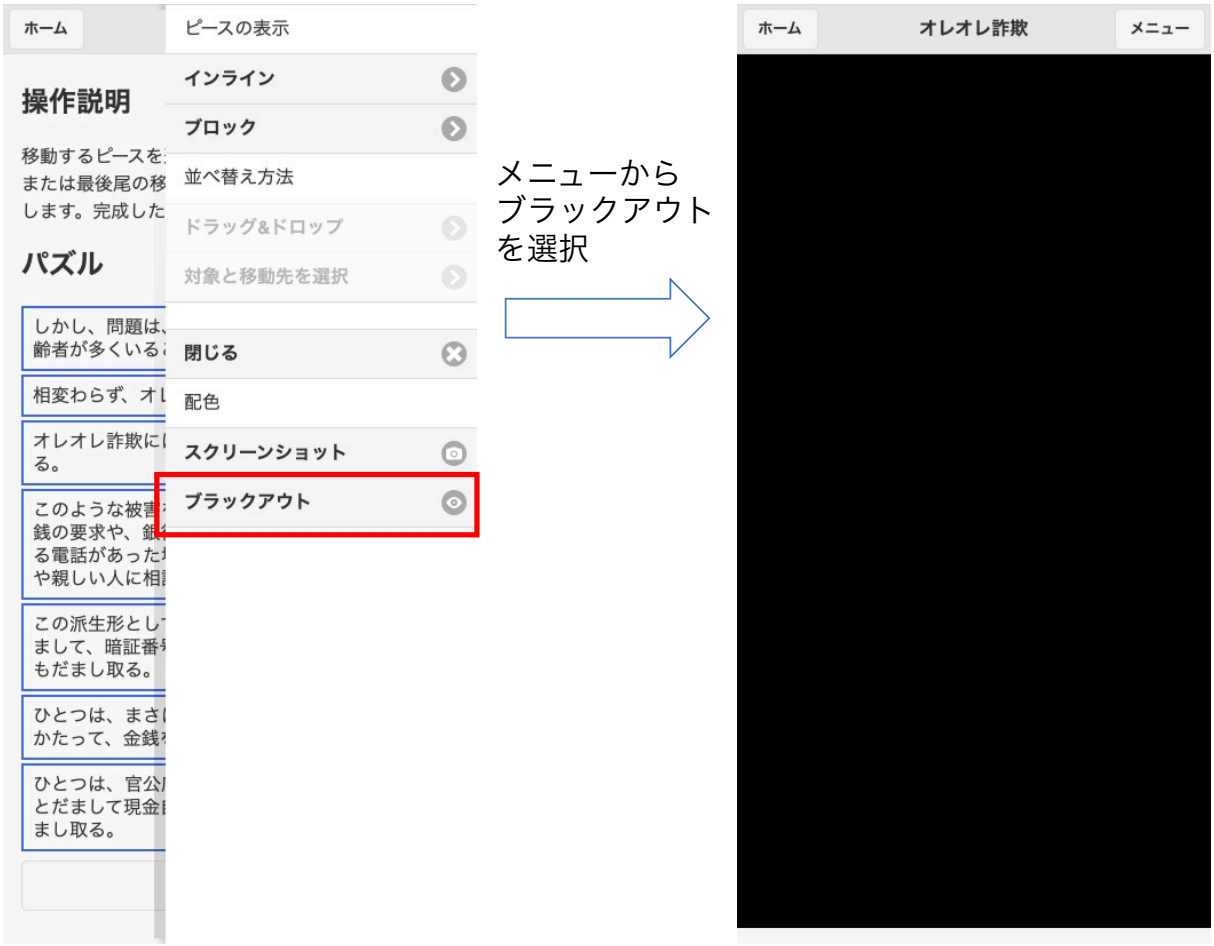



**図98**　画面を真っ黒にすることで目隠し不要でスクリーン・リーダーで操作

### 10.3.2　実験の手順

実験はiPhoneのスクリーン・リーダーVoiceOverを使う。手順は次の通り、練習フェーズを経て実験フェーズに進む:

**練習フェーズ**

- (1) ジグソー・テキスト3に慣れる。
  ドラッグ&ドロップではない操作を覚えて慣れる。
- (2) VoiceOverを覚える。
  実験のための最低限の操作として、項目を読み上げる「タッチ」、アクティベート(通常のタップに相当)の「ダブルタップ」、次の項目に移動する「1本指で右にスワイプ」、前の項目に移動する「1本指で左にスワイプ」、下にスクロールする「3本指で上にスワイプ」、上にスクロールする「3本指で下にスワイプ」を覚える。
  また、VoiceOverのオン/オフを簡単に切り替えるために、ショートカットを確認して、サイドボタンのトリプルクリックをVoiceOverに割り当てておく。

- (3) ピースを見ながらVoiceOverでジグソー・テキスト3の問題を解く。
  ピースを見ながらVoiceOverで操作して数の並べ替え問題を解く。
- (4) ブラックアウトにしてVoiceOverでジグソー・テキスト3の問題を解く。
  図98のブラックアウト状態にして数の並べ替え問題を解く。

**実験フェーズ**

- (5) 文章の並べ替え問題を7問解く。7問は次の通り:
  - 1-1 [実験]
  - 1-2 [電車]
  - 1-3 [マイナスイオン]
  - 2-1 [とき]
  - 2-2 [J-POP]
  - 2-3 [地球温暖化]
  - 4-2 [オレオレ詐欺]

実験は、本研究に通じた2人の被験者に対して行われた。

### 10.3.3 分析方針

一つ一つの操作についてどれくらい考えたかの観点から、操作間隔の時間の分布を分析することにする。操作間隔の時間の長さはどれくらい違うのか。どのタイミングでどれくらい考えるのかが、操作時間の配分に現れると想定する。

## 10.4 結果

### 10.4.1 操作間隔のデータ

**表18**にGUIで視覚的に解いたときの、**表19**にVoiceOverで解いたときの、操作間隔の平均値などを示す。VoiceOverで解いた場合、GUIよりも1桁、所要時間が長い。操作に慣れたり、VoiceOverの読み上げ速度を速くできるようになると、もう少し時間の差が縮まるかもしれない。

**表18**　問題「4-2 [オレオレ詐欺]」を目で見て解いたときの、ピースを動かすまでの所要時間

| ID | 合計所要時間(秒) | 回数 | 中央値(秒) | 平均所要時間(秒) |
|---|---|---|---|---|
| s2 | 129.3 | 21 | 5.2 | 6.2 |
| s3 | 75.5 | 20 | 2.4 | 3.8 |
| s4 | 83.4 | 22 | 2.4 | 3.8 |
| s5 | 150.5 | 25 | 2.1 | 6 |
| s6 | 230.4 | 30 | 2.1 | 7.7 |
| s7 | 167.5 | 22 | 5.6 | 7.6 |
| s8 | 50.5 | 18 | 1.3 | 2.8 |
| 合計 | | 158 | | |

**表19**　問題「4-2 [オレオレ詐欺]」をVoiceOverで解いたときの、ピースを動かすまでの所要時間

| ID | 合計(秒) | 回数 | 中央値(秒) | 平均(秒) |
|---|---|---|---|---|
| s1 | 66.4 | 2 | 33.2 | 33.2 |
| s2 | 0 | 0 | - | 0 |
| s3 | 49.4 | 2 | 24.7 | 24.7 |
| s4 | 58.4 | 3 | 20.1 | 19.5 |
| s5 | 15.7 | 1 | 15.7 | 15.7 |
| s6 | 5.8 | 1 | 5.8 | 5.8 |
| s7 | 0 | 0 | - | 0 |
| s8 | 0 | 0 | - | 0 |
| 合計 | | 9 | | |

### 10.4.2　並べ替え操作の測定結果と分析

パズルを開始してからの経過時間と操作間隔との関係を、GUIとVoiceOverとで比較する。前節で見た通り両者で所要時間のスケールが1桁異なることもあり、全体の時間における個々の操作間隔の比率で比較する。

パズル開始の時刻を$t_0$、n番目の操作をした時刻を$t_n$、完成した時刻を$t_e$、$t_n$時点までの経過時間e = $t_n - t_0$、次の操作までの操作間隔i = $t_{n+1}$ - $t_n$、解くのに要した時間T = $t_e$ - $t_0$とする(**図99**)。完成とは図96で「完成!」ボタンを押した時刻である。図98のUIにも同様のボタンがある。各操作間隔の割合を見るために、$e$と$i$のそれぞれを$T$で割った$e/T$と$i/T$の関係を散布図で示す。

最初の操作は$t_0$なので、最初の操作に対応する点は縦軸の目盛り上に位置する。経過時間の合計は1になるので、操作に対応する点は、縦軸の1と横軸の1を結ぶ直線より上には出ない(**図103**)。各操作に比較的均等に時間を使った場合、散布図は**図100**のようになる。解き始める前に

考えるなどして最初の操作に時間をかけた場合、散布図は**図101**のようになる。最後の並べ替え操作の後、完成までに時間をかけた場合、散布図は**図102**のようになる。

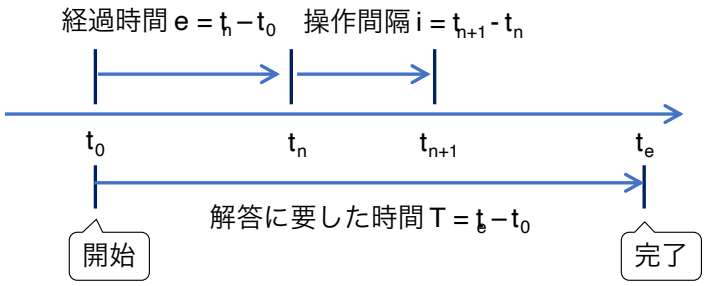



**図99**　経過時間と操作間隔

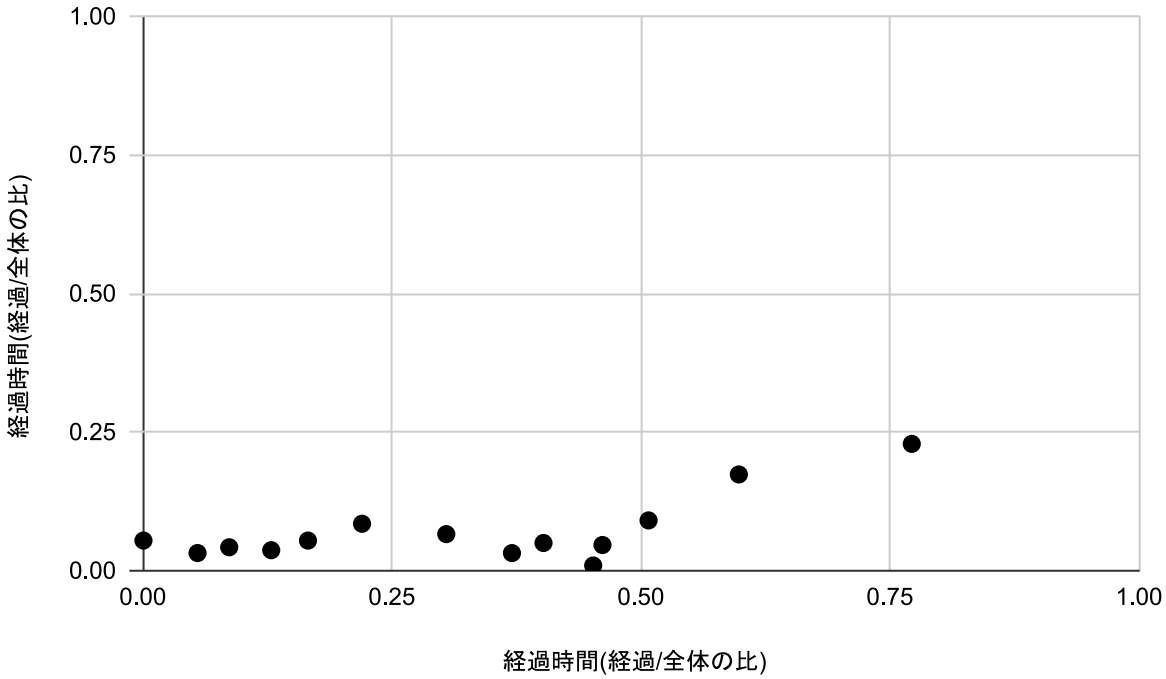



**図100**　各操作に比較的均等に時間を使ったプロセスの散布図

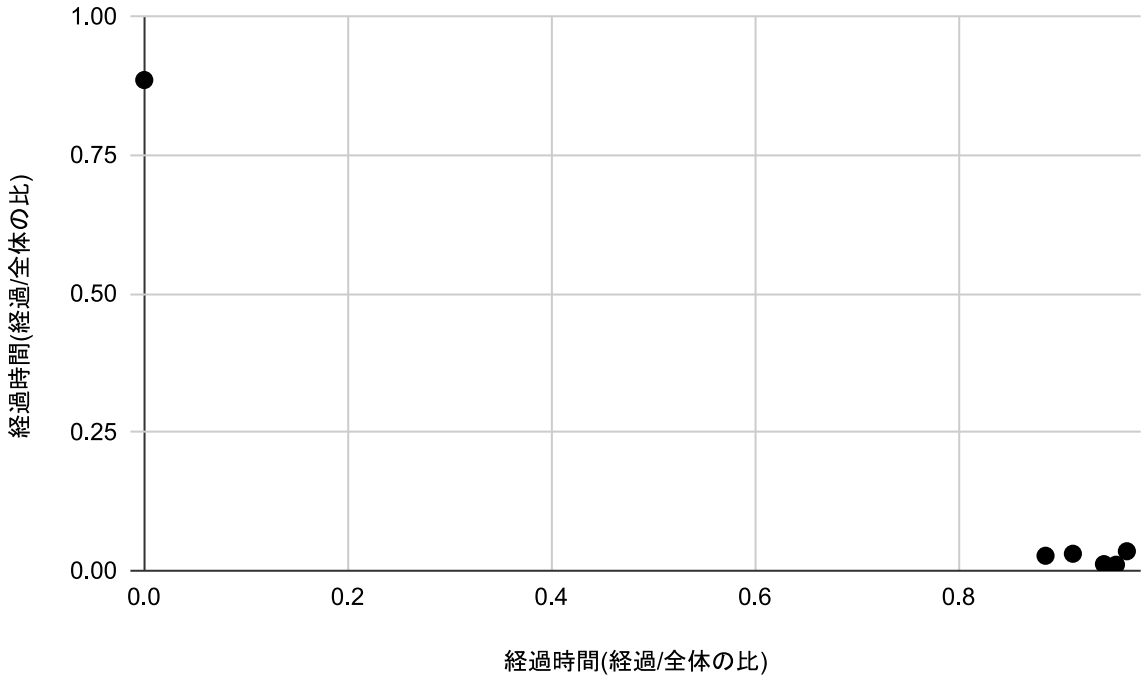



**図101**　最初に時間をかけたプロセスの散布図。最初の操作に全体の89%の時間を使っている

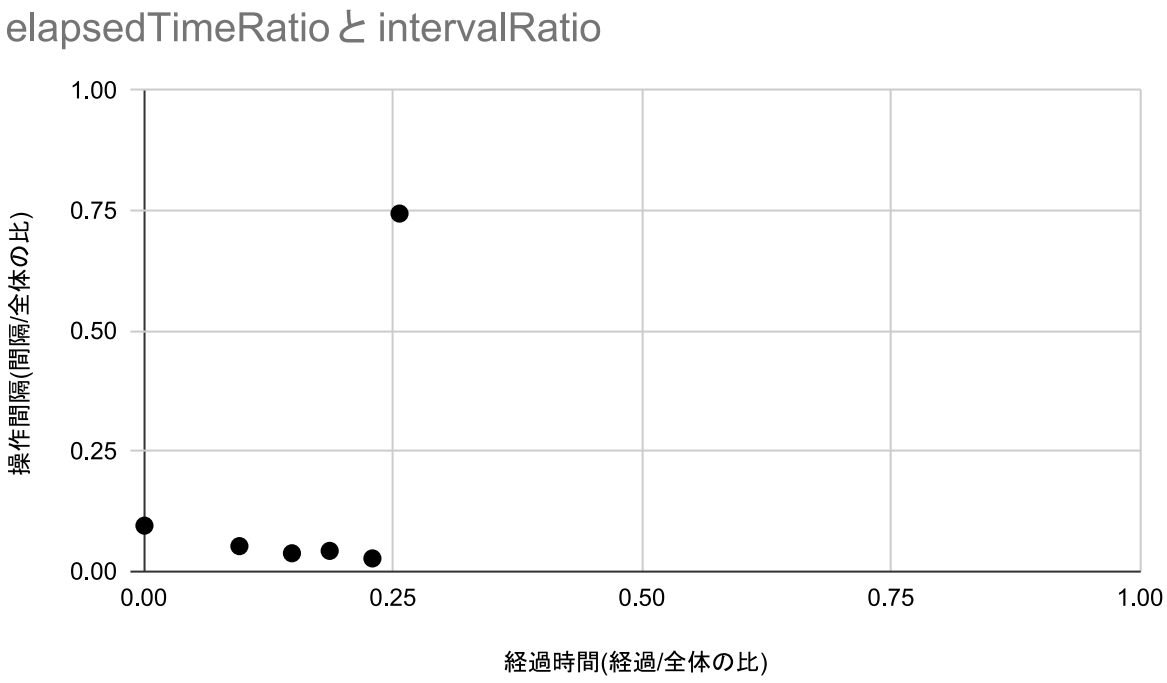



**図102**　最後の並べ替え操作の後、完成までに時間をかけたプロセスの散布図。最後の操作の後、完成までに全体の74%の時間を使っている

経過時間と操作間隔との関係を、GUIで解いたときの各操作について図103に、VoiceOverで解いたときについて**図104**に示す。それぞれは最初の問題をのぞく全問題を解いた全員の全操作をプロットしたものである。最初の問題は、特にVoiceOverで解く場合について、まだ解くのに慣れないと考えられるため除いた。縦軸の0.0から1.0の間と、縦軸1.0と横軸1.0を結ぶ直線とを通る直線で傾向線を引いた。

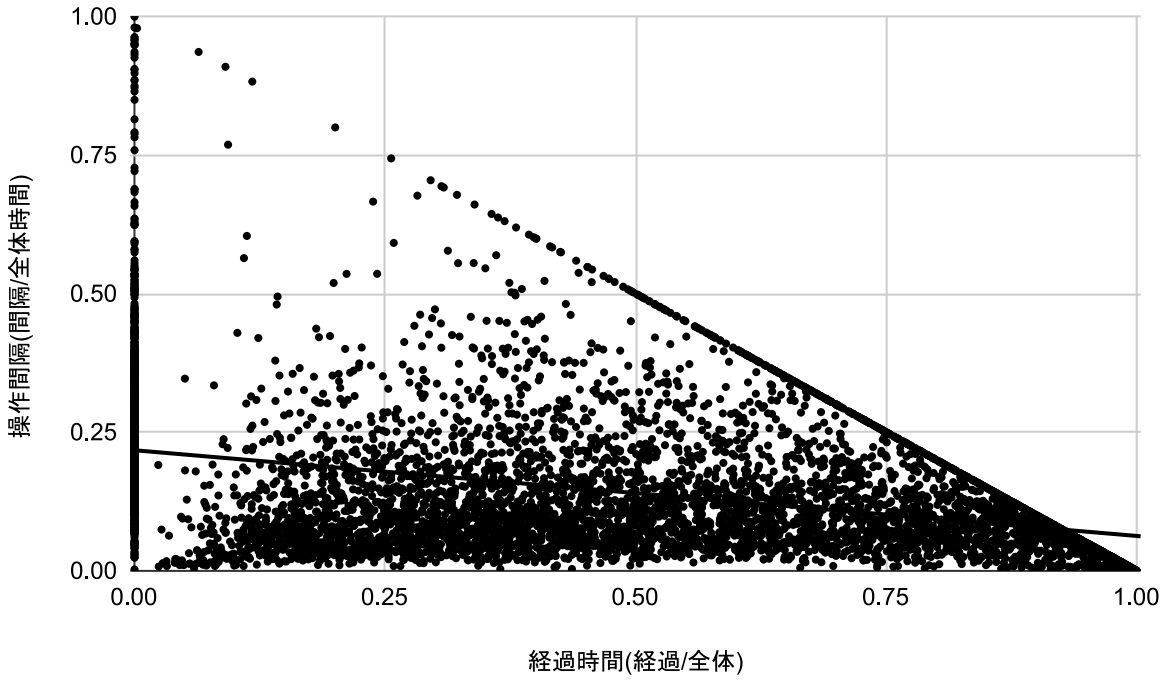



**図103**　目視で解いたときの、経過時間(横軸)と次の操作との間隔(縦軸)の散布図。最初の問題をのぞく全問題を全員(n=222)が解いたときの全操作。横軸の0点には最初の操作が、縦軸1.0と横軸1.0を結ぶ直線上に最後の操作がプロットされる。

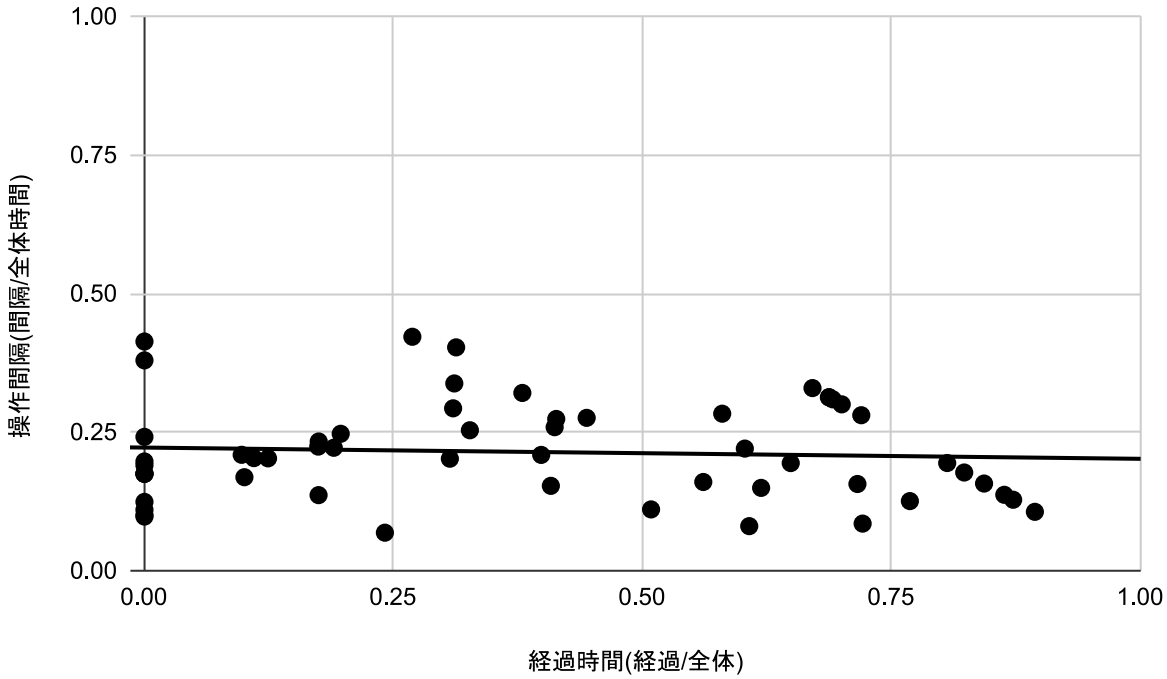



**図104**　VoiceOverで解いたときの散布図。最初の問題をのぞく全問題を全員(n=2)が解いたときの全操作

GUIで解いた図103場合に比べて、VoiceOverで解いた図104は、横軸=0上の点が0.50よりも小さい。また、縦軸1.0と横軸1.0を結ぶ直線上の点も、縦軸の値が0.50よりも小さい。最初の操作や、完成までにかける時間は、GUIの方がVoiceOverよりも多い傾向がある。両者の傾向線も、GUIの方がVoiceOverよりも右肩下がりの傾きが大きく、最初の操作にかける時間が長いことを示している。

## 10.5　考察

### 10.5.1　並べ替え操作の測定結果の分析

以上から、目視とVoiceOverのそれぞれについて、次のように解いた可能性がある:

- 視覚で解いた場合、ピースを動かし始める前に目視で正解の順序が分かってから、ピースを上から順に配置した。
- VoiceOverで聞きながら解いた場合、ピースを並べながら正解を探り、完成に近づくとVoiceOverに全文を読ませて確認する。

今後はこれを仮説として検証する。

### 10.5.2 被験者の振り返りから

被験者の感想をあげて考察する。

**「VoiceOverで解くのは難しい」**
普段は目で文章を読んで考えている者として、VoiceOverで読み上げられた断片から文章を組み立てるのは難しかった。
練習フェーズでは数を並べ替えた。これは数の並べ替えが簡単だとの前提で練習題材に選んだものだったが、やってみると、1桁ならともかく、2桁になると、もう難しかった。
これが本質的な違いであるならば、目で読むのに比べて聞いて読むのは、整合性チェックなどの点で不利なのかもしれない。今後の検証が必要である。
数の並べ替えについては、人による並べ替えとコンピューターのアルゴリズムによる並べ替え(整列)とを比較したことがある**[91][92]**。人による並べ替えについてはジグソー・テキストを使って、目で見て並べ替えた。このときは、人とコンピューターとで明確な類似性を見いだせなかった。今回の結果を踏まえると、人がVoiceOverによって並べ替える場合は、コンピューターのアルゴリズムに近くなるかもしれない。VoiceOverの場合は容易に全体を見渡せないからである。

**「見ると聞くとで違う」**
目で見て読んで違和感のない文章でも、聞いてみると違和感のある問題があった。これはVoiceOverの読み上げ方の問題かもしれないし、それとは別に見るのと聞くのとの違いがあるのかもしれない。これについては今後、問題やアプリーケーションを変えるなどして実験し、印象の違いを明確にしたい。

**「読み間違い、情報が落ちている」**
VoiceOver特有の読み間違いがあり、慣れていないと混乱する。また「とき」ではなく「『とき』」と書いたりする強調表現は、通常設定の読み上げでは無視される。この点は目視に比べてVoiceOverは不利である。問題作成でも考慮する必要がある。

## 10.6 おわりに

### 10.6.1 結論

文章の並べ替え問題を、目で見て読んで解く場合と、スクリーンリーダーの読み上げを聞いて解く場合とで比較する予備実験を行った。その結果、GUIの方がVoiceOverよりも、最初の操作にかける時間が長い傾向が見られた。
今回の限界としては、被験者220名と2名との比較であった。図104で横軸=0上の点の値が0.50より小さいのは、単にデータの数が少ないからの可能性がある。2名は少ないので、より多くの被験者で検証が必要である。

ドラッグ＆ドロップで並べ替える視覚前提のUIでは、図97の「オレオレ詐欺」の場合、ユーザーが「2つある…」「ひとつは…」「ひとつは…」に着目する傾向が見られていた。本章は、このような考え方の傾向をGUIとVUIとで比較して違いを明らかにしようとするものであった。例えば、全文がスマホ1画面に収まるような問題では、単なる所要時間の差を超えて考え方に違いが出るかもしれない。その違いが決定的なものとなったとき、UIが違う場合に同じ問題を解いていると言えるのか？という疑義を提示することになる。

この章で分析したデータは、他の章での分析と同じであり、またVUI用の調整など行っていない。これは、測定・分析の対象がピースだから実現できたと言える。すなわち、3.1.1節で述べた「UIオブジェクトの設計」の効果である。これが、例えば「ユーザーの視線を測定してユーザーの読みを分析する」手法であれば、VUIではデータを得られず比較できない。簡単ではあるが、新しいアプリケーションを提案する研究が、障害のある人々にも利益をもたらす道筋を示せた。

### 10.6.2 将来

本章は、パズル問題を解くときに、視覚的には同じUIに、GUIと読み上げ(VoiceOver)とでアクセスしたときに、同じ考え方をするか？を検証した。

逆に、視覚的には異なるUIだが、GUIと読み上げとで同じ情報を伝えられるか？というアプローチも進めている。

**図105**の散布図に対し、散布図の各点(プロット)の座標を読み上げるといった方法は、実装可能であろう。しかし、数百の点を読み上げられても、散布図が示す相関関係を読み取るのは困難であろう。

そこで、**図106**は、散布図の代替表現として提案するものである。領域を2次元の区間でメッシュ化し、各セル内の点の数で表を構成する。区間数を対話的に変えて(図ではスライダーを操作して)、表を読み上げることで点の分布を把握する。読み取った相関だけでなく、メッシュの変化などから相関を読み取るプロセスを分析する。

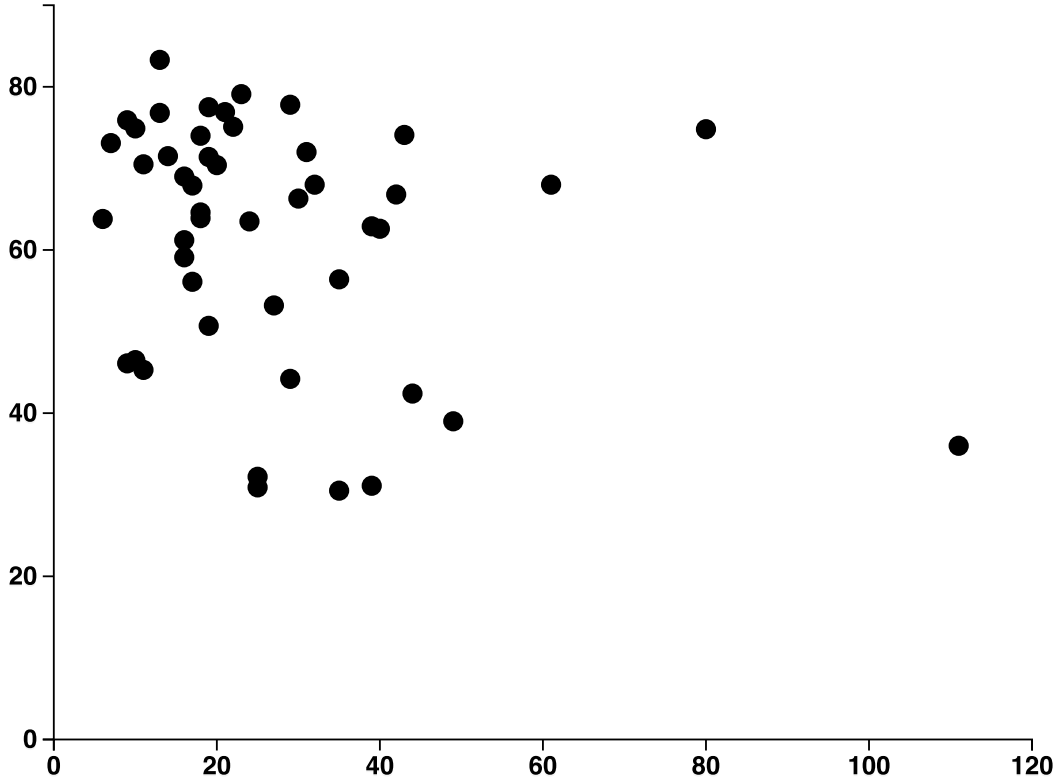



**図105**　散布図

データの出典：政府統計の総合窓口(e-Stat)

5

| | | | | | | |
|---|---|---|---|---|---|---|
| [80, 96) | 1 | 0 | 0 | 0 | 0 | 0 |
| [64, 80) | 14 | 8 | 1 | 1 | 0 | 0 |
| [48, 64) | 6 | 5 | 0 | 0 | 0 | 0 |
| [32, 48) | 3 | 2 | 2 | 0 | 0 | 1 |
| [16, 32) | 0 | 3 | 0 | 0 | 0 | 0 |
| [0, 16) | 0 | 0 | 0 | 0 | 0 | 0 |
| | [0, 22) | [22, 44) | [44, 66) | [66, 88) | [88, 110) | [110, 132) |

セルの数: 36, 最大: 14, 平均: 1.3, 分散: 8, 標準偏差: 2.8,

1シグマ区間の外、すなわち、平均 + 標準偏差 = 4.1 <

2シグマ区間の外、すなわち、平均 + 標準偏差 x2 = 6.9 <

**図106**　散布図の代替表現

データの出典：政府統計の総合窓口(e-Stat)

両者は視覚的には異なるが、分析によって同じ情報を伝えることを示せれば、後者は代替表現として成立すると言えるだろう。

### 10.6.3 意義

この研究の意義は、スクリーン・リーダーによるプレイを、GUIによるものと同様に測定し、GUIによるプレイと同じ分析手法を適用したことである。

本研究に限らず、ある学習分析の研究が学習者に恩恵をもたらすのであれば、その成果が障害者にも同様に恩恵をもたらすように、研究が進んでいくべきであろう。それは別の研究でよく、別の研究者によるものでもよい。クラスに障害者がいても、異なるUIかもしれないが、同じ問題を解いて、同じ手法で分析・評価できる。もし、考え方の違いが発見されて、それがその問題にとって本質的な違いであれば、その問題そのものを修正する必要があるかもしれない。

# 11 考察

ここでは、本研究で開発したアプリケーションを使った事例、編集操作指標EOI、長い文章の時間的な共起行列、本研究の考察とする。

## 11.1 Topic Writer、ジグソー・テキスト、ジグソー・コードの事例

ジグソー・テキストやジグソー・コードは、アプリケーションの仕組みは単なる取捨選択・並べ替え処理である。しかし、そこに乗せるパズル問題によって、様々な演習に適用できた。

### 11.1.1 テクニカル・ライティング演習

Topic Writerはテクニカル・ライティングの授業で、文章のロジック・ツリーを学ぶ題材として使われた[17][18][19][20][21]**[32]**。

ジグソー・テキストのパズル問題「オレオレ詐欺」(図12)はテクニカル・ライティングの授業で、正解を問う使い方ではなく、考えて言語化する題材として使われた。受講生がそれぞれ並べ替えて、数人のグループになって、自分がそのように並べた理由や途中で考えたことを発表し合う。自分の考えを言語化したり、他の人の考えを聞くことに学習効果を求めている。

### 11.1.2 就活の自己紹介書

就職活動(就活)の自己紹介書にTopic Writerを導入した事例がある[32]。**図107**のようなワークシートである。


**図107** Topic Writerによる自己紹介書の画面

自己紹介を効果的なものにするためには、志望企業の特性や職種に応じた内容としなくてはならない。志望企業の業種や希望する職種の特徴などは自己紹介書に含まないが、考慮すべき内容である。そこで、自己紹介書ワークシート(ロジック・ツリー)には、それを明示的に書かせるよ

うにした、これがこの研究の特徴である**図108**。すなわち、文章には書かれない事柄(業種や職種)と、文章の内容(自己紹介の内容)との一貫性である。

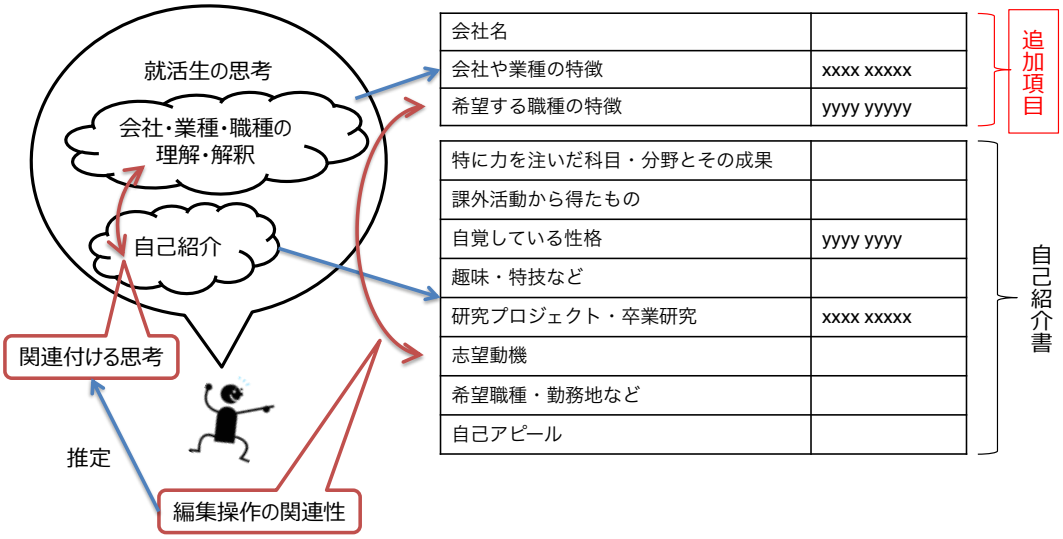



**図108**　自己紹介書の狙い

このワークシートを使ってTopic Writerで書いたときの時間的な共起行列の例を図??に示す。図で、赤い線や青い線で囲ったセルが「最終的な自己紹介書には出力しないが、一貫性をもたせるべき項目」と、自己紹介の内容との共起に該当する。この部分と自己紹介の部分とを交互に編集すると、これらセルの共起頻度が高くなる。

共起行列の集計

| n \ n+1 | cc_37 | cc_47 | cc_50 | cc_13 | cc_16 | cc_21 | cc_24 | cc_27 | cc_30 | cc_33 | cc_44 | other |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| cc_37 | 17 | 10 | 1 | 4 | 3 | 1 | 0 | 0 | 1 | 2 | 0 | 1 |
| cc_47 | 1 | 27 | 16 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 |
| cc_50 | 0 | 5 | 18 | 7 | 0 | 2 | 0 | 0 | 5 | 3 | 3 | 1 |
| cc_13 | 1 | 0 | 0 | 15 | 15 | 2 | 2 | 2 | 0 | 0 | 0 | 0 |
| cc_16 | 2 | 1 | 1 | 2 | 13 | 9 | 3 | 4 | 3 | 0 | 0 | 2 |
| cc_21 | 0 | 2 | 1 | 0 | 0 | 19 | 16 | 2 | 1 | 2 | 0 | 0 |
| cc_24 | 1 | 0 | 0 | 1 | 3 | 3 | 15 | 10 | 2 | 1 | 3 | 0 |
| cc_27 | 1 | 0 | 0 | 0 | 2 | 4 | 2 | 18 | 7 | 3 | 4 | 0 |
| cc_30 | 2 | 3 | 1 | 0 | 0 | 1 | 0 | 1 | 26 | 7 | 8 | 0 |
| cc_33 | 2 | 2 | 2 | 1 | 3 | 0 | 1 | 3 | 1 | 19 | 7 | 0 |
| cc_44 | 2 | 1 | 4 | 3 | 0 | 1 | 0 | 1 | 6 | 3 | 16 | 0 |
| other | 2 | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 9 |

**図109**　自己紹介書の時間的な共起行列、教員の個別指導前

### 11.1.3　ロジカル・シンキング

ロジカルシンキングの演習では、「結論」と「根拠」、「目的」と「手段」、「問題」と「原因」と「原因の原因」といった、それぞれに該当するピースを与えて、適切な順序に配置するように問う[93][94]。

**図110**は授業で使われたパズル問題の例である。「【目的】」などは見出しの役割をするピースで、【目的】の次に目的となるピースを配置するように授業では求める。解答は**図111**のようになる。

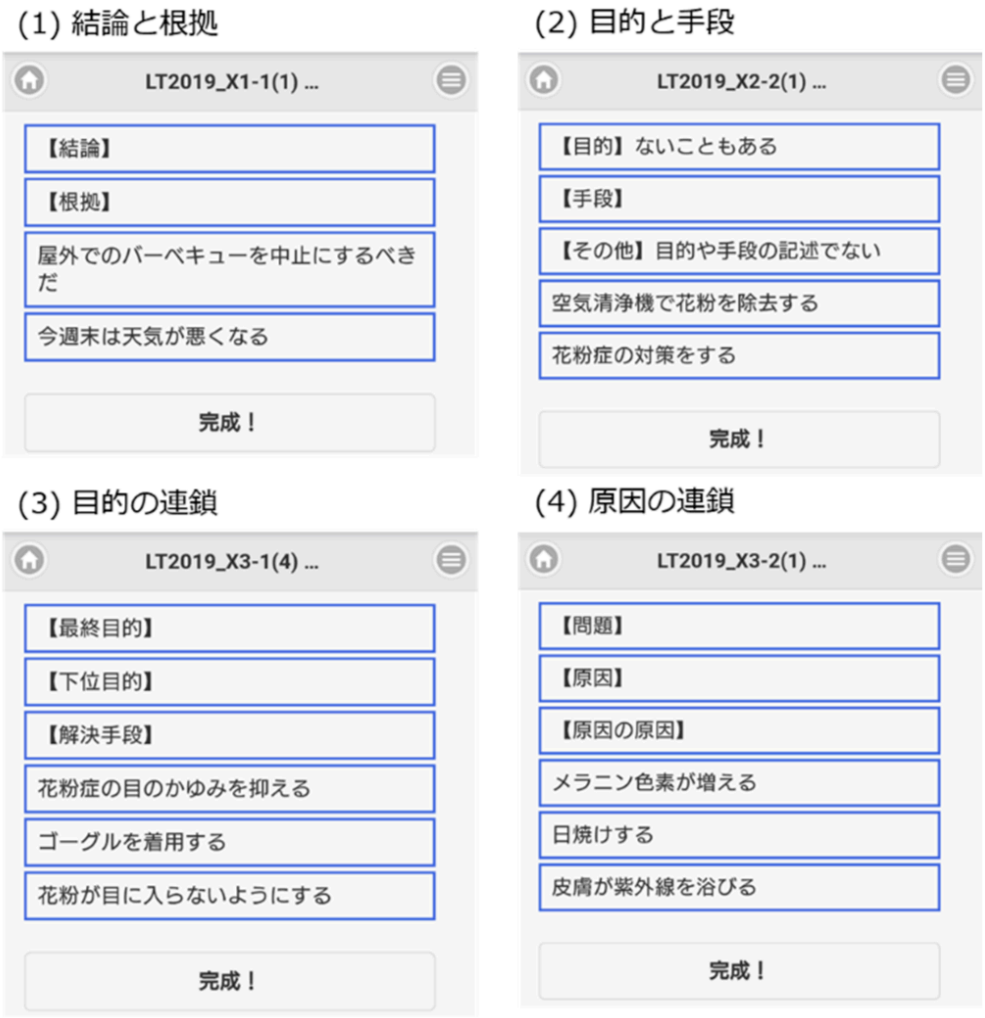


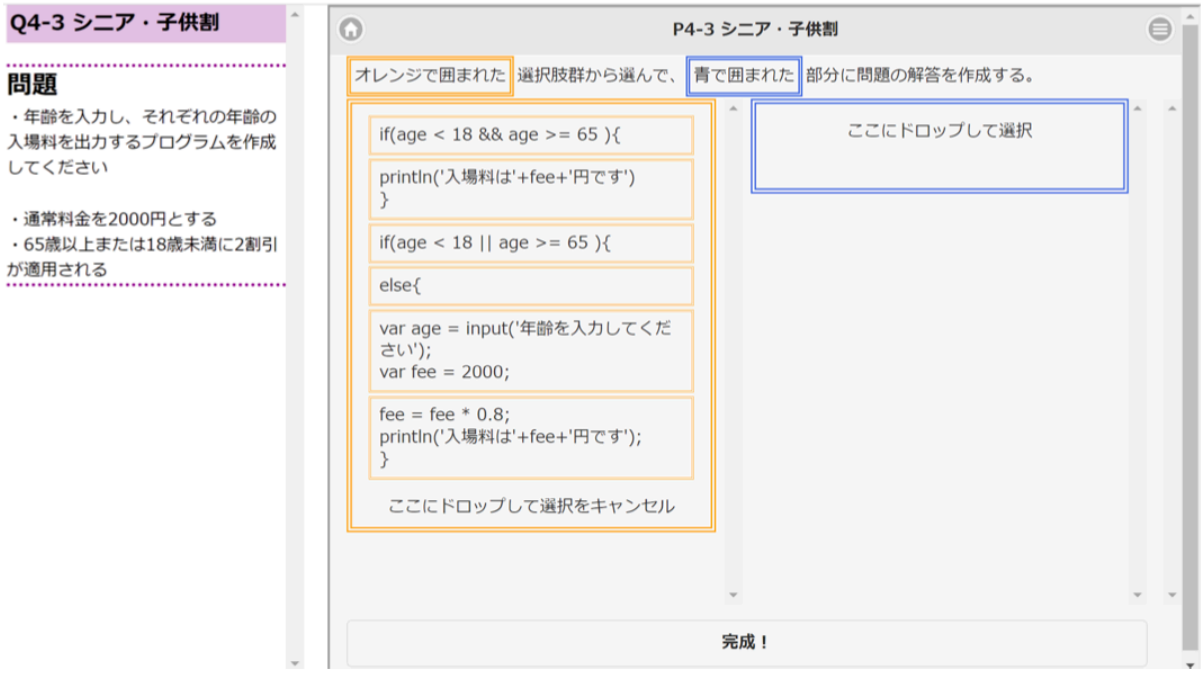

**図110**　ジグソー・テキストによるロジカル・シンキングの演習[94]


**図 111**　解答の例[94]

### 11.1.4　プログラミング演習

5章のプログラミンの演習にジグソー・コードを使った事例では、間違いやすいピースを選択肢に追加し、最後の解答だけでなく、途中でそれら選択肢を試したかどうかを調べた。パズル問題の作り方の一つであり、評価ポイントでもあった。

コンピューター関連の授業でも、問題の内容は、JavaScript といったプログラミング言語の文法を問うだけでなく、プログラミング言語を日本語におきかえた問題(**図 112**)、SQL(**図 113**)、タートルで図形を描く問題(図 114)もあった**[95][96][97][98]**。



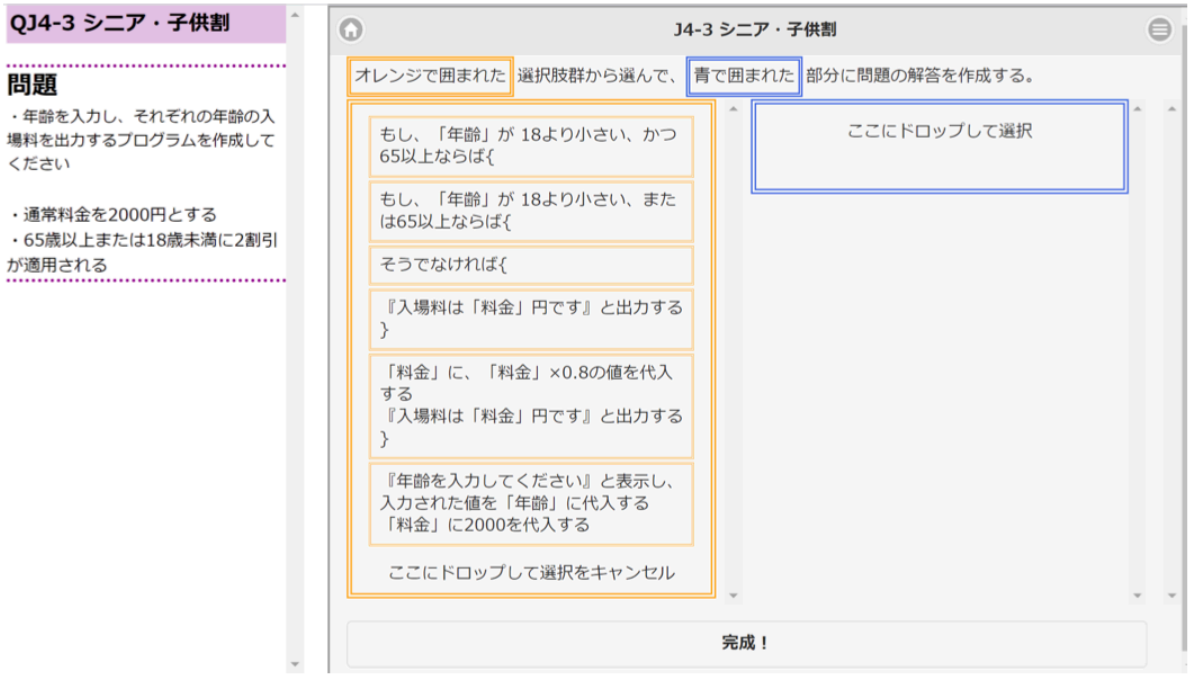




**図112**　プログラミング言語の問題(上)と日本語に翻訳した問題(下)[97]

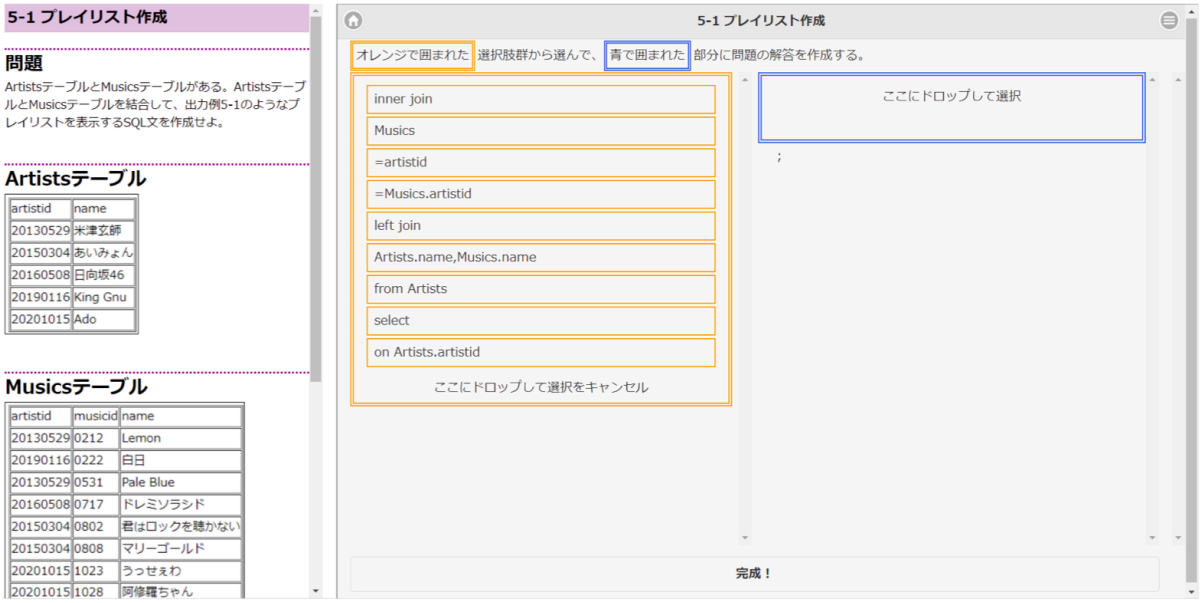



**図113**　SQLの問題[99]

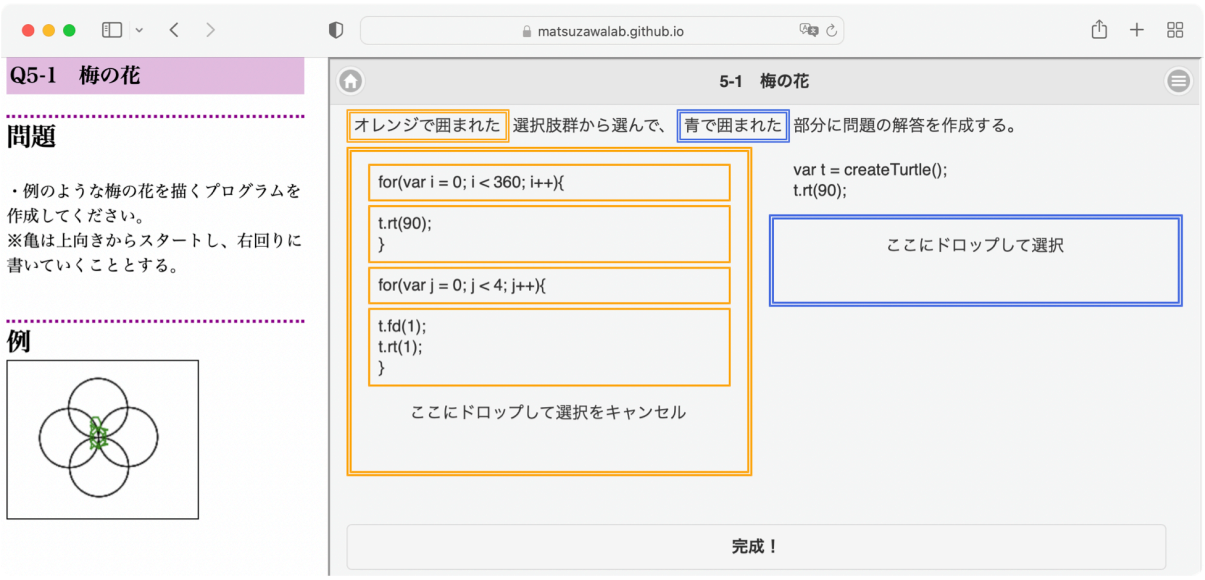



**図114**　タートルで図形を描く問題[96]

これらは、どれも3.2.2節の「システム連携」の仕組みを使って、ジグソー・コードを外部アプリケーションの画面の中に埋め込んでいる。それぞれ、左側の画面が、外部アプリケーションの画面である。ジグソー・コードはプレイのdocument idを外部アプリケーションに送り、外部アプリケーションが受講生とdocument idとを結びつけて保管している。

また、ジグソー・コードのパズル問題を項目反応理論(Item Response Theory, IRT)で分析した研究もあった**[99][100]**。

### 11.1.5　図形の証明問題

高校の数学Aの授業では図形の証明問題を解いた**[101]**。ここでもジグソー・コードをアプリケーションに埋め込んで使っている(**図115**)。問題を工夫することで誤答パターンを見つけられた(図??)。

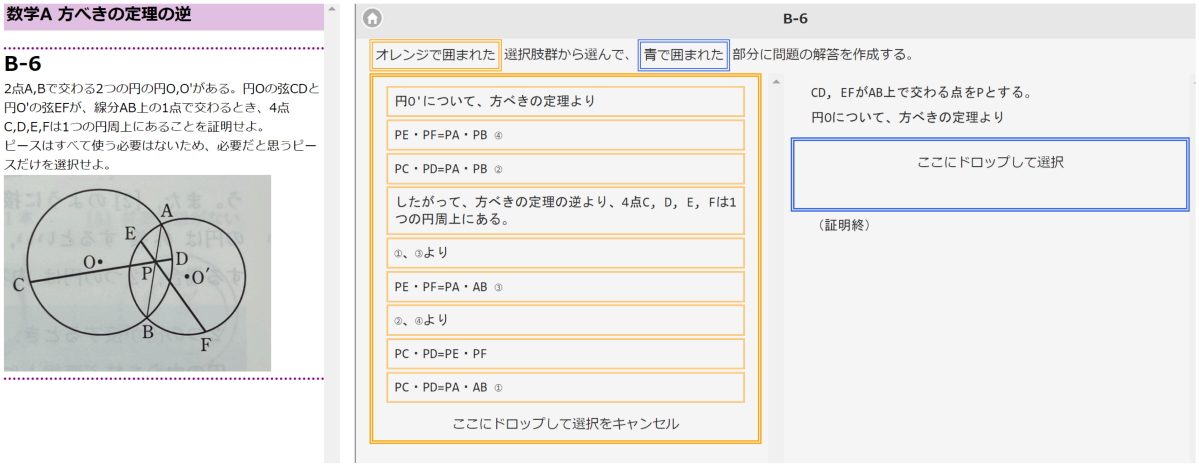



**図115**　図形の証明問題[101]

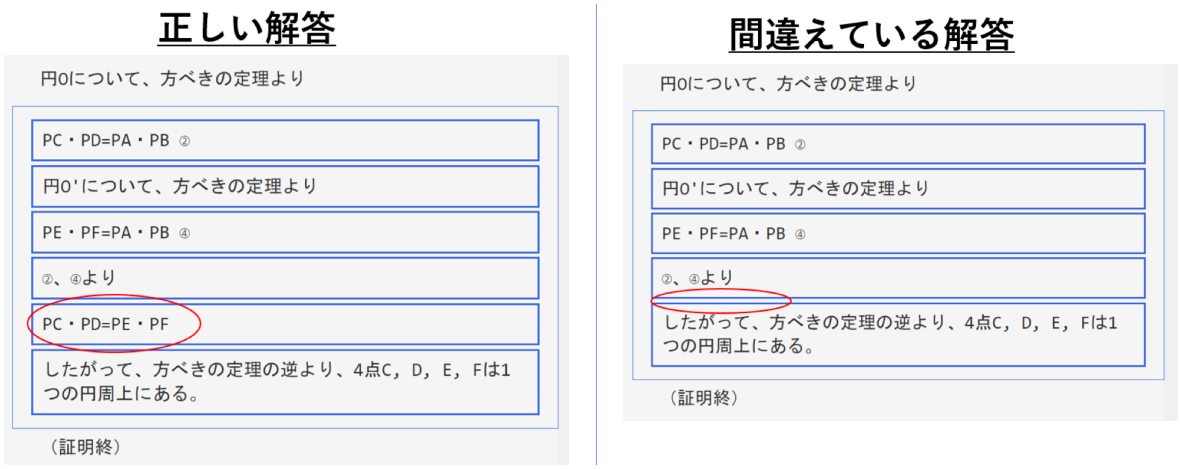



**図116**　論理に飛躍がある解答[101]

教員からは「現場のニーズに応えるものであり生徒が前向きに喜んでいた」、「教員の立場からも指導に活かすことができた」、「ソフト自体の操作性もよく、 集中して取り組んでいた」、「現状のままでも、十分に活用できるソフ トでした」とのコメントをもらえた。

なお、並べ替えによる証明問題の学習教材としては公立はこだて未来大学のPBLに先行研究があり**[102]**、この事例の研究報告でも参照している。

### 11.1.6　プロダクト・オーナーの育成

企業でも採用されている。スクラムを採用したアジャイル開発で、プロダクトの価値を最大化することに責任を持つプロダクト・オーナー(Product Owner、PO)の育成支援のワークショップなどでジグソー・テキストが使われている[58]。7章を参照されたい。

### 11.1.7　データサイエンス演習

以上は先生や実験者が問題を作った事例であるが、受講生自身が問題を作る事例もある。データサイエンス教育として、自分で問題を作り、結果を予想し、互いに問題を解いて、解くプロセスを分析し、予想と突き合わせる。予めよく考えて問題を作る必要があるとか、解くときに考える時間をどれくらい与えればよいか、といった気づきが、受講生にあった。また、作った問題の内容も、県の面積の広い順に並べるといった問題だけでなく、トロッコ問題といった答えのない問題もあったことは注目に値する(**図117、図118**)**[103]**。

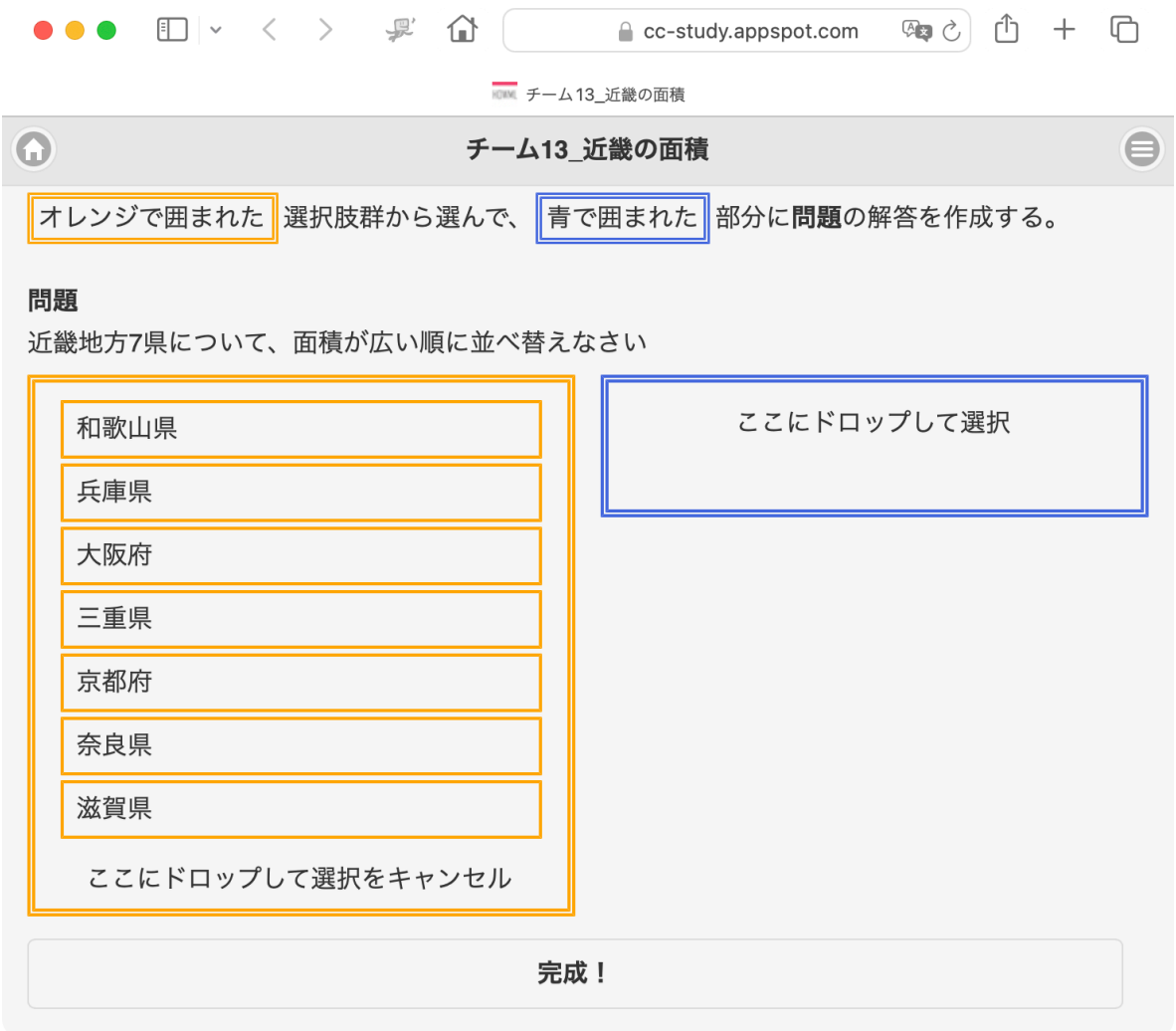



**図117**　「近畿の面積」

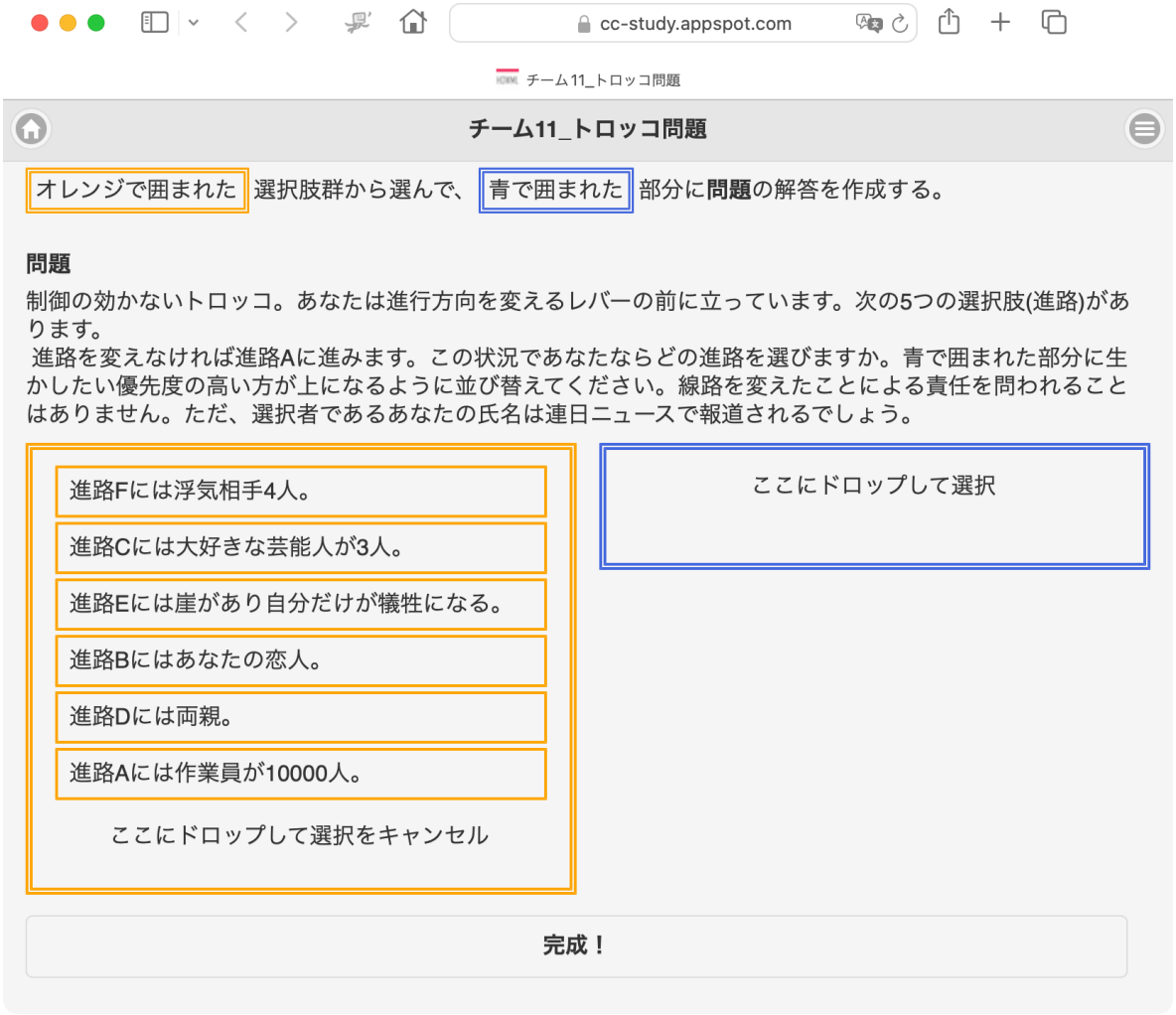



**図118　「トロッコ問題」**

## 11.2　編集操作指標EOI

Topic Writerによるワークシート作文(ロジック・ツリーに従う作文)の時間的な共起行列は、例えば3項目のロジック・ツリーならば**図119**のようになる。3つの項目、cc_8、cc_10、cc_12が画面上の上から下、あるいは左から右に並んでいると、多くの場合、時間的な共起行列は図のようになる。行列の対角線は同じ項目を続けて編集したことに対応し、対角線の右隣は次(下、または右)の項目に移ったことに対応する。それよりも右上のセルは、先へ飛んで編集したことに、左下のセルは戻って編集したことに対応する。同じロジック・ツリーに従って同じ時間に書いても、人によって共起行列は様々となる(**図120**)。しかし、何も指示しないと、対角線および対角線の右隣に共起頻度が集中することが多い。すなわち、人は(日本語で)文章を書こうとすると、上から下、あるいは左から右へ書いていくことが多い。

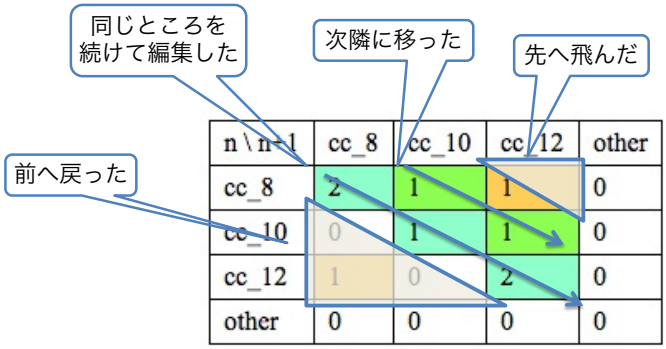



**図119**　Topic Writerにおける編集操作の時間的な共起行列。3項目のロジック・ツリーの場合の例(otherの行列は無視する)。

d466c40c-5b2b-11e4-b87d-09e097bce355

| n \ n+1 | cc_27 | cc_8 | cc_10 | cc_12 | cc_29 | other |
|---|---|---|---|---|---|---|
| cc_27 | 1 | 0 | 0 | 0 | 0 | 1 |
| cc_8 | 0 | 2 | 2 | 0 | 0 | 0 |
| cc_10 | 0 | 0 | 3 | 1 | 1 | 0 |
| cc_12 | 1 | 1 | 0 | 2 | 0 | 0 |
| cc_29 | 0 | 0 | 0 | 1 | 1 | 0 |
| other | 0 | 1 | 0 | 0 | 0 | 12 |

1ff1b1c0-5b31-11e4-ad9c-09e097bce355

| n \ n+1 | cc_27 | cc_8 | cc_10 | cc_12 | cc_29 | other |
|---|---|---|---|---|---|---|
| cc_27 | 1 | 0 | 0 | 1 | 0 | 0 |
| cc_8 | 0 | 2 | 1 | 0 | 0 | 0 |
| cc_10 | 0 | 0 | 4 | 0 | 0 | 0 |
| cc_12 | 0 | 0 | 0 | 1 | 1 | 0 |
| cc_29 | 0 | 1 | 0 | 0 | 1 | 0 |
| other | 0 | 0 | 0 | 0 | 0 | 0 |

7e7613c0-5a8d-11e4-a655-25ddab3704da

| n \ n+1 | cc_27 | cc_8 | cc_10 | cc_12 | cc_29 | other |
|---|---|---|---|---|---|---|
| cc_27 | 2 | 0 | 0 | 0 | 0 | 1 |
| cc_8 | 1 | 2 | 1 | 1 | 0 | 0 |
| cc_10 | 0 | 0 | 1 | 0 | 1 | 0 |
| cc_12 | 0 | 1 | 0 | 1 | 0 | 0 |
| cc_29 | 0 | 1 | 0 | 0 | 1 | 0 |
| other | 0 | 0 | 0 | 0 | 0 | 3 |

c4ee8bf0-5b25-11e4-91d7-09e097bce355

| n \ n+1 | cc_27 | cc_8 | cc_10 | cc_12 | cc_29 | other |
|---|---|---|---|---|---|---|
| cc_27 | 1 | 1 | 0 | 0 | 0 | 0 |
| cc_8 | 0 | 1 | 1 | 1 | 1 | 0 |
| cc_10 | 0 | 0 | 3 | 0 | 0 | 1 |
| cc_12 | 0 | 1 | 0 | 1 | 0 | 0 |
| cc_29 | 0 | 1 | 0 | 0 | 1 | 0 |
| other | 0 | 0 | 0 | 0 | 0 | 6 |

1f2210d1-58cc-11e4-945b-ff2c26b7cf17

| n \ n+1 | cc_27 | cc_8 | cc_10 | cc_12 | cc_29 | other |
|---|---|---|---|---|---|---|
| cc_27 | 0 | 0 | 0 | 0 | 0 | 0 |
| cc_8 | 0 | 1 | 2 | 0 | 0 | 0 |
| cc_10 | 0 | 0 | 0 | 0 | 1 | 2 |
| cc_12 | 0 | 1 | 0 | 0 | 0 | 0 |
| cc_29 | 0 | 1 | 0 | 0 | 0 | 0 |
| other | 0 | 0 | 1 | 1 | 0 | 2 |

724156e8-5a8b-11e4-90f3-25ddab3704da

| n \ n+1 | cc_27 | cc_8 | cc_10 | cc_12 | cc_29 | other |
|---|---|---|---|---|---|---|
| cc_27 | 1 | 0 | 0 | 0 | 1 | 1 |
| cc_8 | 0 | 2 | 1 | 0 | 0 | 1 |
| cc_10 | 0 | 0 | 1 | 1 | 0 | 0 |
| cc_12 | 0 | 1 | 0 | 1 | 0 | 0 |
| cc_29 | 0 | 1 | 0 | 0 | 1 | 0 |
| other | 1 | 1 | 0 | 0 | 0 | 6 |

af36d3d4-5a04-11e4-a664-25ddab3704da

| n \ n+1 | cc_27 | cc_8 | cc_10 | cc_12 | cc_29 | other |
|---|---|---|---|---|---|---|
| cc_27 | 1 | 1 | 0 | 0 | 0 | 0 |
| cc_8 | 0 | 1 | 1 | 0 | 0 | 0 |
| cc_10 | 0 | 0 | 2 | 1 | 0 | 0 |
| cc_12 | 0 | 0 | 0 | 1 | 2 | 0 |
| cc_29 | 0 | 0 | 0 | 1 | 1 | 0 |
| other | 0 | 0 | 0 | 0 | 0 | 0 |

**図120**　様々なTopic Writerにおける時間的な共起行列
対角線に寄ったもの、左下や右上に散らばったもの

ここで、遠く離れた部分同士の一貫性(coherence)を思い出すと、一貫性に配慮して文章を書いた場合は、共起行列の左下や右上に共起頻度が散らばる可能性がある。そこで、対角線に集まった共起行列では値が小さく、左下や右上に散らばった共起行列では値が大きくなるように導入した指標が、編集操作指標(Editing Operation Indicator, EOI)である。**図121**にEOIの計算式を示す**[104]**。このように導入したEOIと、Topic Writerを使って書いたメール文のルーブリックによる評価点との関係を調べたところ、「文章作成力が高い人は作成時に一通り書いてから見直してブラッシュアップしているであろう」ことが示された**[105]**。

$$\mathrm{EOI} = \sum_{i=1,j=1}^{n} w_{ij}\, m_{ij}$$

$$w_{ij} = \begin{cases} 2(j-(i-1)-1), i < j \\ 0, i = j \\ 2(i-(j-1)-1), i > j \end{cases}$$

$$w_{ij} = \begin{pmatrix} 0 & 1 & 3 & 5 & & & & 2(j-1)-1 \\ 2 & 0 & 1 & 3 & & & & \\ 4 & 2 & 0 & 1 & & & & \\ 6 & 4 & 2 & 0 & & & & \\ & & & & \ddots & & & \\ & & & & & \ddots & & 5 \\ & & & & & & \ddots & 3 \\ & & & & & & & 1 \\ 2(i-(j-1)-1) & & & & 6 & 4 & 2 & 0 \end{pmatrix}$$

**図121**　EOIの計算式

### 11.3　長い文章の時間的な共起行列

Topic Writerを使う事例では、ロジック・ツリー(ワークシート)の項目は数個であることが多い。では、長い文章の場合、Topic Writer同様に編集操作を測定したら、時間的な共起行列はどのようになるだろうか。この節では、そのような例を示す。

**図122**は、研究発表会の原稿・論文の編集プロセスを、Topic Writer同様に測定して時間的な共起行列に集計する概念を示す。Topic Writerとことなり、個別の段落に固有の意味付けはしていない。図の右側の行列が時間的な共起行列で、行列が見出しや段落に該当すると思ってよい。これらは最終的に残った見出しや段落が、文章中で占める位置の順番に、行列に並んでいる。途中で削除されたものは、操作は記録されているが、並び順が不定である。

このような共起行列の共起頻度を円の大きさで示して散布図に仕立てたのが**図123**である。青い丸は対角線、すなわち同じ段落を続けて編集したことに該当する。2つの文章の時間的な共起行列が載っている。上下2つの共起行列を比べてみると、上は対角線に集まっていて、下は散らばってる。何らかの元となる文章があって、それをアレンジするような編集をすると、上のような共起行列となる傾向がある。

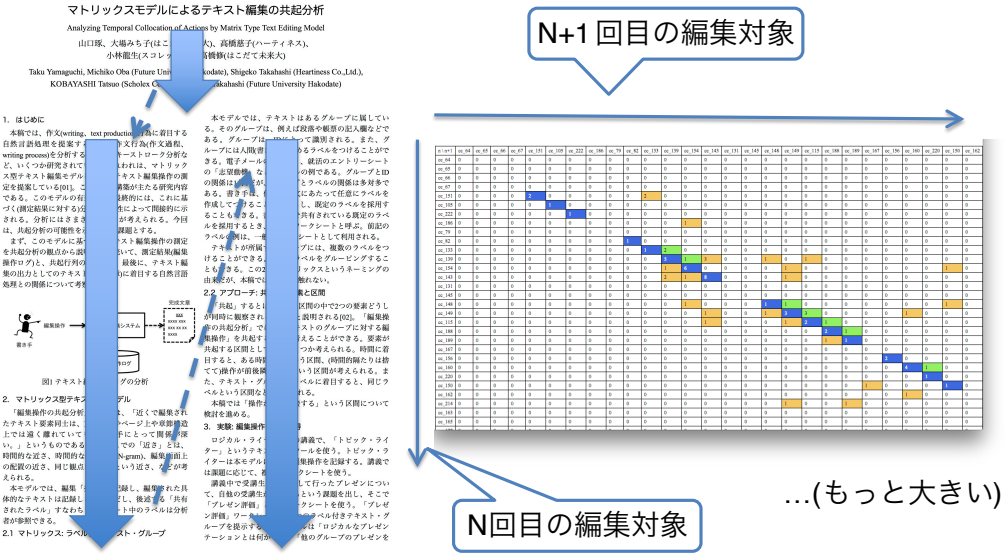



**図122**　長い文章の編集操作の時間的な共起行列

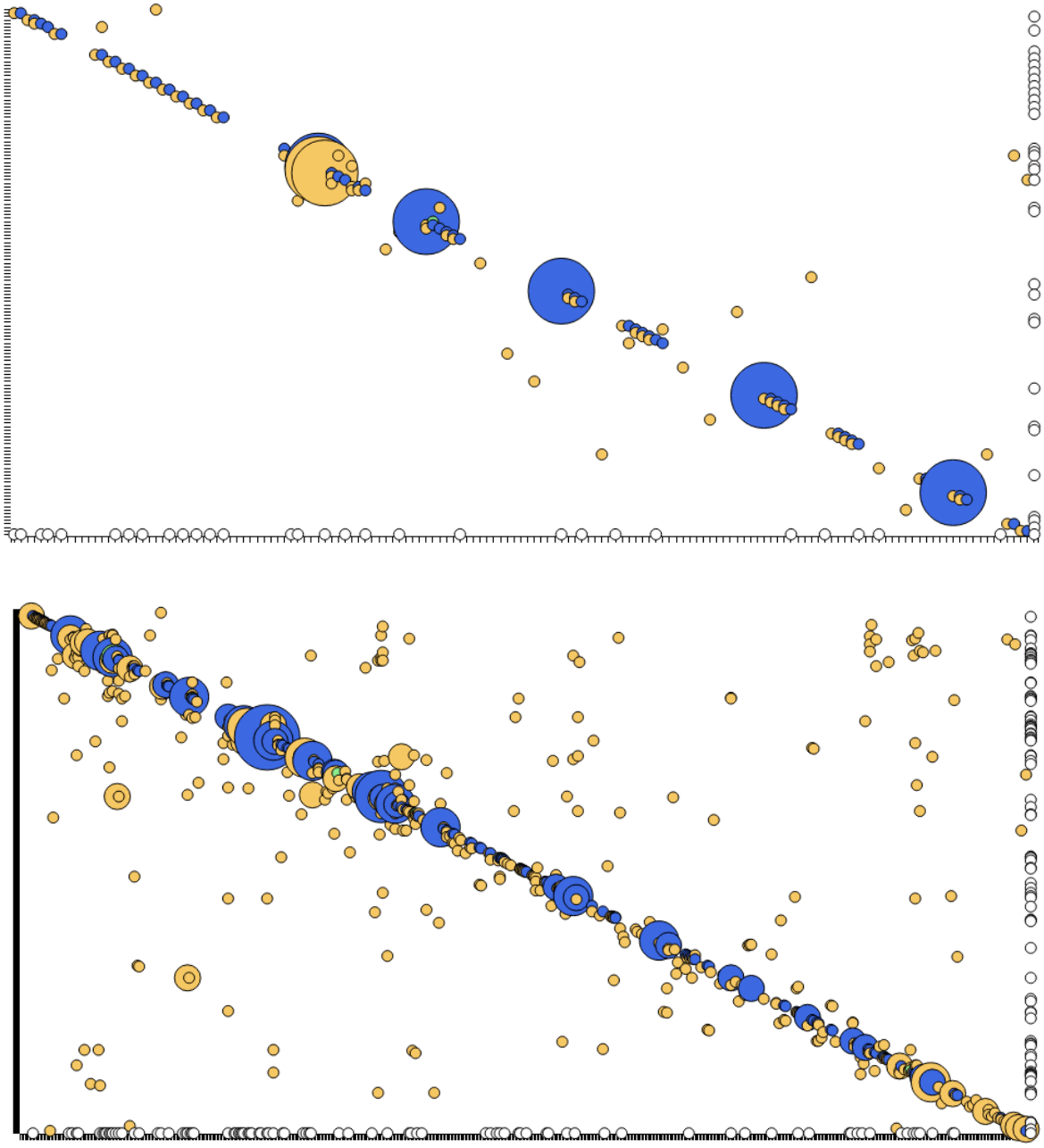


**図 123**　長い文章の時間的な共起行列の 2 つの例
共起頻度の大きさを円の大きさで表現している

長い文章を長い時間かけて編集するプロセスの分析は興味深い。しかし、Topic Writer(の事例)と比べて、長い文章を安定して編集できて、本研究が提案する性質を備えたアプリケーションを用意する必要がある。今後の課題である。

### 11.4　コンピューターによる解くプロセスとの比較分析

学習アプリケーションをうまく設計すれば、学習者の考え方が操作のパターンに表れると、我々は期待している。例えば、並べ替えプログラミングの操作を測定・分析して、正解・不正解

にかかわらず学習者の解き方を推定してきた。では、並べ替えのアルゴリズムがあらかじめ分かっているとき、外から観察できる並べ替え操作を測定・分析して、結果の違いをアルゴリズムの違いで説明できるだろうか？そこで、1から10の10個の数を、よく知られた整列アルゴリズムでコンピューターに整列させ、途中の数の交換・移動といったデータ操作を記録した。交換操作は、添字の大きな数を小さな方の前に移動し、互いに隣り合っていれば操作終了、隣り合っていなければ添字の小さな方を大きな方の元の位置に移動するという順序で記録した。操作対象に着目して時間的な共起を分析したところ、ヒープソートではその数より1つ大きい/小さい数と前後して動かされることが多く(**図126**)、バブル・ソートではその数自身と前後して動かされることが多い(**図124**)など、整列アルゴリズムの特徴が共起行列に表れた。コンピューターによる整列はどれも正解するが、内部的なアルゴリズムが異なると、外から観察できる並べ替え操作が異なることを確認できた[91]。

| n \ n+1 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 13,063,680 | 595,200 | 670,880 | 616,896 | 507,360 | 382,848 | 263,520 | 158,400 | 70,560 | 0 |
| 2 | 1,128,960 | 10,160,640 | 661,178 | 709,932 | 646,340 | 515,808 | 365,760 | 222,720 | 99,360 | 0 |
| 3 | 529,200 | 1,411,200 | 7,620,480 | 910,176 | 737,500 | 616,288 | 448,416 | 276,384 | 123,600 | 0 |
| 4 | 272,880 | 730,800 | 1,411,200 | 5,443,200 | 1,286,880 | 684,688 | 505,368 | 314,664 | 141,312 | 0 |
| 5 | 154,920 | 395,280 | 806,400 | 1,209,600 | 3,628,800 | 1,645,168 | 524,316 | 328,216 | 148,644 | 0 |
| 6 | 92,784 | 224,208 | 464,352 | 745,920 | 887,040 | 2,177,280 | 1,791,420 | 300,068 | 138,728 | 0 |
| 7 | 55,758 | 129,192 | 267,408 | 446,400 | 569,520 | 524,160 | 1,088,640 | 1,574,748 | 98,378 | 0 |
| 8 | 31,310 | 70,758 | 146,136 | 249,600 | 331,920 | 327,600 | 201,600 | 362,880 | 953,498 | 0 |
| 9 | 13,699 | 30,714 | 63,660 | 110,400 | 150,120 | 152,880 | 100,800 | 0 | 0 | 0 |
| 10 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**図124**　バブルソートの時間的な共起行列

| n \ n+1 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 533,400 | 404,520 | 664,220 | 709,792 | 627,770 | 569,382 | 521,569 | 430,344 | 377,547 | 334,896 |
| 2 | 55,512 | 330,792 | 393,940 | 591,884 | 643,412 | 612,126 | 572,400 | 465,422 | 401,321 | 347,040 |
| 3 | 191,408 | 154,180 | 323,848 | 428,540 | 628,932 | 711,788 | 687,245 | 544,890 | 455,513 | 368,040 |
| 4 | 581,762 | 343,798 | 204,750 | 253,388 | 402,562 | 605,533 | 713,512 | 586,496 | 481,105 | 358,200 |
| 5 | 420,714 | 410,502 | 341,188 | 212,100 | 209,086 | 391,977 | 626,804 | 651,199 | 531,405 | 359,352 |
| 6 | 470,138 | 433,990 | 415,608 | 336,266 | 134,819 | 192,840 | 498,395 | 674,131 | 647,392 | 414,012 |
| 7 | 483,562 | 426,628 | 462,432 | 439,514 | 273,830 | 157,665 | 234,206 | 573,350 | 683,807 | 466,740 |
| 8 | 419,436 | 394,330 | 426,916 | 452,942 | 391,489 | 288,112 | 128,844 | 199,219 | 482,283 | 478,560 |
| 9 | 410,208 | 370,108 | 391,506 | 425,446 | 401,518 | 407,129 | 257,670 | 87,482 | 105,982 | 217,680 |
| 10 | 429,644 | 381,696 | 384,936 | 402,712 | 354,246 | 445,546 | 396,119 | 226,613 | 73,238 | 0 |

**図125**　クイックソートの時間的な共起行列

| n \ n+1 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 3,938,756 | 2,908,320 | 3,908,456 | 3,505,804 | 2,414,256 | 2,213,716 | 1,630,812 | 2,159,600 | 1,103,232 | 1,233,976 |
| 2 | 6,360,968 | 1,728,344 | 2,661,264 | 2,016,332 | 2,219,548 | 1,787,904 | 1,656,784 | 1,537,396 | 1,412,516 | 1,386,012 |
| 3 | 6,411,964 | 6,324,196 | 695,640 | 622,256 | 1,083,552 | 1,165,272 | 1,295,364 | 1,059,988 | 1,421,492 | 1,406,268 |
| 4 | 3,982,384 | 3,593,440 | 5,625,328 | 390,768 | 409,000 | 645,736 | 848,116 | 725,644 | 1,267,180 | 1,320,916 |
| 5 | 2,505,680 | 2,451,816 | 2,708,272 | 5,911,952 | 185,088 | 283,208 | 522,440 | 507,368 | 1,007,816 | 1,176,104 |
| 6 | 1,983,464 | 1,853,640 | 1,791,968 | 2,055,544 | 6,042,056 | 67,200 | 250,292 | 363,092 | 704,852 | 1,023,092 |
| 7 | 1,582,248 | 1,495,800 | 1,409,064 | 1,449,136 | 1,738,564 | 6,004,612 | 17,280 | 267,552 | 456,192 | 915,552 |
| 8 | 954,864 | 1,212,352 | 1,248,080 | 1,206,900 | 1,263,408 | 1,609,152 | 5,972,352 | 0 | 302,286 | 907,086 |
| 9 | 532,392 | 640,392 | 709,352 | 749,112 | 825,678 | 1,085,046 | 1,646,046 | 6,273,006 | 0 | 401,056 |
| 10 | 392,112 | 492,912 | 589,632 | 681,352 | 772,648 | 876,256 | 1,005,856 | 1,199,056 | 4,513,696 | 0 |

**図126**　ヒープソートの時間的な共起行列

ただし、各アルゴリズムを分かったうで違いを説明でき、あるいは3択で当てることはできそうだが、この結果からアルゴリズムを推定するにはいたっていない。

### 11.5　タームの時間的な共起

「オレオレ詐欺」の分析では、ピースの時間的な共起行列を分析して、「2つある」、「ひとつは」といったタームに着目して解釈した(**図127**)。ここでピースの共起は機械的に導かれた客観的な事実だが、「2つある」や「ひとつは」に着目するのは人間の解釈である。では、「2つある」等への着目も機械的に導けないだろうか？

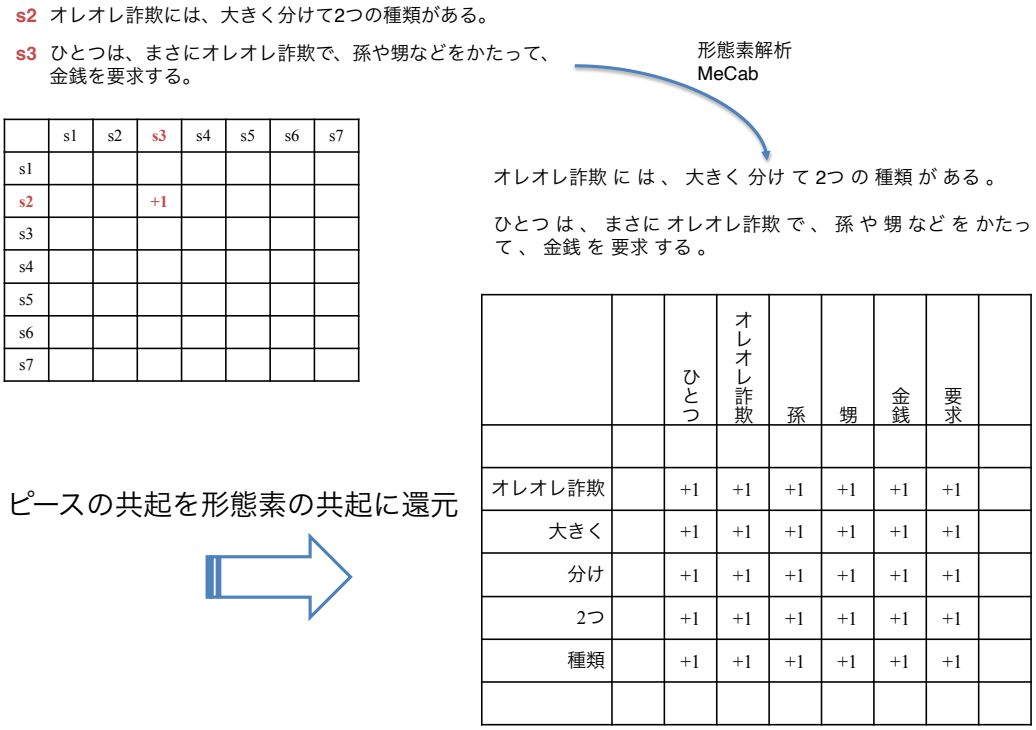
s1　相変わらず、オレオレ詐欺の被害が減らない。
s2　オレオレ詐欺には、大きく分けて2つの種類がある。
s3　ひとつは、まさにオレオレ詐欺で、孫や甥などをかたって、金銭を要求する。
s4　ひとつは、官公庁や銀行を騙って、還付金があるとだまして現金自動預払機を操作させ、金銭をだまし取る。
s5　この派生形として、口座が不正に操作されたとだまして、暗証番号を聞き出した上で、銀行カードもだまし取る。
s6　このような被害を防ぐ最良の方法は、電話での金銭の要求や、銀行口座やクレジットカードに係わる電話があった場合は、まず、詐欺を疑い、家族や親しい人に相談することだ。
s7　しかし、問題は、身近に相談できる人がいない高齢者が多くいることにあるのかもしれない。

| n \ n+1 | s1 | s2 | s3 | s4 | s5 | s6 | s7 |
|---|---|---|---|---|---|---|---|
| s1 | 18 | 17 | 7 | 6 | 3 | 9 | 7 |
| s2 | 7 | 9 | 30 | 16 | 1 | 1 | 1 |
| s3 | 6 | 4 | 16 | 25 | 7 | 14 | 4 |
| s4 | 2 | 2 | 11 | 15 | 41 | 7 | 6 |
| s5 | 15 | 0 | 8 | 3 | 9 | 19 | 8 |
| s6 | 3 | 3 | 4 | 7 | 4 | 11 | 38 |
| s7 | 14 | 2 | 2 | 11 | 7 | 8 | 5 |

**図127**　ピースの共起とタームに基づく解釈

ここでは結論を述べるに留めるが、うまくいかなかった。ピースの時間的な共起を、ピースを構成するタームや字句の時間的な共起に還元する方法は**図128**や**図129**に示すとおりである。ここで「オレオレ詐欺」を100万回解いたとしても、「オレオレ詐欺には大きく分けて2つの種類がある。」を構成する「オレオレ詐欺」や「大きく」といったターム間に、それぞれに等しく共起

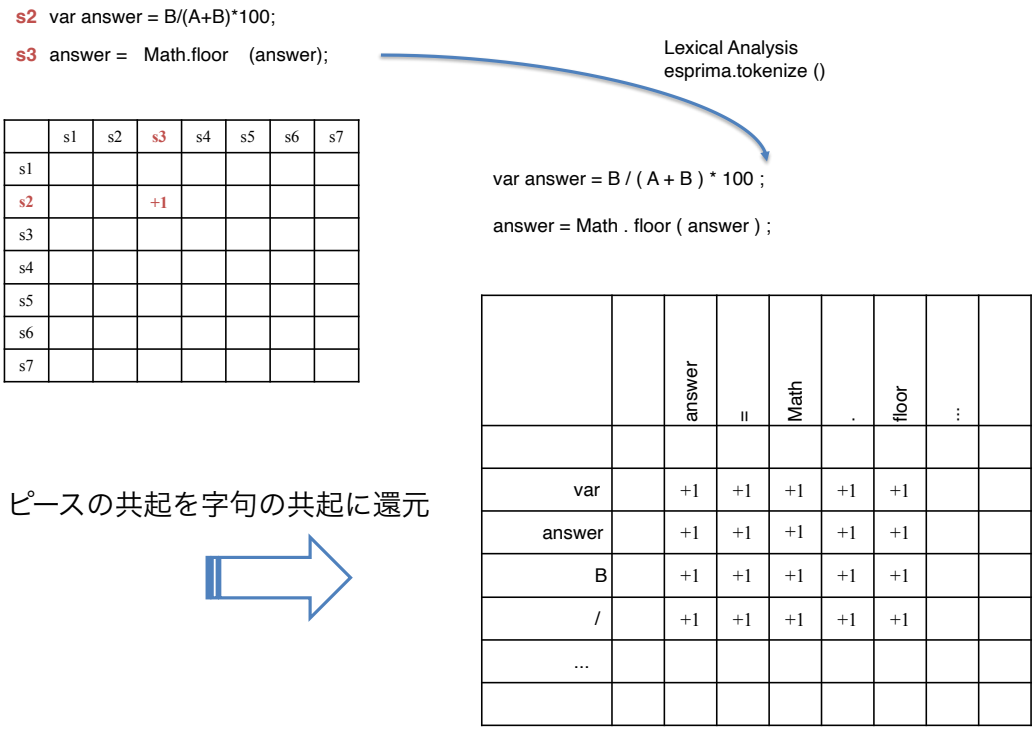
頻度が加算されることになり、「2つある」などを特別扱いしない。かといって、ターム単位でピースを構成すると、格段に難しいパズル問題となるだろう。これについては、さらなる検討が必要である。




| | s1 | s2 | s3 | s4 | s5 | s6 | s7 |
|---|---|---|---|---|---|---|---|
| s1 | | | | | | | |
| s2 | | | +1 | | | | |
| s3 | | | | | | | |
| s4 | | | | | | | |
| s5 | | | | | | | |
| s6 | | | | | | | |
| s7 | | | | | | | |

| | | ひとつ | オレオレ詐欺 | 孫 | 甥 | 金銭 | 要求 | |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | |
| オレオレ詐欺 | | +1 | +1 | +1 | +1 | +1 | +1 | |
| 大きく | | +1 | +1 | +1 | +1 | +1 | +1 | |
| 分け | | +1 | +1 | +1 | +1 | +1 | +1 | |
| 2つ | | +1 | +1 | +1 | +1 | +1 | +1 | |
| 種類 | | +1 | +1 | +1 | +1 | +1 | +1 | |
| | | | | | | | | |

**図128**　「オレオレ詐欺」のピースの時間的な共起をタームの時間的な共起に還元




| | s1 | s2 | s3 | s4 | s5 | s6 | s7 |
|---|---|---|---|---|---|---|---|
| s1 | | | | | | | |
| s2 | | | +1 | | | | |
| s3 | | | | | | | |
| s4 | | | | | | | |
| s5 | | | | | | | |
| s6 | | | | | | | |
| s7 | | | | | | | |

| | | answer | = | Math | . | floor | : | |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | |
| var | | +1 | +1 | +1 | +1 | +1 | | |
| answer | | +1 | +1 | +1 | +1 | +1 | | |
| B | | +1 | +1 | +1 | +1 | +1 | | |
| / | | +1 | +1 | +1 | +1 | +1 | | |
| ... | | | | | | | | |
| | | | | | | | | |

**図129**　プログム・コードのピースの時間的な共起を字句の時間的な共起に還元

# 12 結論とまとめ

思考プロセスを検出できる対話型アプリケーションの持つべき性質として、そのUIを構成するオブジェクトが、試行錯誤など、分析によって検出したいユーザーの思考を記述することばと対応すべき、などを提示した(3章)。そのような性質を持つアプリケーションが測定した時系列データからは、時間的な共起分析によって有益な情報を抽出できることを示した(4章)。時間的な共起分析の対象を取捨選択操作に絞ると、考えさせるために挿入した偽ピースがプレヤーに考えさせたかどうかを定量的に検出でき、作問を評価できる(5章)。解の状態遷移における閉路を分析することで、アンドゥによるバックトラックよりも多くの試行錯誤を機械的に指摘できた(6章)。提案アプリケーションの測定データからプレイヤーの操作間隔を分析すると、PO経験者と未経験者とでは、PO経験者の方が操作間隔の平均値が小さくバラツキが小さかった。提案アプリケーションの測定データが思考プロセスの特徴を捉えるものであることを、別の側面から確かめられた(7章)。「正解との距離の変化」によって思考プロセスを評価する研究があるが、コンピューターの整列を分析して、「正解との距離が単調に減少しなかったからといって、試行錯誤を捉えているとは限らない」ことを示した。(8章)。提案アプリケーションは測定データの欠損を検出できた。これによって、実際に起きている欠損が提案する分析手法に与える影響が十分に小さいことを示せた。これは提案アプリケーションが持つべきとした性質の効果である(9章)。提案に基づいてVUIで操作するアプリケーションを開発しGUIによる操作との違いを分析できた。これは、測定・分析の対象がピースだから実現できたと言え、提案アプリケーションが持つべきとした性質の効果である(10章)。提案アプリケーションは幅広い内容の実験や、実際の授業で使うことができた(11章)。

以上より、人が情報に一貫性をもたらすための思考プロセスを、試行錯誤を含めて検出できるアプリケーションの性質と手法を明らかにできた。

本研究の現状は、まだ扉を開いたばかりといったところである。将来的には、このようなアプリケーションや分析手法によって、あるクラス全体や学年全体や国全体といった集団について、試行錯誤の実態を明らかにできるだろう。その結果、試行錯誤が本当に良いことなのか、やはり役に立たないのかを、データに基づいて検証できるだろう。

# 謝辞

本稿の主要な部分はJSPS科研費20H01728(基盤研究(B))、17K01085(基盤研究(C))、26560124(挑戦的萌芽研究)の助成による成果である。

本研究の全過程を通じ、懇切なご指導ご鞭撻を賜った公立はこだて未来大学 システム情報科学部 新美礼彦 教授に感謝する。大学院博士後期課程において親切な指導と助言をいただくとともに、本研究をまとめるにあたり、時間を割いて丁寧な教示をいただいた公立はこだて未来大学 奥野拓 副学長、公立はこだて未来大学 システム情報科学部 角康之 教授、公立はこだて未来大学 システム情報科学部 伊藤恵 教授、広島工業大学 情報学部 松本慎平 教授に感謝する。

本研究は多くの研究者が提案アプリケーションを使った成果の上に立っている。公立はこだて未来大学(当時)および京都橘大学の大場研究室ほかの学生のみなさん、函館工業高等専門学校の藤原亮准教授、九州大学の峯恒憲教授、青山学院大学の松澤芳昭准教授や学生のみなさん、武蔵野大学の林浩一教授、大日本印刷株式会社の和泉直樹氏、鳥居昭彦氏、加藤輝実氏、歌田夢香氏、ほか大日本印刷のみなさん、各氏に感謝する。また、本研究は情報処理学会のデジタルドキュメント研究会(ドキュメントコミュニケーション研究会)およびコンピュータと教育研究会での活発な議論で磨かれてきた。

本研究のルーツをたどると、「山口さん、コンコーダンスって知ってますか？」と問われたことにたどり着く。このきっかけをくれたのは医薬品製造業における文書管理やメディカル・ライティングのコミュニティであった。大橋靖雄氏は、日本メディカルライター協会[*12]に門外漢の筆者を温かく迎え入れてくれた。

さらに遡ると、ジグソー・テキストのように文章を要素の集まりと考える発想は、大谷真氏の指導の元で参加したODA (Open Documnt Architecture)の経験に由来する。

文章をマークアップ(XML、HTML)によって多彩なビューで編集する、例えば本稿で取り上げたV字エディタのような発想は、浮川和宣社長と浮川初子専務のxfyに参加したことで勇気づけられている。

今回開発したアプリケーションはMacで開発したが、Windows、iPhone/iPad、AndroidやChromebookでも動く。Windows + EdgeやAndroidデバイスで使ってもらうために、何も追加開発していない。縦書きのジグソー・テキストもある。読み上げで操作するアクセシブルなUIもできた。日本語で開発したが、恐らく、中国語や右横書き言語のパズルも使えるだろう。なぜならUnicodeとWebの標準技術を使って開発したからである。それら標準技術を日本語でも使えるのは、樋浦秀樹氏、"fantasai" Elika J. Etemad氏、村田真氏、木田泰夫氏、小林龍生氏をはじめとする標準技術コミュニティの功績である。

*12 特定非営利活動法人 日本メディカルライター協会 (Japan Medical and Scientific Communicators Association, JMCA): https://www.jmca-npo.org/ ↗

従って、本稿もWeb技術で実現すべきである。本稿はHTML (HTML Living Standard)で記述しCSS (Cascading Style Sheets)を適用してVivliostyle[*13]で組まれている。研究を始めた頃は、ブラウザーで章節や図表の番号を本文から参照することすらできなかった。ブラウザーでは今でもできないが、これらを実現するCSSの仕様を実装したVivliostyleは、村上真雄氏、小形克宏氏をはじめとするVivliostyleオープンソース・コミュニティの功績である。

本研究で多くの人に使われ、本稿でも何度も登場するジグソー・テキストのパズル「オレオレ詐欺」は小林龍生氏が作成した。Topic Writerの、特に初期のワークシートは高橋慈子氏が作成した。

当初は誰にも理解されなかった本研究をここまで続けてこられたのは、共同研究者であり妻の大場みち子の功績である。

2024-08-05

*13 Vivliostyle: `https://vivliostyle.org/` ↗

# 業績一覧

2021年9月から2024年9月までの業績。

**学術雑誌(査読有り)**

- Yamaguchi, T., Matsuzawa, Y., Niimi, A., Oba, M. (2023). Cycles in State Transition as Trial-and-Errors in Solving Programming Exercises. In: Keane, T., Lewin, C., Brinda, T., Bottino, R. (eds) Towards a Collaborative Society Through Creative Learning. WCCE 2022. IFIP Advances in Information and Communication Technology, vol 685. Springer, Cham.
  https://doi.org/10.1007/978-3-031-43393-1_49 ↗
- 山口 琢, 松澤 芳昭, 新美 礼彦, 大場 みち子, 並べ替えプログラミング・パズルにおけるユーザアクセス，測定データのリトライと欠損, 電気学会論文誌C（電子・情報・システム部門誌）, 2024, 144 巻, 8 号, p. 764-770, 公開日 2024/08/01, Online ISSN 1348-8155, Print ISSN 0385-4221
  https://doi.org/10.1541/ieejeiss.144.764 ↗

**国際会議(査読有り)**

- Yamaguchi, T., Matsuzawa, Y., Niimi, A., Oba, M. (2022). Cycles in State Transition as Trial-and-Errors in Solving Programming Exercises. IFIP WCCE 2022: World Conference on Computers in Education
  https://wcce2022.org/ ↗

**国内学会・研究会**

- 山口 琢，歌田 夢香，加藤 輝実，新美 礼彦，大場 みち子, プロダクト・オーナーによるプロダクト・バックログの優先順位付け操作の分析, 日本ソフトウェア科学会, 第10回 実践的IT教育シンポジウム rePiT2024 in 京都, 2024【最優秀論文賞】
- 山口 琢, 松澤 芳昭, 新美 礼彦, 大場 みち子, 並べ替えプログラミング・パズルにおけるユーザ操作の傾向と性能設計, 電気学会 C部門 第93回情報システム研究会, (2023-10-26)
- 山口 琢, 歌田 夢香, 加藤 輝美, 新美 礼彦, 大場 みち子, スクラムにおけるプロダクト・バックログの優先順位付け操作の時間間隔の分析, 情報処理学会 研究報告コンピュータと教育(CE), 2023-CE-171(3), 1-11 (2023-10-14), 2188-8930

- 山口 琢, 新美 礼彦, 大場 みち子, ワードプロセッサーの機能の歴史, 情報処理学会 研究報告ドキュメントコミュニケーション(DC), 2023-DC-129(1), 1-2 (2023-07-28), 2188-8892
- 山口 琢, 新美 礼彦, 大場 みち子, スクリーン・リーダーによる複雑な情報処理における考え方の分析, 情報処理学会 研究報告ヒューマンコンピュータインタラクション(HCI), 2023-HCI-201(38), 1-6 (2023-01-09), 2188-8760
- 山口 琢, 新美 礼彦, 大場 みち子, 教育パズルを解くプロセスの分析手法の考察, 電気学会 C部門 第90回情報システム研究会, (2022-12-05)
- 山口 琢, 新美 礼彦, 大場 みち子, ジグソー・コードを使った考えるプロセスのトレーニングの仕組みの開発, 研究報告コンピュータと教育(CE), 2022-CE-164(13), 1-5 (2022-03-05), 2188-8930
- 山口 琢, 新美 礼彦, 大場 みち子, 文章のV字エディタの開発, 情報処理学会 研究報告コンピュータと教育(CE), 2022-CE-163(7), 1-6 (2022-01-29), 2188-8930
- 山口 琢, 松澤 芳昭, 新美 礼彦, 大場 みち子, 解答の状態遷移の閉路の分析によるジグソー・コードにおける試行錯誤の検出, 研究報告コンピュータと教育(CE), 2021-CE-162(28), 1-9 (2021-11-27), 2188-8930

**その他**

- 山口 琢, 新美 礼彦, 大場 みち子, 考えるプロセスの測定・分析のすすめ―学習プロセスの時間的な共起分析―, 知能と情報, 2021, 33 巻, 4 号, p. 117-125, 2021.
  https://doi.org/10.3156/jsoft.33.4_117 ↗

# 参考文献

各文献が見つかりやすくなるように、各文献の表記は、「Cite this paper」に示された記述など「そのページの表記」を尊重した。DOI (Digital Object Identifier, デジタルオブジェクト識別子)があればリンクを付している。文献本体PDFへのリンクだけでなく、文献のメタ情報へのリンクも添えた。例えばそれが国際学会の発表でプログラムのページが見つかれば、プログラムのページへのリンクも付している。これらの結果、文献の表記は様々となっている[*14]。

[1] 庵 愛, 竹川 佳成, 平田 圭二, 寺井 あすか, 推敲支援に向けた文章の階層構造を考慮した一貫性に関する評価指標の提案, 日本教育工学会論文誌, 2020, 44 巻, 4 号, p. 513-525, 公開日 2021/03/25
https://doi.org/10.15077/jjet.44048 ↗

[2] P. E. Ceruzzi and B. Grad, "Guest Editors' Introduction: PC Software--Word Processing for Everyone," in IEEE Annals of the History of Computing, vol. 28, no. 4, pp. 4-5, Oct.-Dec. 2006
doi: 10.1109/MAHC.2006.65 ↗

[3] P. Ceruzzi and B. Grad, "Guest Editors' Introduction: PC Software--Spreadsheets for Everyone," in IEEE Annals of the History of Computing, vol. 29, no. 3, pp. 4-5, July-Sept. 2007,
doi: 10.1109/MAHC.2007.4338437 ↗

[4] 情報処理学会歴史特別委員会編, 日本のコンピュータ史, オーム社, 2010.10
https://cir.nii.ac.jp/crid/1130282269357237888 ↗

[5] 相田洋, NHK取材班, 大墻敦, 新・電子立国 第3巻 世界を変えた実用ソフト, NHK出版, 1996
https://www.nhk-book.co.jp/detail/000000802731996.html ↗
https://cir.nii.ac.jp/crid/1573387449624759040 ↗

*14 著者にとって、自分の文献を見つけてもらえるファインダビリティは重要であろう。各国の研究力を評価するときには被引用数を取り上げることがあるし、国際学会や雑誌への投稿勧誘は登録されているコーパス(検索エンジン)をアピールしている(これらそのものの良し悪しは別である)。検索エンジンの処理方法・アルゴリズムは知る由もないが、「Yamaguchi, T.」と「Taku Yamaguchi」で検索結果が異なる現実がある。他方では、著者が参考文献にDOIを付記しても、それを削除して掲載する雑誌が存在する。冒頭の「コンピューター」表記の注釈でも述べたが、現在のICTインフラを踏まえて、このような慣行を見直すべきである。

[6] 中央教育審議会：幼稚園、小学校、中学校、高等学校及び特別支援学校の学習指導要領等の改善及び必要な方策等について(答申)(中教審第197号)、文部科学省（オンライン）
https://www.mext.go.jp/b_menu/shingi/chukyo/chukyo0/toushin/1380731.htm ↗
(参照2021-09-13)

[7] 国立教育政策研究所 教育課程研究センター：「指導と評価の一体化」のための学習評価に関する参考資料(高等学校編) 情報、(オンライン)、入手先
https://www.nier.go.jp/kaihatsu/shidousiryou.html ↗
(参照 2021-09-13).

[8] Perkins, D. N., Hancock, C., Hobbs, R., Martin, F., & Simmons, R. (1986). Conditions of Learning in Novice Programmers. Journal of Educational Computing Research, 2(1), 37-55.
https://doi.org/10.2190/GUJT-JCBJ-Q6QU-Q9PL ↗

[9] Yeyu Wang, Shimin Kai, and Ryan Shaun Baker. 2020. Early Detection of Wheel-Spinning in ASSISTments. In Artificial Intelligence in Education: 21st International Conference, AIED 2020, Ifrane, Morocco, July 6–10, 2020, Proceedings, Part I. Springer-Verlag, Berlin, Heidelberg, 574–585.
https://doi.org/10.1007/978-3-030-52237-7_46 ↗

[10] Beck, J.E., Gong, Y. (2013). Wheel-Spinning: Students Who Fail to Master a Skill. In: Lane, H.C., Yacef, K., Mostow, J., Pavlik, P. (eds) Artificial Intelligence in Education. AIED 2013. Lecture Notes in Computer Science(), vol 7926. Springer, Berlin, Heidelberg.
https://doi.org/10.1007/978-3-642-39112-5_44 ↗

[11] Yuko Suzuki, Review Best Practice: Improving the Review of Clinical-Regulatory Documents by Project Teams, Drug Information Association (DIA), 2005-09-30

[12] 笹田昌良, 他, GCP文書作成資料集, 情報機構, 2007
https://cir.nii.ac.jp/crid/1130282271361809536 ↗

[13] 山口 琢, 新美 礼彦, 大場 みち子, 考えるプロセスの測定・分析のすすめ―学習プロセスの時間的な共起分析―, 知能と情報, 2021, 33 巻, 4 号, p. 117-125, 公開日 2022/05/15,
https://doi.org/10.3156/jsoft.33.4_117 ↗

[14] 吉田賢志朗, 松澤芳昭: “初学者のプロセス改善を目指した漸進的プログラミング支援システムの開発と評価,” 情報処理学会 研究報告コンピュータと教育(CE), Vol.2021-CE-159, No.12, pp.1-9, 2021.
http://id.nii.ac.jp/1001/00209877/ ↗

[15] Shimada, A., Konomi, S. and Ogata, H. (2018), "Real-time learning analytics system for improvement of on-site lectures", Interactive Technology and Smart Education, Vol. 15 No. 4, pp. 314-331.
https://doi.org/10.1108/ITSE-05-2018-0026 ↗

[16] Taku Yamaguchi, Michiko Oba, "Measurable Interactive Application to Find Out User Recognition and Strategy when Problem Solving," Journal of Software vol. 15, no. 1, pp. 12-22, 2020.,
DOI: 10.17706/jsw.15.1.12-22 ↗

[17] 高橋 慈子, 山口 琢, 大場 みち子, 小林 龍生, 文章力向上教育におけるトピックライティングツールの活用, 情報処理学会 研究報告ドキュメントコミュニケーション(DC), 2015-DC-98(7),1-6 (2015-07-06) , 2188-8892
http://id.nii.ac.jp/1001/00142607/ ↗

[18] 高橋 慈子 , 山口 琢 , 大場 みち子 , 小林 龍生, 文書作成教育におけるトピックライティングツール活用と効果, 情報処理学会 研究報告ドキュメントコミュニケーション(DC), 2016-DC-101(9),1-6 (2016-03-17) , 2188-8892
http://id.nii.ac.jp/1001/00158315/ ↗

[19] 高橋 慈子, 大場 みち子, 山口 琢, 小林 龍生, テクニカルライティング教育におけるクラウドツール活用と相互レビューの効果分析, 情報処理学会 研究報告ドキュメントコミュニケーション(DC), 2017-DC-104(3),1-7 (2017-03-03) , 2188-8892
http://id.nii.ac.jp/1001/00178181/ ↗

[20] 高橋 慈子, 大場 みち子, 山口 琢, 藤原 亮, 小林 龍生, 作文教育におけるツール活用とアクティブラーニングの考察, 情報処理学会 研究報告コンピュータと教育(CE), 2018-CE-144(29),1-7 (2018-03-10) , 2188-8930
http://id.nii.ac.jp/1001/00186666/ ↗

[21] 高橋 慈子, 大場 みち子, 山口 琢, 文章の型の提供と相互レビューツールによる作文指導の授業実践, 情報処理学会 研究報告コンピュータと教育(CE),2019-CE-149(16),1-7, 2019
http://id.nii.ac.jp/1001/00194645/ ↗

[22] 山口 琢, 大場 みち子, 高橋 慈子, 小林 龍生, ジグソー・テキストによる文並べ替え操作の測定, 情報処理学会 研究報告コンピュータと教育(CE), 2017-CE-142(27),1-6, 2017
http://id.nii.ac.jp/1001/00184931/ ↗

[23] 嶋津 恵子：INCOSE標準基盤型情報システム設計提案、情報処理学会 研究報告グループウェアとネットワークサービス(GN)、Vol.2011-GN-81、No.4、pp.1-7（オンライン）、入手先（2011）。
http://id.nii.ac.jp/1001/00077394/ ↗

[24] 中央教育審議会, 2040年に向けた高等教育のグランドデザイン(答申)（中教審第211号）, 2018
https://www.mext.go.jp/b_menu/shingi/chukyo/chukyo0/toushin/1411360.htm ↗

[25] 久野靖, 思考力・判断力・表現力を評価する試験問題の作成手順, 情報処理学会 情報教育シンポジウム論文集,2018(1),1-8, 2018
http://id.nii.ac.jp/1001/00190680/ ↗

[26] 中山 陽平, 掛下 哲郎, プログラミング穴埋め問題における穴抜きの難易度と学生の解答過程のクラスタ分析, 情報教育シンポジウム論文集, 2018(23),166-173,2018
http://id.nii.ac.jp/1001/00190702/ ↗

[27] 石田 基広、Rによるテキストマイニング入門、森北出版、2008
https://cir.nii.ac.jp/crid/1130000796093686784 ↗

[28] 稲場 大樹, 福井 健一, 佐藤 一永, 水崎 純一郎, 沼尾 正行, 燃料電池における損傷パターン抽出のための共起クラスタマイニング, 人工知能学会論文誌, 2012, 27 巻, 3 号, p. 121-132, 公開日 2012/04/06
https://doi.org/10.1527/tjsai.27.121 ↗

[29] 福井 健一, 沼尾 正行, 事象系列データからの共起性マイニング : 燃料電池の損傷間および地震間の相互作用抽出(<特集>データ中心科学), 人工知能, 2015, 30 巻, 2 号, p. 238-246, 公開日 2020/09/29
https://doi.org/10.11517/jjsai.30.2_238 ↗

[30] Amruth N. Kumar. 2019. Representing and Evaluating Strategies for Solving Parsons Puzzles. In Intelligent Tutoring Systems: 15th International Conference, ITS 2019, Kingston, Jamaica, June 3–7, 2019, Proceedings. Springer-Verlag, Berlin, Heidelberg, 193–203.
https://doi.org/10.1007/978-3-030-22244-4_24 ↗

[31] 大場 みち子, 山口 琢, 川北 紘正, パズルを利用したプログラミング思考過程の分析, 情報処理学会 情報教育シンポジウム論文集,2019,152-159 (2019-08-10)
http://id.nii.ac.jp/1001/00198554/ ↗

[32] 大場みち子、山口琢、作文行動の記録・分析ツールを用いた就活自己紹介書の作成と分析, 情報処理学会 研究報告コンピュータと教育（CE）,2018-CE-147(12),1-7 (2018-11-24) , 2188-8930
http://id.nii.ac.jp/1001/00192427/ ↗

[33] 宮澤賢治, 〔雨ニモマケズ〕, 青空文庫,
https://www.aozora.gr.jp/cards/000081/card45630.html ↗

[34] 山口 琢 , 松澤 芳昭 , 新美 礼彦 , 大場 みち子, 取捨選択操作の時間的な共起分析によるプログラミング・プロセスでの迷いの検出, 情報処理学会 研究報告コンピュータと教育（CE）,2021-CE-160(8),1-6 (2021-05-29)
http://id.nii.ac.jp/1001/00211364/ ↗

[35] 森永 笑子, 松本 慎平, 村上 瑠香, 林 雄介, 平嶋 宗, カード操作方式によるプログラミング学習支援システムでの学習過程の可視化方法の提案, 人工知能学会研究会資料 先進的学習科

学と工学研究会, 2019, 85 巻, 85回 (2019/3), p. 18-, 公開日 2021/06/28
https://doi.org/10.11517/jsaialst.85.0_18 ↗

[36] 森永 笑子, 松本 慎平, 林 雄介, 平嶋 宗, カード操作方式によるプログラミング学習支援システムにおける学習者の活動に基づく学習課題の特徴分析, 電気学会【C】電子・情報・システム部門 情報システム研究会, pp. 41—44 (2019).
https://www.bookpark.ne.jp/cm/ieej/detail/IEEJ-IS19068-PDF/ ↗

[37] Maharjan, S., Kumar, A.: Using Edit Distance Trails to Analyze Path Solutions of Parsons Puzzles. In: The 13th International Conference on Educational Data Mining (EDM 2020) , Anna N. Rafferty, Jacob Whitehill, Violetta Cavalli-Sforza, and Cristobal Romero (eds.), pp. 638—642 (2020).
https://educationaldatamining.org/files/conferences/EDM2020/papers/paper_163.pdf ↗
Proceedings - Educational Data Mining 2020 (EDM2020) ↗

[38] 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, 情報処理学会 研究報告コンピュータと教育(CE),2020-CE-154(4),1-8 (2020-03-07) , 2188-8930
http://id.nii.ac.jp/1001/00204096/ ↗
(2020)

[39] 山口 琢, 中村 陽太, 大場 みち子, プログラム・コードの並べ替えパズルにおける正解との距離の変化, 情報処理学会 情報教育シンポジウム論文集,2020,47-53 (2020-12-12)
http://id.nii.ac.jp/1001/00208673/ ↗

[40] 山口 琢, 大場 みち子, コンピュータの整列処理で正解との距離は単調に減少するか?, 情報処理学会 研究報告コンピュータと教育(CE),2020-CE-157(6),1-8 (2020-10-31) , 2188-8930
http://id.nii.ac.jp/1001/00207490/ ↗

[41] 大場 みち子, 山口 琢, プログラミング行動の測定と分析に関する一考察, 情報処理学会 研究報告コンピュータと教育 (CE) ,2017-CE-140(6),1-6 (2017-07-01) , 2188-8930
http://id.nii.ac.jp/1001/00182392/ ↗

[42] S. Kawaguchi, Y. Sato, H. Nakayama, R. Onuma, S. Nakamura and Y. Miyadera, "Machine Learning Model for Analyzing Learning Situations in Programming Learning," 2018 IEEE Conference on Big Data and Analytics (ICBDA), Langkawi, Malaysia, 2018, pp. 74-79,
10.1109/ICBDAA.2018.8629776 ↗
(2018).

[43] Kawaguchi, S., Nishikawa, T., Sato, Y., Onuma, R., Nakayama, H., Nakamura, S. and Miyadera, Y.: Development of a Training Data Creation Support Environment for Estimating Programming Learning Situations, Proc.*2019 18th International Conference on Information Technology Based Higher Education and Training* (*ITHET*)(online), DOI:

10.1109/ITHET46829.2019.8937339 ↗
(2019).

[44] 川口 翔大, 佐藤 克己, 大沼 亮, 中山 祐貴, 中村 勝一, 宮寺 庸造, プログラミング演習授業におけるAI手法を用いた学習状況自動推定システム, 電子情報通信学会 技術研究報告 教育工学研究会(ET), Vol. 119, No. 468, ET2019-101, pp.141—146 (ET) (2020-03-07).
https://www.ieice.org/publications/search/summary.php?id=107794&tbl=ken ↗

[45] Du, Y., Luxton-Reilly, A. and Denny, P.: A Review of Research on Parsons Problems, Proc. ACE'20: Twenty-Second Australasian Computing Education Conference, pp.195–202 (online), DOI: 10.1145/3373165.3373187 ↗
(2020).

[46] Ueno, S., Terui, Y., Imamura, R., Kuno, Y. and Egi, H.: Analysis of the Answering Processes in Split-Paper Testing to Promote Instruction. In: Rodrigo, M. M. T. et al. (eds.) Proceedings of the 29th International Conference on Computers in Education. Asia-Pacific Society for Computers in Education
http://id.nii.ac.jp/1001/00212996/ ↗
(2021).

[47] 山口 琢, 大場 みち子, できごと、手順、プログラムや地理の並べ替え操作の測定と分析, 情報処理学会 情報教育シンポジウム論文集,2018(24),174-178 (2018-08-12),
http://id.nii.ac.jp/1001/00190704/ ↗
(2018).

[48] 中村 陽太, 大場 みち子, 山口 琢, 伊藤 恵, 学習進度に対応するパズルを利用したプログラミング思考過程の分析, 情報処理学会研究報告コンピュータと教育(CE), 2019-CE-151(1), 2019-09-28
http://id.nii.ac.jp/1001/00199559/ ↗

[49] 中村 陽太, 大場 みち子, 山口 琢, 伊藤 恵, 授業進度に対応するパズルを利用したプログラミング思考過程の分析と教育支援システムの開発, 情報処理学会 第82回全国大会講演論文集, 2020-02-20
http://id.nii.ac.jp/1001/00205909/ ↗

[50] 山口 琢, 大場 みち子, 編集操作の時間的共起分析の提案, 情報処理学会 研究報告コンピュータと教育（CE）,2019-CE-151(9),1-7 (2019-09-28), 2188-8930
http://id.nii.ac.jp/1001/00199567/ ↗

[51] Yamaguchi, T., Matsuzawa, Y., Niimi, A., Oba, M. (2023). Cycles in State Transition as Trial-and-Errors in Solving Programming Exercises. In: Keane, T., Lewin, C., Brinda, T., Bottino, R. (eds) Towards a Collaborative Society Through Creative Learning. WCCE 2022.

IFIP Advances in Information and Communication Technology, vol 685. Springer, Cham. https://doi.org/10.1007/978-3-031-43393-1_49 ↗

[52] Helminen, J., Ihantola, P., Karavirta, V., Malmi, L.: How do students solve parsons programming problems?: an analysis of interaction traces. In: Proceedings of the ninth annual international conference on International computing education research (ICER '12). Association for Computing Machinery, New York, NY, USA, pp. 119—126. (2012). https://doi.org/10.1145/2361276.2361300 ↗

[53] Nakayama, Y., Kuno, Y., Kakuda, H.: Split-Paper Testing: A Novel Approach to Evaluate Programming Performance. Journal of Information Processing 28, 733–743 (2020). http://id.nii.ac.jp/1001/00207329/ ↗

[54] Aronson, E.: Jigsaw Basics. (n. d.). https://www.jigsaw.org/pdf/JigsawBasics.pdf ↗

[55] Jeannette, W.: Computational Thinking, Communications of the ACM, vol. 49, No.3, pp.33–35. Association for Computing Machinery, New York, NY, USA (2006). DOI: 10.1145/1118178.1118215 ↗

[56] 山口 琢, 歌田 夢香, 加藤 輝実, 新美 礼彦, 大場 みち子, プロダクト・オーナーによるプロダクト・バックログの優先順位付け操作の分析, 実践的IT教育シンポジウム rePiT 論文集, 2024, 2024 巻, 第10回 実践的IT教育シンポジウム rePiT2024 in 京都, p. 43-52, 公開日 2024/03/07 https://doi.org/10.11309/repit.2024.0_43 ↗

[57] 教育システム情報学会：教育システム情報学会(JSiSE) 2022年度 特集論文研究会 講演・参加募集、(オンライン)、 https://www.jsise.org/research_society/2022_cfp_special ↗ (参照2023-09-23)。

[58] 歌田 夢香, 加藤 輝実, 山口 琢, 大場 みち子, 新美 礼彦, アジャイル開発におけるプロダクトオーナー育成支援 ワークショップの実践, 実践的IT教育シンポジウム rePiT 論文集, 2024, 2024 巻, 第10回 実践的IT教育シンポジウム rePiT2024 in 京都, p. 35-42, 公開日 2024/03/07 https://doi.org/10.11309/repit.2024.0_35 ↗

[59] Ken Schwaber, Jeff Sutherland, スクラムガイド 2020年11月, (オンライン), https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-Japanese.pdf ↗ (参照 2024-01-16)

[60] 西村 直人, 永瀬 美穂, 吉羽 龍太郎, SCRUM BOOT CAMP THE BOOK【増補改訂版】スクラムチームではじめるアジャイル開発, 翔泳社, 2020 https://cir.nii.ac.jp/crid/1130285378194668672 ↗

[61] 山口 琢, 歌田 夢香, 加藤 輝美, 新美 礼彦, 大場 みち子, スクラムにおけるプロダクト・バックログの優先順位付け操作の時間間隔の分析, 情報処理学会 研究報告コンピュータと教育(CE),2023-CE-171(3),1-11 (2023-10-14), 2188-8930
http://id.nii.ac.jp/1001/00228336/ ↗

[62] 山崎 治, 瀬田 和久, 「答えのない課題の解決に挑む学び」を支える教育システム・デザイン, 教育システム情報学会誌, 2023, 40 巻, 2 号, p. 103-104, 公開日 2023/04/01
https://doi.org/10.14926/jsise.40.103 ↗

[63] 文部科学省：小学校学習指導要領（平成29年告示)、(オンライン)、
https://www.mext.go.jp/content/20230120-mxt_kyoiku02-100002604_01.pdf ↗
（参照2023-09-26)。

[64] 野崎 有以, 家庭科教科書における「生活時間」, 家政学原論研究, 2015, 49 巻, p. 20-29, 公開日 2017/04/07
https://doi.org/10.20596/jphe.49.0_20 ↗

[65] 岡崎 善弘, 井邑 智哉, 高村 真広, 徳永 智子, 夏休みの宿題に取り組む計画・実際の一致と取り組み方がストレスに与える影響, 時間学研究, 2018, 9 巻, p. 1-7, 公開日 2019/05/01
https://doi.org/10.20740/timestudies.9.0_1 ↗

[66] 井邑 智哉, 岡崎 善弘, 高村 真広, 徳永 智子, 児童の時間管理が長期休暇中の学習に及ぼす影響, 時間学研究, 2021, 12 巻, p. 53-60, 公開日 2022/07/01
https://doi.org/10.20740/timestudies.12.0_53 ↗

[67] 三宅 幹子 , 松田 文子, 大学生における時間管理能力--レポート課題への取り組みを通して--, 福山大学人間文化学部紀要, 2009-03
https://fukuyama-u.repo.nii.ac.jp/records/5666 ↗

[68] 岡崎 善弘, 時間管理研究の現在, 時間学研究, 2012, 2 巻, p. 45-53, 公開日 2017/02/28
https://doi.org/10.20740/timestudies.2.0_45 ↗

[69] 山口 琢, 松澤 芳昭, 新美 礼彦, 大場 みち子: 解答の状態遷移の閉路の分析によるジグソー・コードにおける試行錯誤の検出, 情報処理学会 研究報告コンピュータと教育(CE), Vol. 2021-CE-162, No. 28, pp. 1-9 (2021)（2021)。
http://id.nii.ac.jp/1001/00214065/ ↗

[70] 田柳 恵美子, 社会研究における「認知的アプローチ」の潮流, 認知科学, 2007, 14 巻, 1 号, p. 60-73, 公開日 2008/12/15
https://doi.org/10.11225/jcss.14.60 ↗

[71] 森永 笑子, 松本 慎平, 林 雄介, 平嶋 宗, カード操作方式によるプログラミング学習支援システムにおける学習者の活動に基づく学習課題の特徴分析, 電気学会C部門 情報システム研究会(CIS), 2019-11-10
https://www.bookpark.ne.jp/cm/ieej/detail/IEEJ-IS19068-PDF/ ↗

[72] 伊東大輝, 島川博光, コードパズルによるプログラミング的思考力を考慮した理解度の推定, 情報科学技術フォーラム(FIT2020)講演論文集, (2020-08-18)
K-021 ↗

[73] 奥村晴彦, ［改訂新版］C言語による標準アルゴリズム事典, 技術評論社, 2018
https://cir.nii.ac.jp/crid/1130282271245363328 ↗,
https://gihyo.jp/dp/ebook/2018/978-4-7741-9787-6 ↗

[74] 神嶌敏弘, 順序の距離と確率モデル, 人工知能学会第二種研究会資料, 2009, 2009 巻, DMSM-A902 号, p. 07-, 公開日 2021/08/28
https://doi.org/10.11517/jsaisigtwo.2009.DMSM-A902_07 ↗

[75] SymPy Development Team, SymPy 1.6.2 documentation, 2020-08-09.
https://docs.sympy.org/latest/index.html ↗

[76] CRAN, PerMallows: Permutations and Mallows Distributions, 2017-04-28
https://cran.r-project.org/package=PerMallows ↗

[77] 松本 慎平, 岩井 健吾, 前田 一誠, 山元 翔, 林 雄介, 平嶋 宗, 学習ログデータからの特異性の検出と情報構造に基づく意味的分析による学習課題の再検討 ——単文統合型作問学習支援システムモンサクンの実践データを事例として——, 教育システム情報学会誌, 2023, 40 巻, 3 号, p. 203-218 (2023)
https://doi.org/10.14926/jsise.40.203 ↗

[78] 長島 和平, 久保 龍斗, 並木 美太郎, 長 慎也, 兼宗 進, 作業時間を用いたプログラミング学習者のつまずき検出手法の検討, 情報処理学会 研究報告コンピュータと教育（CE）,2023-CE-168(7),1-8 (2023-02-04)
http://id.nii.ac.jp/1001/00224022/ ↗

[79] 長 慎也, 岸本 有生, 長島 和平, 兼宗 進, 並木 美太郎, Bit Arrowにおける組み込み機器実行機能と、収集データの共有・分析機能との連携, 情報処理学会 研究報告コンピュータと教育(CE), Vol. 2021-CE-161, NO. 7, pp.1-7 (2021-10-16)
http://id.nii.ac.jp/1001/00213309/ ↗

[80] 緒方 広明, 殷 成久, 毛利 考佑, 大井 京, 島田 敬士, 大久保 文哉, 山田 政寛, 小島 健太郎, 教育ビッグデータの利活用に向けた学習ログの蓄積と分析, 教育システム情報学会誌, 2016, 33 巻, 2 号, p. 58-66 (2016)
https://doi.org/10.14926/jsise.33.58 ↗

[81] 山口 琢, 松澤 芳昭, 大場 みち子: リアルタイムでモニタする並べ替えプログラミング・パズルにおけるユーザ操作の傾向と性能設計、情報処理学会 研究報告コンピュータと教育(CE), Vol. 2021-CE-159, NO. 24, pp.1-10 (2021-03-06)
http://id.nii.ac.jp/1001/00209889/ ↗

[82] 山口琢, 松澤芳昭, 新美礼彦, 大場みち子, 並べ替えプログラミング・パズルにおけるユーザ操作の傾向と性能設計, 電気学会 第93回情報システム研究会 (2023-10-26)

https://www.bookpark.ne.jp/cm/ieej/detail/IEEJ-20231026C00801-010-PDF/ ↗

[83] 平成28〜30年度文部科学省 大学入学者選抜改革推進委託事業, 情報学的アプローチによる「情報科」大学入学者選抜における評価手法の研究開発
最終成果報告書 別添資料6 大規模CBTシステム構築への課題とその解決策
https://www.mext.go.jp/content/1412881_5_1.pdf ↗

[84] 清水 康敬, 中山 実, 向後 千春, 教育工学研究の方法, 教育工学選書, ミネルヴァ書房, 2012
https://cir.nii.ac.jp/crid/1130282272414480896 ↗
https://www.minervashobo.co.jp/book/b100591.html ↗

[85] 矢野 米雄, 平嶋 宗, 教育工学とシステム開発, 教育工学選書, ミネルヴァ書房, 2012
https://cir.nii.ac.jp/crid/1130000796795460992 ↗
https://www.minervashobo.co.jp/book/b102919.html ↗

[86] VoiceOver, iPhoneユーザガイド
https://support.apple.com/ja-jp/guide/iphone/iph3e2e415f/16.0/ios/16.0 ↗

[87] 喜多 敏博, 長岡 千香子, 平岡 斉士, スマートスピーカーを通じたLMS上での学習活動, 情報処理学会 研究報告教育学習支援情報システム(CLE), Vol. 2018-CLE-26, No. 16, pp.1—5, 2018-11-30
http://id.nii.ac.jp/1001/00192606/ ↗

[88] 藤本 光史, スライドパズルにおける回転型操作とアクセシビリティ, 情報処理学会 研究報告アクセシビリティ(AAC), Vol. 2017-AAC-5, No. 3, pp.1—6, 2017-12-01
http://id.nii.ac.jp/1001/00184617/ ↗

[89] 山口琢, 大場みち子, スクリーンリーダーで操作するジグソー・テキスト - アクセシブルな学習分析とComputer Based Testing, 情報処理学会 研究報告コンピュータと教育(CE), Vol. 2019-CE-149, No. 15, pp.1—8, 2019-02-23
http://id.nii.ac.jp/1001/00194644/ ↗

[90] 山口琢, 大場みち子, 音声UIでアクセスする学習分析システムの図表, 情報処理学会 情報教育シンポジウム論文集, Vol. 2019, pp.46—53, 2019-08-10
http://id.nii.ac.jp/1001/00198540/ ↗

[91] 山口琢, 大場みち子, コンピューターの整列処理におけるデータ操作の時間的共起分析, 情報処理学会 研究報告コンピュータと教育(CE), Vol. 2020-CE-156, No. 1 pp.1—7, 2020-08-22。
http://id.nii.ac.jp/1001/00206288/ ↗

[92] 山口琢, 新美礼彦, 大場みち子, 数の整列プロセスの人とコンピュータの比較, 認知科学会第38回大会, 2021-09-05
https://www.jcss.gr.jp/meetings/jcss2021/proceedings/pdf/JCSS2021_P2-41.pdf ↗

[93] 林 浩一, 山口 琢, 大場 みち子, ロジカルシンキングにおける基本的関係についてのジグソー・テキストを用いた理解度評価, 情報処理学会 研究報告コンピュータと教育(CE), 2019-CE-151(13),1-8 (2019-09-28) , 2188-8930
http://id.nii.ac.jp/1001/00199571/ ↗

[94] 林 浩一, 山口 琢, 大場 みち子, ロジカルシンキングにおける目的と手段が反転する誤答発生の過程分析, 情報処理学会 研究報告コンピュータと教育(CE), 2019-CE-152(21),1-8 (2019-11-08) , 2188-8930
http://id.nii.ac.jp/1001/00200286/ ↗

[95] 藤井 沙苗, 松澤 芳昭, パズル型問題を利用したプログラミング初学者の理解度と思考過程の分析, 情報処理学会 第83回全国大会講演論文集,2021(1),769-770 (2021-03-04)
http://id.nii.ac.jp/1001/00215600/ ↗

[96] 西村 萌, 米澤 彩乃, 上島 綺夏, 黒木 奏子, 松澤 芳昭, プログラミング初学者のためのパズル型問題開発と問題分析, 情報処理学会 研究報告コンピュータと教育(CE), 2022-CE-164(7),1-8 (2022-03-05) , 2188-8930
http://id.nii.ac.jp/1001/00217282/ ↗

[97] 米澤 彩乃 , 西村 萌 , 松澤 芳昭, アルゴリズム構築学習のための日本語訳プログラミングパズルの開発と評価, 情報処理学会 研究報告コンピュータと教育(CE), 2023-CE-169(9),1-8 (2023-03-04) , 2188-8930
http://id.nii.ac.jp/1001/00225008/ ↗

[98] 西村 萌, 米澤 彩乃, 松澤 芳昭, パズル型プログラミング問題を利用した初学者の迷いや思考プロセスの個別分析の試み, 情報処理学会 研究報告コンピュータと教育(CE), 2023-CE-169(10),1-8 (2023-03-04) , 2188-8930
http://id.nii.ac.jp/1001/00225009/ ↗

[99] 黒木 奏子, 松澤 芳昭, SQL学習者のためのパズル型問題の開発と評価, 研究報告コンピュータと教育（CE）,2023-CE-169(4),1-9 (2023-03-04)
http://id.nii.ac.jp/1001/00225003/ ↗

[100] 藤田 然, 金森大輝, 松澤芳昭, 鈴木彩未, アルゴリズム構築を指向したプログラミングパズル問題の開発と実践, 情報処理学会 第86回全国大会講演論文集, 2024

[101] 村田 菜月, 松澤 芳昭, 論理的思考力を養うことを狙ったパズル型証明問題の試作と高校での実践, 情報処理学会 研究報告コンピュータと教育(CE), 2024-CE-174(4),1-8 (2024-03-02) , 2188-8930
http://id.nii.ac.jp/1001/00232996/ ↗

[102] 高橋 沙織, 長谷川 要, 栗山 健太, 豊田 和人, 佐藤 義哉, 長内 将吾, 鉢呂 誠市, 神野 香菜子, 野辺 陽平, 岡澤 尚也, 渡邉 健, 数理科学を学ぶ環境のデザイン, 公立はこだて未来大学 2018 年度 システム情報科学実習 グループ報告書, 2019
https://www.fun.ac.jp/pbl-theme#1-6 ↗

https://www.fun.ac.jp/wp-content/uploads/2020/03/project04-1.pdf ↗
https://www.fun.ac.jp/wp-content/uploads/2020/03/document04_A-1.pdf ↗

[103] 大場 みち子, 山口 琢, 地理情報を題材とした並べ替えアプリを用いたデータサイエンス教育の実践, 情報処理学会 研究報告ドキュメントコミュニケーション（DC）,2024-DC-132(2),1-6 (2024-03-18) , 2188-8892
↗http://id.nii.ac.jp/1001/00233414/

[104] 大場 みち子, 山口 琢, 高橋 慈子, 小林 龍生, 文章作成における文章評価と編集操作との関係分析, 情報処理学会 情報教育シンポジウム 2016 論文集,2016,67-73 (2016-08-15)
http://id.nii.ac.jp/1001/00176079/ ↗

[105] 大場 みち子 , 山口 琢 , 高橋 慈子 , 小林 龍生 , 藤原 亮, 文章作成とレビュー効果の測定と分析, 情報処理学会 研究報告コンピュータと教育(CE), 2018-CE-144(28),1-7 (2018-03-10) , 2188-8930
http://id.nii.ac.jp/1001/00186665/ ↗