UNIVERSIDADE FEDERAL DO AMAZONAS

FACULDADE DE TECNOLOGIA

ENGENHARIA DA COMPUTAÇÃO

Uso de problemas de Parson no ensino de programação introdutória

Anilton Carlos de Lima Reis

Manaus - AM

Julho, 2023

Anilton Carlos de Lima Reis

Uso de problemas de Parson no ensino de programação introdutória

Monografia de Graduação apresentada ao Instituto de Computação da Universidade Federal do Amazonas como requisito parcial para a obtenção do grau de bacharel em Engenharia da Computação.

Orientadora

Profa. Dra. Elaine Harada Teixeira de Oliveira

Coorientador

Prof. Dr. Leandro Silva Galvão de Carvalho

Universidade Federal do Amazonas

Faculdade de Tecnologia

Manaus - AM

Julho, 2023

Ficha Catalográfica

Ficha catalográfica elaborada automaticamente de acordo com os dados fornecidos pelo(a) autor(a).

Lima Reis, Anilton Carlos de
L732u Uso de problemas de Parson no ensino de programação introdutória / Anilton Carlos de Lima Reis . 2023
45 f.: il. color; 31 cm.

Orientadora: Elaine Harada Teixeira de Oliveira
Coorientador: Leandro Silva Galvão de Carvalho
TCC de Graduação (Engenharia da Computação) - Universidade Federal do Amazonas.

1. Problemas de Parson. 2. Introdução à programação. 3. Carga cognitiva. 4. Ganho de aprendizagem. I. Oliveira, Elaine Harada Teixeira de. II. Universidade Federal do Amazonas III. Título

Monografia de Graduação sob o título Uso de problemas de Parson no ensino de programação introdutória apresentada por Anilton Carlos de Lima Reis e aceita pela Faculdade de Tecnologia da Universidade Federal do Amazonas, sendo aprovada por todos os membros da banca examinadora abaixo especificada:

Profa. Dra. Elaine Harada Teixeira de Oliveira
Orientadora
Instituto de Computação
Universidade Federal do Amazonas

Prof. Dr. David Braga Fernandes de Oliveira
Instituto de Computação
Universidade Federal do Amazonas

Prof. Dr. Eduardo James Pereira Souto
Instituto de Computação
Universidade Federal do Amazonas

Manaus - AM, 05 de julho de 2023.

Dedico este trabalho aos meus pais, que se esforçaram ao máximo para que eu pudesse estudar na UFAM e me apoiaram a todo momento para continuar firme nessa caminhada.

Agradecimentos

Primeiramente agradeço aos meus pais, Aucirete Medeiros de Lima e Manoel Anilton de Lima Reis, que sempre se esforçaram mais do que podiam para que eu pudesse estudar na UFAM e sempre estiveram ao meu lado, apoiando e auxiliando para continuar nesta caminhada.

À minha família, especialmente meu tio, Amilton Barbosa Reis Filho, que incentivou meu desenvolvimento através do esporte e esteve ao lado dos meus pais, auxiliando para que eu pudesse chegar até a universidade.

Aos amigos que fiz durante a faculdade, integrantes do Centro Acadêmico de Engenharia da Computação de 2019, que compartilharam dos momentos mais difíceis do curso e me ajudaram a crescer pessoalmente e profissionalmente. Sem estes amigos, esta caminhada teria sido muito mais árdua.

Agradeço à UFAM pela estrutura de qualidade que me ofereceu, aos professores pelos ensinamentos e experiências compartilhadas e aos meus orientadores: o Prof. Dr. Leandro Silva Galvão de Carvalho, que me apresentou e auxiliou no início deste trabalho, e a Profa. Dra. Elaine Harada Teixeira de Oliveira, que se dispôs a substituir o Prof. Leandro durante sua ausência médica e nos ajudar a finalizar este trabalho.

Uso de problemas de Parson no ensino de programação introdutória

Autor: Anilton Carlos de Lima Reis
Orientadora: Profa. Dra. Elaine Harada Teixeira de Oliveira
Coorientador: Prof. Dr. Leandro Silva Galvão de Carvalho

Resumo

Este trabalho tem como objetivo explorar o uso de problemas de Parson no ensino de introdução à programação. Os problemas de Parson são um tipo de exercício de programação baseado em arrastar e soltar blocos. Os alunos recebem blocos contendo trechos de código desorganizados e precisam organizá-los corretamente para solucionar o desafio proposto. Esse tipo de atividade tem o objetivo de reduzir a carga cognitiva dos problemas capazes de sobrecarregar os alunos novatos, como por exemplo escrever o código do zero. Neste trabalho foram catalogados diferentes tipos ou variações de problemas de Parson através de um mapeamento sistemático da literatura (MSL) e os de ganho de aprendizagem relatados em pesquisas.

Palavras-chave: Problemas de Parson, Introdução à programação, Carga cognitiva, Ganho de aprendizagem.

Uso de problemas de Parson no ensino de programação introdutória

Autor: Anilton Carlos de Lima Reis
Orientadora: Profa. Dra. Elaine Harada Teixeira de Oliveira
Coorientador: Prof. Dr. Leandro Silva Galvão de Carvalho

Abstract

This work aims to explore the use of Parson's problems in teaching introductory programming. Parson's problems are a drag-and-drop type of programming exercise. Students receive blocks containing unorganized code snippets and need to organize them correctly to solve the proposed challenge. This type of activity aims to reduce the cognitive load of problems that can overwhelm novice students, such as writing code from scratch. In this work, different types or variations of Parson's problems were cataloged through a systematic literature mapping (MSL) and learning gains reported in research.

Keywords: Parson's problems, Introductory programming, Cognitive load, Learning gain.

Lista de figuras

Lista de tabelas

Lista de abreviaturas e siglas

MSL	Mapeamento Sistemático da Literatura

Sumário

1 Introdução

O ensino de programação é um grande desafio para os educadores. Os cursos de introdução a programação devem ensinar lógica de programação, sintaxe e semântica, estruturas de dados, depuração e entre outros assuntos que formam a base da computação. Tradicionalmente, em cursos universitários os alunos praticam programação ao escrever códigos e, dependendo do nível técnico, este tipo de atividade pode levar bastante tempo para ser concluído.

Diversas abordagens e ferramentas de ensino têm sido desenvolvidas para facilitar a compreensão e o aprendizado dos alunos nas disciplinas de computação. O artigo de Dale Parsons e Patricia Haden (PARSONS; HADEN, 2006) descreve um tipo de exercício interativo de ensino de programação utilizando quebra-cabeças. As questões apresentavam a solução dividida em vários blocos desorganizados e o objetivo dos alunos era organizá-los corretamente. Esse tipo de exercício foi intitulado de *Parson's Programming Puzzles* e é comumente chamado de problema de Parson.

1.1 Contextualização ou definição do problema

Um problema de Parson é um tipo de exercício de programação que consiste em ordenar blocos de códigos. O exercício pode ser apresentado de diferentes formas, com o objetivo de impulsionar a aprendizagem dos alunos em programação. Este trabalho pretende identificar os tipos de problemas de Parson.

1.2 Objetivos

1.2.1 Objetivos gerais

Este trabalho visa realizar um mapeamento sistemático da literatura (MSL) sobre os problemas de Parson com o propósito de apresentar os diferentes tipos ou abordagens desses exercícios utilizados no ensino de programação.

1.2.2 Objetivos específicos

- Catalogar os diferentes tipos de problemas de Parson existentes.
- Apresentar exemplos ilustrativos de cada tipo de problema.
- Apresentar os relatos de aprendizagem encontrados.

1.3 Organização do Trabalho

Este trabalho está estruturado da seguinte forma: No Capítulo 2 é apresentada uma revisão bibliográfica sobre os problemas de Parson no ensino de programação.

Em seguida, o Capítulo 3 descreve a MSL, metodologia utilizada para catalogar os trabalhos publicados sobre Problemas de Parson e analisá-los a fim de responder às questões de pesquisa definidas neste capítulo.

No Capítulo 4, apresentam-se os resultados obtidos a partir da análise dos trabalhos selecionados e as respostas às questões de pesquisa.

2 Fundamentação teórica

2.1 Taxa de aprovação em Introdução à Computação

De acordo com os estudos realizados por Jens Bennedsen e Michael Caspersen, a média de aprovação em disciplinas de introdução à programação em instituições de ensino de todo o mundo subiu de 67% em 2007 (BENNEDSEN; CASPERSEN, 2007) para 72,6% em 2019 (BENNEDSEN; CASPERSEN, 2019). Esse aumento pode ser atribuído ao desenvolvimento de metodologias de ensino mais eficazes, o aprimoramento dos recursos, ferramentas e materiais didáticos disponíveis e a crescente popularização da programação.

Apesar do avanço, ainda é possível aumentar o percentual de aprovação. Dentre as reprovações relatadas, 10,7% dos alunos desistem antes de fazer o exame final. Vários motivos podem influenciar a desistência, sendo um deles a frustração por não superar as dificuldades da programação à primeira vista (BENNEDSEN; CASPERSEN, 2019).

2.2 Teoria da Carga Cognitiva

No mercado de trabalho em computação, é esperado que se tenha domínio dos conceitos básicos de programação, como a habilidade de ler e escrever códigos. Portanto, os alunos de cursos de computação precisam praticar a escrita de códigos desde o começo para incentivar o desenvolvimento de materiais autênticos. Porém, sem a devida orientação os alunos podem se frustrar ao passarem tempo demais tentando fazer um código funcionar, seja por se prenderem a erros de lógica ou erros de sintaxe referentes à linguagem escolhida (ERICSON; MARGULIEUX; RICK, 2017).

De acordo com a Teoria da Carga Cognitiva, a mente humana possui uma capacidade limitada de processamento de informações. Essa capacidade é chamada de carga cognitiva. Quando a carga cognitiva exigida para resolver

determinado problema é maior do que a capacidade mental, o desempenho e a absorção de informações é prejudicada. Logo, se a carga cognitiva for gerenciada adequadamente, o aprendizado pode se tornar eficiente (SWELLER, 1988).

2.2.1 Exercícios de conclusão

Baseado na teoria de Sweller, algumas estratégias foram se aprimorando para potencializar o desenvolvimento dos estudantes, oferecendo cada vez mais suporte para uma prática progressiva. Escrever um código do zero pode ser uma tarefa muito complexa para alunos não habituados a este exercício. Uma estratégia é utilizar exercícios de conclusão, na qual são fornecidas partes da solução e os alunos são instruídos a completá-la com a resposta apropriada (MERRIENBOER, 1990).

Como estratégia para o ensino de programação, utilizar exercícios de conclusão permite que os alunos se concentrem apenas em aprender os conceitos de programação e se familiarizar com a estrutura e a lógica de um programa. Esta estratégia é mais indicada para alunos que possuem pouca experiência em programação, servindo como direcionamento para que construam uma base sólida e posteriormente construam suas soluções de maneira autônoma.

2.3 Problemas de Parson

2.3.1 Definição e Características

A primeira aparição dos problemas de Parson ocorreu no artigo publicado por Parsons e Haden (2006), que descreve uma ferramenta de ensino automatizada para utilizar em classes de introdução à programação. Esta ferramenta aplica um conjunto de exercícios do estilo arrastar e soltar blocos, na qual são fornecidos blocos desorganizados de um código para que os alunos rearranjem corretamente e formem a solução do problema. Estes exercícios

são chamados de problemas de Parson.

Um problema de Parson disponibiliza a solução do problema solicitando que seja organizada, portanto é um exercício de conclusão. Os alunos não precisam se preocupar em escrever o programa inteiro, mas se concentrar apenas em organizar e entender a solução de forma prática.

De acordo com Parsons e Haden (2006), um problema de Parson possui as seguintes características:

- **Maximização do engajamento:** A estruturação do exercício como um jogo de quebra-cabeças consegue atrair a atenção dos alunos com mais facilidade.
- **Restrição lógica:** O exercício fornece as instruções do código e o aluno pode focar na estrutura sequencial e lógica de controle, identificando a sequência correta para formar a solução.
- **Permitir erros comuns:** É normal que os alunos cometam erros de sintaxe várias vezes ao escrever um código. Os Problemas de Parson ainda podem permitir que estes erros ocorram ao inserir blocos de códigos contendo soluções erradas para que os alunos visualizem diretamente os erros comuns que cometem e como a construção correta do código deve parecer.
- **Boas práticas:** Ao fornecer diversos exemplos corretos de construção de código, os alunos são expostos a boas práticas de construção de código.
- ***Feedback* imediato:** Os Problemas de Parson fornecem uma resposta imediata aos alunos, permitindo que identifiquem suas escolhas erradas e compreendam as implicações geradas por estas escolhas.

2.3.2 Estrutura

Um problema de Parson é estruturado como se fosse um jogo de quebra-cabeça, onde a resposta é obtida após ordenar corretamente os blocos de código

disponíveis. As Figuras 1 e 2 ilustram a estrutura básica de um Problema de Parson que consiste em:

1. Especificação do problema;
2. Conjunto de blocos de código desordenados;
3. Espaço em branco para posicionar os blocos e ordenar a solução;
4. *Feedback* baseado em linha.
5. *Feedback* baseado em execução.

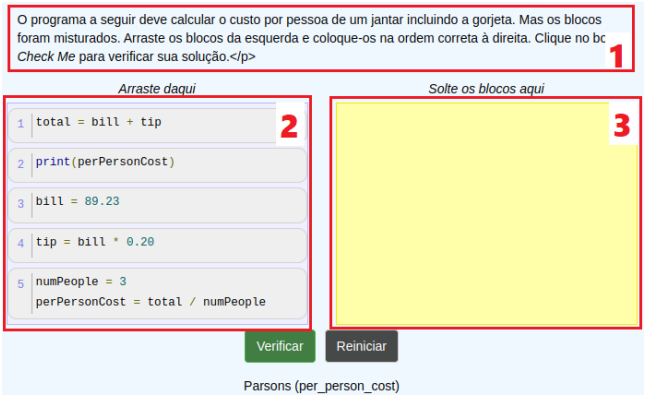


Figura 1 – Exemplo de exercício utilizando a abordagem de Problema de Parson (RUNESTONE, 2018).

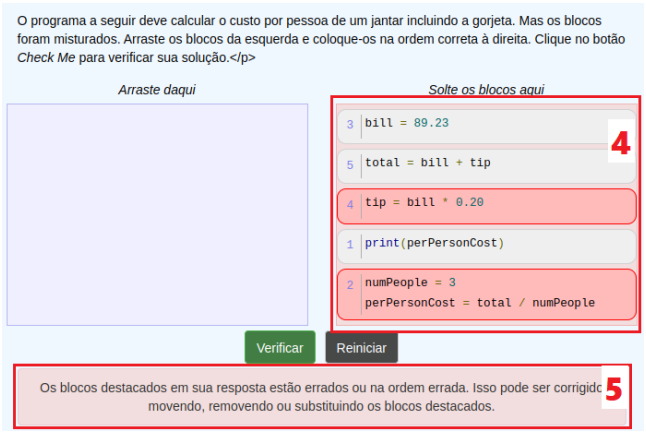


Figura 2 – Exemplo de exercício utilizando a abordagem de Problema de Parson (RUNESTONE, 2018).

2.3.2.1 Programação baseada em Blocos

Apesar de serem parecidas, a programação baseada em blocos e os problemas de Parson são exercícios bem diferentes, mas que visam auxiliar o aprendizado em programação para alunos iniciantes. A programação em blocos facilita a visualização e manipulação de vários conceitos de programação, enquanto os problemas de Parson enfatizam a compreensão da lógica e estrutura de código ao limitar a variedade de conceitos e diminuir a dificuldade do exercício.

A programação baseada em blocos possui uma variedade muito maior de blocos para representar os diferentes comandos e estruturas da programação. A Figura 3 ilustra o ambiente de desenvolvimento da plataforma de programação e modelagem 3D Tinkercad, na qual os usuários podem utilizar a interface de arrastar e soltar blocos para combiná-los e construir programas que controlam o comportamento de componentes eletrônicos (TINKERCAD, 2021).

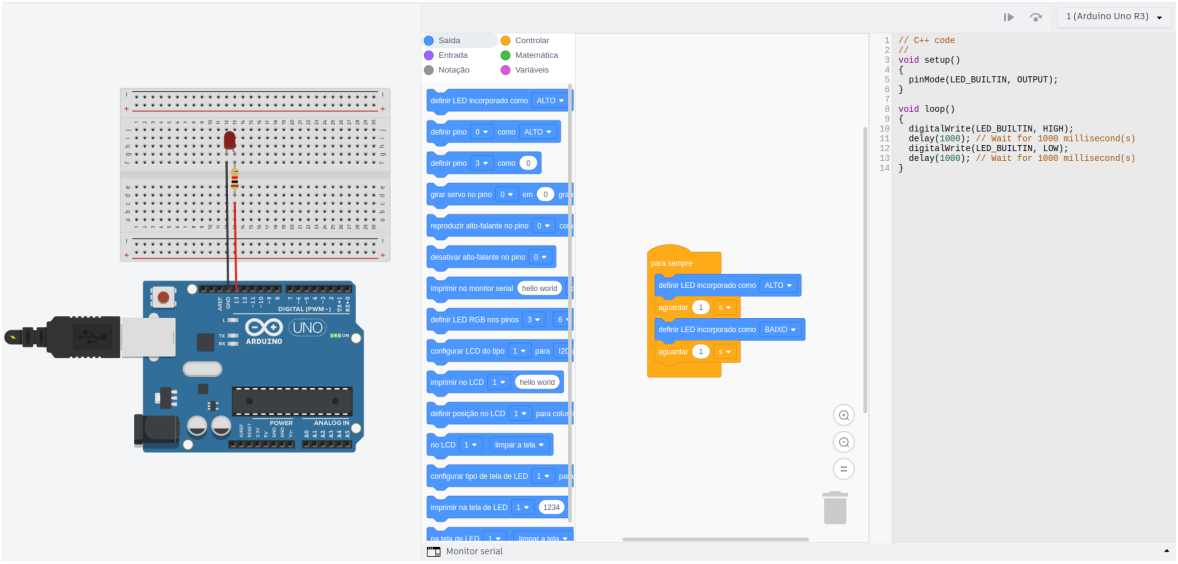


Figura 3 – Ambiente de programação em blocos do Tinkercad (TINKERCAD, 2021).

3 Metodologia

3.1 Revisão sistemática da literatura

Neste trabalho foi conduzido um mapeamento sistemático da literatura (MSL), uma metodologia que visa mapear e sintetizar as evidências disponíveis sobre um determinado tema de estudo (PETERSEN et al., 2008).

Um mapeamento sistemático da literatura é dividido em três fases principais: planejamento, condução e relatório (PETERSEN et al., 2008). A fase de planejamento consiste em definir questões de pesquisa relacionadas ao tema de estudo e desenvolver um protocolo de revisão. Na segunda fase acontece a identificação da pesquisa, seleção e avaliação de estudos e a extração e síntese de dados dos estudos selecionados. A fase de relatório consiste em organizar os resultados para publicação.

3.1.1 Questões de Pesquisa

O objetivo deste trabalho é realizar descobrir quais tipos de problemas de Parson são utilizados no ensino de programação e quais os benefícios relatados pelo uso desta ferramenta de ensino. Portanto, as seguintes questões de pesquisa foram definidas:

1. Quais os tipos ou variantes de problemas de Parson têm sido empregadas no ensino de programação introdutória?

2. Quais os ganhos de aprendizado de programação têm sido relatados no uso de problemas de Parson?

3.1.2 Protocolo de revisão

O protocolo de revisão visa definir termos de busca para utilizar na extração de literaturas em bibliotecas digitais. O artigo de Du, Luxton-Reilly e Denny (2020) é uma revisão da literatura de problemas de Parson e foi encontrado a partir de uma busca exploratória. Este artigo foi utilizado como guia para escrever a fundamentação teórica e serviu para extração de palavras-chave e a definição da seguinte *string* de busca:

Abstract:((“parson” OR “parsons” OR “parson’s”) AND (“problem” OR “problems” OR “puzzle” OR “puzzles” OR “programming” OR “problemas” OR “quebra-cabeça”))

Esta *string* de busca foi utilizada no banco de dados ACM Digital Library. Ela está limitada a encontrar as palavras-chave nos resumos de cada publicação para prevenir resultados que contenham apenas autores com o nome ou sobrenome Parson e que não abordam o tema deste trabalho.

A mesma *string* de busca foi utilizada no banco de dados da IEEE Xplore, mas não se mostrou eficiente ao listar diversos trabalhos contendo apenas uma ou outra palavra, por exemplo “*problems*” ao invés de “*parsons problems*”. Portanto, os conectivos “*AND*” foram removidos para utilizar somente os conectivos “*OR*” da seguinte forma:

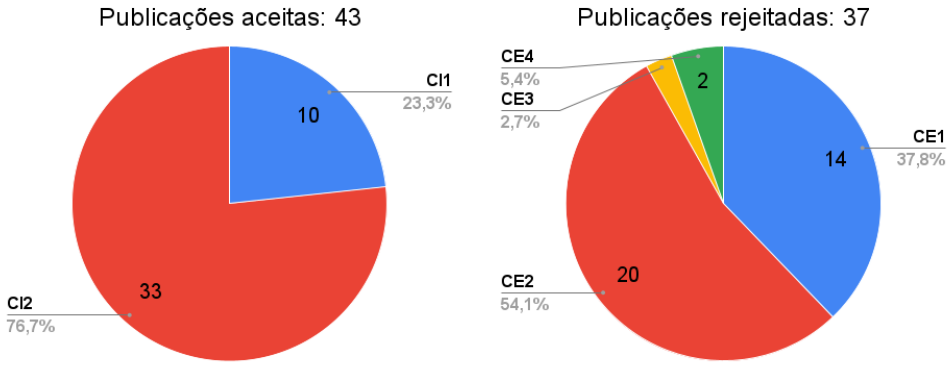
Abstract:(“parson problems” OR “parson puzzles” OR “parson programming” OR “parson’s problems” OR “parson’s puzzles” OR “parson’s programming” OR “parsons programming” OR “problemas de parson” OR “quebra-cabeça de parson”)

Ao todo foram obtidas 80 publicações resultantes, sendo 73 a partir da base de dados da ACM Digital Library e apenas 7 da IEEE Xplore. Nas próximas seções serão descritas as duas filtragens realizadas nas publicações obtidas.

3.1.3 Análise de critérios

Nesta etapa, as 80 publicações resultantes tiveram seus títulos e resumos examinados para determinar se eram relevantes para revisão a partir de 2 critérios de inclusão e 4 de exclusão. Os critérios são os seguintes:

- Critérios de inclusão:
  - CI1: A publicação descreve ou aborda algum tipo de problema de Parson;
  - CI2: A publicação relata sobre o uso de problemas de Parson.
- Critérios de exclusão:
  - CE1: A publicação não é focada em problemas de Parson;
  - CE2: *Short paper*;
  - CE3: A publicação não está disponível para acesso;
  - CE4: Publicação duplicada.



(a) Critérios de inclusão. (b) Critérios de exclusão.

Figura 4 – Resultados da análise de critérios dos trabalhos encontrados.

Os gráficos das Figuras 4a e 4b ilustram os resultados da análise de critérios, na qual 43 publicações foram aprovadas para a próxima análise descrita na Seção 3.1.4. Dentre as publicações rejeitadas, 20 foram pelo critério de publicações com conteúdo limitado, 2 páginas ou menos, geralmente conteúdos resumidos para apresentação em eventos.

O gráfico da Figura 5 ilustra o ano das publicações que foram tanto aprovadas quanto recusadas na análise de critérios. É possível notar o aumento anual da quantidade de publicações referentes aos problemas de Parson. Esta MSL foi finalizada no mês de junho de 2023, sendo o possível motivo da redução da quantidade de publicações comparada ao ano anterior.

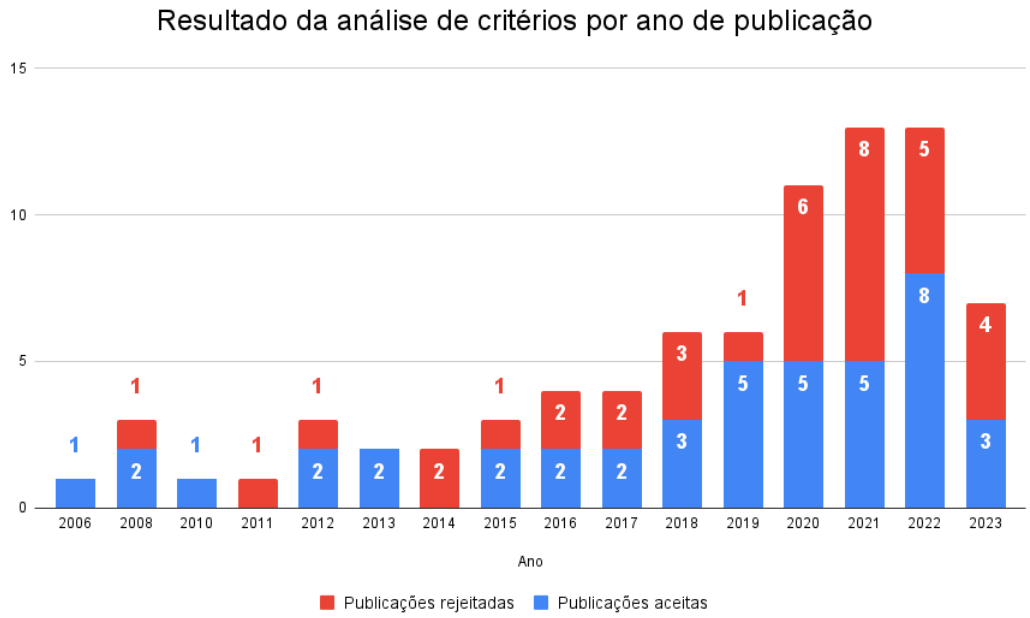


Figura 5 – Resultado da análise de critérios por ano de publicação.

3.1.4 Análise de qualidade

Nesta etapa, as 43 publicações aprovadas pela análise de critérios foram reanalisadas para determinar a qualidade do conteúdo com relação aos objetivos de pesquisa deste mapeamento sistemático da literatura.

Para avaliar a qualidade do conteúdo das publicações foram definidas as seguintes questões:

- Q1: A publicação descreve ou aborda um tipo específico de Problema de Parson?

- Q2: A publicação descreve o uso de problemas de Parson em geral na Computação?
- Q3: A publicação relata sobre os ganhos de aprendizagem relacionados ao uso de problemas de Parson?
- Q4: Considerando o tempo de publicação e quantidade de citações, o estudo é relevante?

A primeira pergunta verifica se algum tipo de problema de Parson é abordado pela publicação a fim de responder diretamente à primeira questão de pesquisa da MSL. As duas próximas perguntas verificam se a publicação descreve sobre como os problemas de Parson têm sido utilizados e quais os ganhos de aprendizagem são relatados. A quarta e última pergunta verifica se a publicação é relevante baseando-se em suas métricas.

Cada pergunta poderia ser respondida com sim (1 ponto), parcialmente (0,5 pontos) e não (0 pontos). Ao todo, cada publicação poderia somar no máximo 4 pontos e as que obtiveram nota maior do que 2 pontos foram consideradas com qualidade suficiente para a extração e síntese de dados. A criação das perguntas e a atribuição das pontuações foram realizadas pelo autor deste trabalho e pode ser visualizada na Tabela 1, no qual 27 de 43 publicações foram aprovadas conforme ilustrado na Figura 6. No próximo capítulo serão apresentados os resultados obtidos após a extração de dados das publicações aprovadas na análise de qualidade, respondendo às questões de pesquisa.

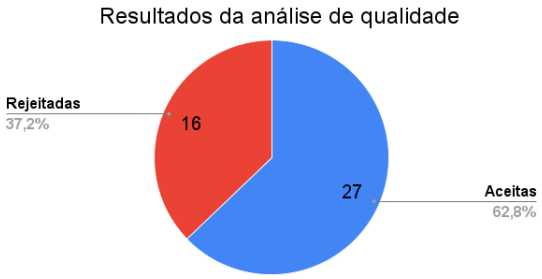


Figura 6 – Resultados da análise de qualidade.

Publicação	Q1	Q2	Q3	Q4	Pontos	Situação
(KARAVIRTA; HELMINEN; IHANTOLA, 2012)	1,0	1,0	0,0	1,0	3,0	Aceita
(DU; LUXTON-REILLY; DENNY, 2020)	0,5	1,0	1,0	1,0	3,5	Aceita
(DICHEVA; HODGE, 2018)	0,0	0,5	0,5	1,0	2,0	Rejeitada
(WAITE et al., 2021)	0,0	1,0	0,0	0,5	1,5	Rejeitada
(WANG et al., 2020)	0,0	0,5	0,5	1,0	2,0	Rejeitada
(SMITH; ZILLES, 2023)	1,0	1,0	1,0	1,0	4,0	Aceita
(HARMS; CHEN; KELLEHER, 2016)	0,5	0,5	1,0	1,0	3,0	Aceita
(KUMAR, 2018)	0,0	0,5	0,5	1,0	2,0	Rejeitada
(DENNY; LUXTON-REILLY; SIMON, 2008)	0,0	1,0	1,0	1,0	3,0	Aceita
(ZHI et al., 2019)	0,0	1,0	1,0	1,0	3,0	Aceita
(ERICSON; FOLEY; RICK, 2018)	1,0	0,5	1,0	1,0	3,5	Aceita
(WEINMAN; FOX; HEARST, 2020)	1,0	0,0	0,0	0,5	1,5	Rejeitada
(FROMONT; JAYAMANNE; DENNY, 2023)	1,0	0,5	1,0	1,0	3,5	Aceita
(KUMAR, 2019a)	0,5	1,0	1,0	1,0	3,5	Aceita
(HELMINEN et al., 2012)	0,0	1,0	1,0	1,0	3,0	Aceita
(IHANTOLA; HELMINEN; KARAVIRTA, 2013)	0,0	1,0	0,5	1,0	2,5	Aceita
(WEINMAN; FOX; HEARST, 2021)	1,0	0,5	1,0	1,0	3,5	Aceita
(OYELERE; SUHONEN; LAINE, 2017)	0,0	1,0	0,0	0,5	1,5	Rejeitada
(ERICSON; MCCALL; CUNNINGHAM, 2019)	1,0	1,0	1,0	1,0	4,0	Aceita
(BENDER et al., 2022)	0,5	1,0	1,0	1,0	3,5	Aceita
(KUMAR, 2019b)	0,0	1,0	0,5	0,5	2,0	Rejeitada
(IHANTOLA; KARAVIRTA, 2010)	0,0	1,0	0,0	1,0	2,0	Rejeitada
(PARSONS; HADEN, 2006)	1,0	1,0	1,0	1,0	4,0	Aceita
(ERICSON et al., 2022)	1,0	1,0	1,0	1,0	4,0	Aceita
(LOPEZ et al., 2008)	0,0	0,0	0,0	0,5	0,5	Rejeitada
(ERICSON; MILLER, 2020)	0,5	1,0	0,5	1,0	3,0	Aceita
(ERICSON; MARGULIEUX; RICK, 2017)	0,5	1,0	1,0	1,0	3,5	Aceita
(LOJO; FOX, 2022)	1,0	1,0	0,0	0,5	2,5	Aceita
(HAYNES-MAGYAR; ERICSON, 2022)	1,0	1,0	1,0	1,0	4,0	Aceita
(HAYNES, 2020)	0,5	0,5	0,0	0,0	1,0	Rejeitada
(GOLETTI; MENS; HERMANS, 2021)	0,0	0,5	0,5	1,0	2,0	Rejeitada
(MULDNER; JENNINGS; CHIARELLI, 2022)	0,5	0,5	0,5	0,5	2,0	Rejeitada
(ERICSON; HAYNES-MAGYAR, 2022)	0,5	1,0	0,5	1,0	3,0	Aceita
(ERICSON; GUZDIAL; MORRISON, 2015)	0,5	1,0	0,5	1,0	3,0	Aceita
(SIRKIä, 2016)	1,0	0,5	1,0	1,0	3,5	Aceita
(ERICSON; MAEDA; DHILLON, 2022)	0,5	1,0	1,0	1,0	3,5	Aceita
(HELMINEN et al., 2013)	0,5	1,0	0,5	0,5	2,5	Aceita
(KARAVIRTA et al., 2015)	0,0	0,5	0,5	1,0	2,0	Rejeitada
(HAYNES; ERICSON, 2021)	0,5	1,0	1,0	1,0	3,5	Aceita
(THOMAS et al., 2019)	0,0	0,5	0,0	0,5	1,0	Rejeitada
(HAYATPUR et al., 2023)	0,5	0,5	0,5	0,5	2,0	Rejeitada
(MORRISON et al., 2016)	0,5	1,0	0,5	0,5	2,5	Aceita
(STEPHENSON; MANGAT, 2021)	0,0	1,0	0,0	0,5	1,5	Rejeitada
Total de publicações aceitas					27	
Total de publicações rejeitadas					16	

Tabela 1 – Análise de qualidade dos trabalhos aprovados na análise de critérios.

4 Resultados

Neste capítulo, os 27 trabalhos aprovados na análise de qualidade da Seção 3.1.4 tiveram seus dados extraídos para responder às questões de pesquisa da revisão sistemática.

4.1 Questão 1: Variantes de problemas de Parson

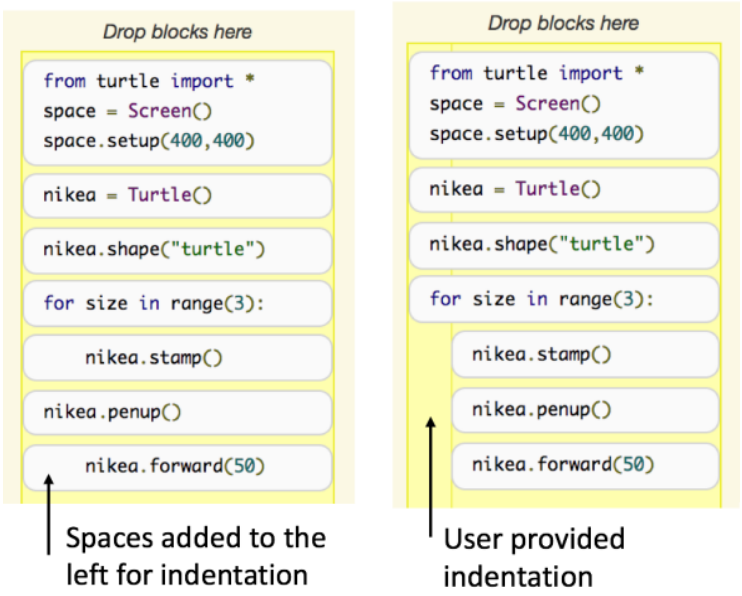
Nesta seção são apresentadas as respostas obtidas para a primeira questão de pesquisa: "Quais os tipos ou variantes de problemas de Parson têm sido empregadas no ensino de programação introdutória?".

4.1.1 Unidimensionais e bidimensionais

Os problemas de Parson podem ser unidimensionais, caracterizados por questões de organização vertical dos blocos conforme ilustrado na Figura 7a, e bidimensionais onde se é necessário uma organização vertical e horizontal dos blocos conforme ilustrado na Figura 7b.

Os problemas unidimensionais são mais comuns em linguagens cuja indentação não é obrigatória e a delimitação dos blocos de código é explícita, como em Java. Já os problemas bidimensionais são comuns em linguagens como Python, cuja delimitação dos blocos de código é realizada por espaços ou tabulações.

Mesmo que a linguagem tenha a delimitação explícita dos blocos de código, ela pode ser utilizada em problemas bidimensionais ao criar blocos contendo apenas as delimitações de código. A Figura 8 ilustra um problema de Parson bidimensional em Java, no qual os blocos 1 e 4 contém os delimitadores de fim de bloco de código.



(a) Unidimensional. (b) Bidimensional.

Figura 7 – Problema de Parson unidimensional versus bidimensional em *Python* (ERICSON; MCCALL; CUNNINGHAM, 2019).

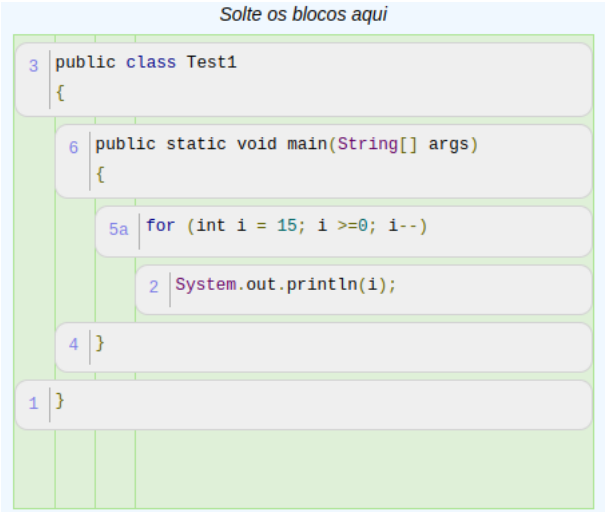


Figura 8 – Solução de um problema de Parson bidimensional em Java (RUNESTONE, 2018).

4.1.2 Distratores

Os distratores são opções de resposta incorretas projetadas para induzir os alunos ao erro. Essas distrações se assemelham com blocos de códigos corretos, mas possuem algum erro sintático ou lógico. O uso de distrações aumenta a carga cognitiva do exercício com o objetivo de desafiar os alunos a analisarem com cuidado todas as opções. A Figura 9 ilustra um conjunto de

blocos de código com erros comuns ao utilizar classes em *Python*. A distração é sinalizada e mostra as duas opções para compor a solução, sendo uma delas incorreta quanto ao uso do "*self*".

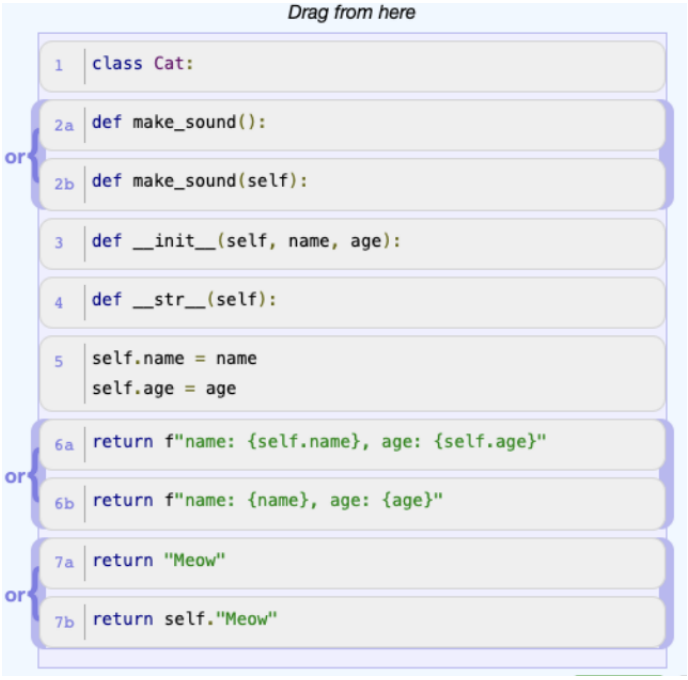


Figura 9 – Problema de Parson com distratores (ERICSON et al., 2022).

Alunos novatos ao trabalharem com problemas de Parson sem distratores têm um desempenho significativamente melhor em termos de tempo necessário para conclusão e precisão nos resultados (HARMS; CHEN; KELLEHER, 2016). Com o aumento da dificuldade vem o aumento de tempo necessário para definir as soluções. Para que as distrações tenham efeito positivo no desenvolvimento dos alunos é preciso que sejam utilizados com cautela, por exemplo, conforme explicado na Subseção 4.1.3 com os problemas com adaptação da dificuldade.

4.1.2.1 Distratores pareados e não pareados

Na abordagem de distratores pareados, os blocos com soluções erradas são posicionados estrategicamente ao redor do bloco semelhante com a solução correta, conforme ilustrado na Figura 10, o bloco distrator 4b está posicionado logo abaixo do bloco correto 4a.

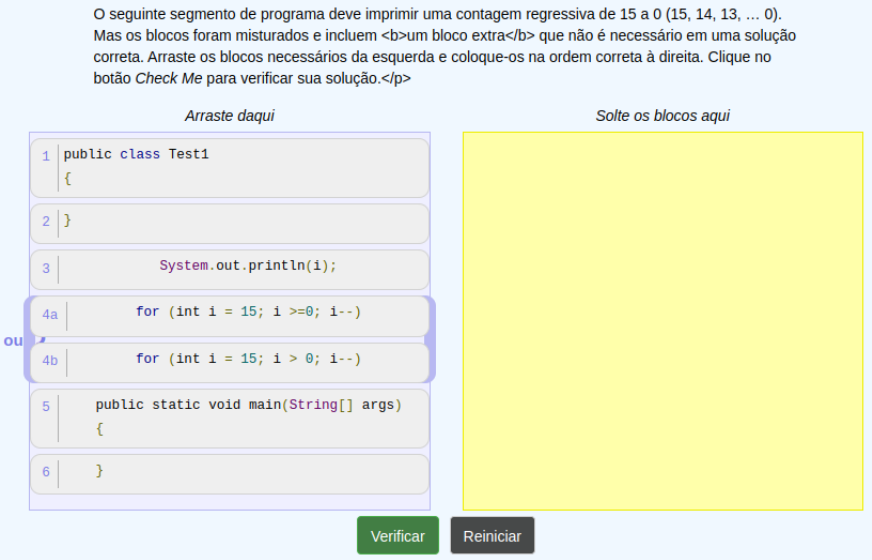


Figura 10 – Problema de Parson com distratores pareados (RUNESTONE, 2018).

Ao contrário dos pareados, os distratores colocados de maneiras aleatórias entre as soluções são chamados de distratores não pareados. Conforme ilustrado na Figura 11, o bloco 2 é um distrator referente ao bloco 7, que altera o resultado da contagem de 15 até 0 para 15 até 1.

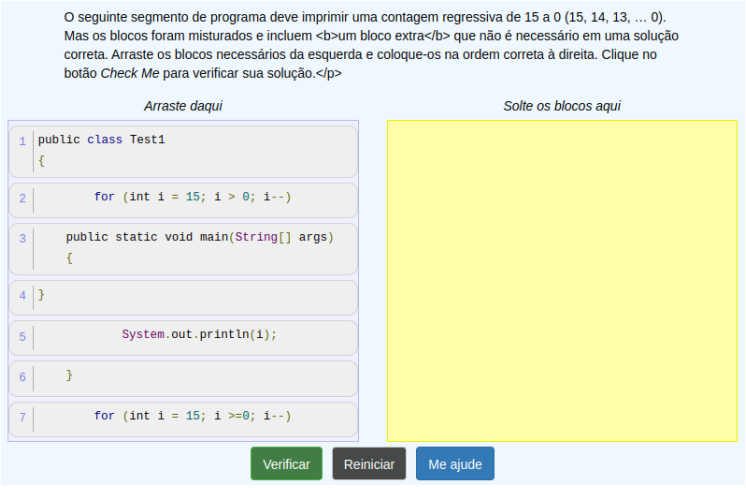


Figura 11 – Problemas de Parson com distratores não pareados (RUNESTONE, 2018).

4.1.3 Problemas de Parson adaptativos

Os problemas de Parson adaptativos alteram dinamicamente a dificuldade dos exercícios de acordo com o desempenho dos alunos. Esta técnica visa manter o engajamento dos alunos e auxiliar no desenvolvimento dos mesmos. O estudo de Ericson, Foley e Rick (2018) apresenta as abordagens de problemas adaptativos intra-problemas e inter-problemas.

Na adaptação intra-problemas, se o aluno estiver com muita dificuldade em solucionar o exercício atual, o exercício pode ser facilitado fornecendo dicas, removendo os distratores ou até auto-combinando os blocos. Na adaptação inter-problemas, o próximo exercício será modificado com base no desempenho do exercício anterior (ERICSON; FOLEY; RICK, 2018). Da mesma forma que os exercícios podem ser facilitados, se os alunos tiverem bons desempenhos, os próximos exercícios podem ser dificultados.

A abordagem de tentativa e erro pode se tornar comum entre os alunos se não houver limitações dos recursos de adaptação intra-problema (SIRKIä, 2016). Portanto é necessário inibir este comportamento impondo limitações de envio de resposta ou solicitações de ajuda. Na Figura 12, o botão “Me ajude” foi disponibilizado após 3 tentativas diferentes sem sucesso e, ao ser clicado, removeu o bloco de distração 2 da solução proposta pelo aluno e desativou o bloco para não ser utilizado novamente.

4.1.4 Problemas de Parson incompletos

Faded Parson problems ou problemas de Parson incompletos é uma abordagem que remove partes do código para que os alunos completem adequadamente (WEINMAN; FOX; HEARST, 2021), conforme ilustrado na Figura 13 que ilustra o uso variáveis e valores incompletos. É uma abordagem que os alunos devem construir ativamente a estrutura e a lógica dos programas.

A remoção de partes dos código é chamada de “*fading*” em inglês e possui algumas estratégias que foram identificadas no artigo de Fromont, Jayamanne

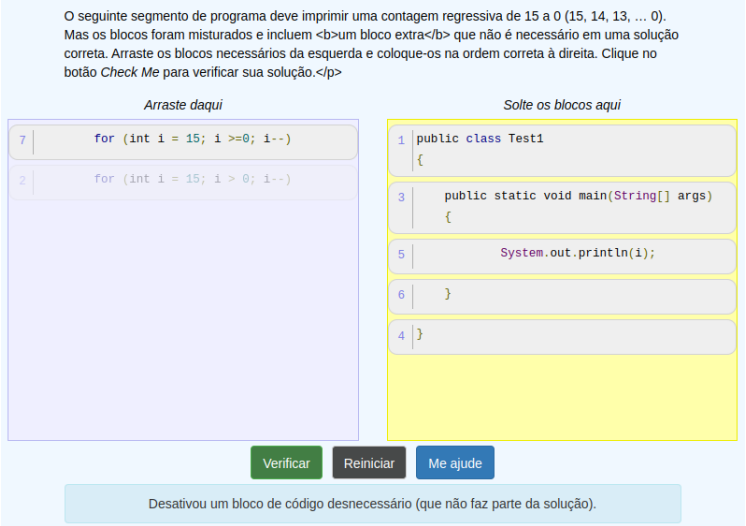


Figura 12 – Problema de Parson adaptativo intra-problema (RUNESTONE, 2018).

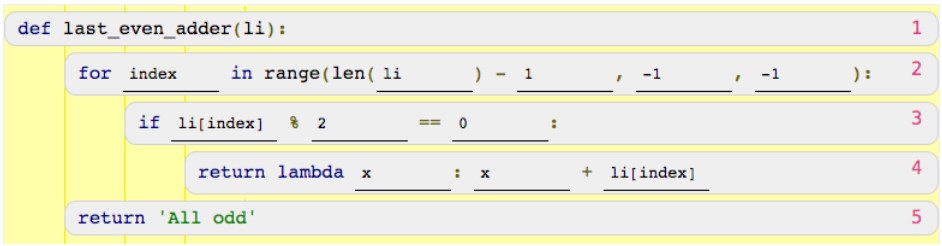


Figura 13 – Problema de Parson incompleto (WEINMAN; FOX; HEARST, 2021).

e Denny (2023) e agrupados na Tabela 2.

Estratégia de "fading"	Descrição	Exemplo
Palavra-chave	Elementos sintáticos de uma linguagem	for, while, if, else, return
Operador	Operadores lógicos ou aritméticos	+ , -, ==, <, &
Variável	Qualquer identificador definido pelo usuário	i, array, element
Condicional	Tanto o operador lógico quanto os operandos	i<=100, array[i]==element
índice	Índice de uma matriz ou coleção	array[i], list.get(i-1)
Função	Quando uma função padrão é chamada	print(..),scanf(..)

Tabela 2 – Tipos de estratégias de *fading* identificadas (FROMONT; JAYAMANNE; DENNY, 2023).

4.1.5 Problemas de Parson horizontais

Essa abordagem apresenta instruções e estruturas de código que devem ser ordenadas em uma única linha para formar um programa funcional. A

Figura 14 ilustra um exemplo de problema horizontal com o objetivo de montar uma expressão regular que inicie com “ab” e termine com 0 ou mais “c”.

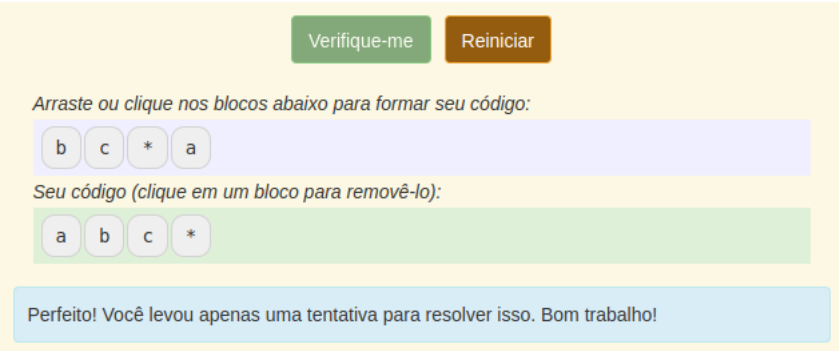


Figura 14 – Problema de Parson horizontal (RUNESTONE, 2018).

Neste capítulo apresentamos os diferentes tipos de problemas de Parson que podem se adequar a qualquer linguagem de programação. Os problemas adaptativos alteram a dificuldade dos exercícios ao utilizar de outros tipos de problemas de Parson e, na próxima seção, serão abordados os relatos dos pesquisadores ao utilizar dos problemas de Parson e suas variações.

4.2 Questão 2: Ganhos de aprendizado

Nesta seção são apresentadas as respostas obtidas para a segunda questão de pesquisa “Quais os ganhos de aprendizado de programação têm sido relatados no uso de problemas de Parson?”. Inicialmente será mostrado um breve resumo sobre as ferramentas abordadas nos trabalhos analisados e, em seguida, serão apresentados os relatos sobre o uso de problemas de Problemas de Parson e suas variações no ensino de programação.

4.2.1 Ferramentas

Desde que os problemas de Parson surgiram, diversas ferramentas para implementá-los foram desenvolvidas tanto para sistemas *Web* quanto *Mobile*. Quando surgiram, os problemas de Parson eram desenvolvidos a partir do

software *Hot Potatoes* (PARSONS; HADEN, 2006). Em 2010, foi desenvolvida a biblioteca *js-parsons* que oferece suporte a problemas de Parson bidimensionais e *feedback* baseado em linha (IHANTOLA; KARAVIRTA, 2010). A biblioteca *js-parsons* é comumente utilizada para implementação de problemas de Parson em diversos ambientes de aprendizagem *Web*, como por exemplo a plataforma *Runestone* ilustrada na Figura 15.

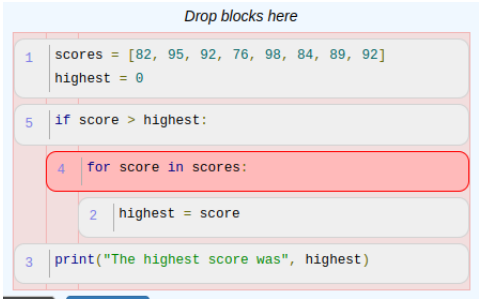


Figura 15 – Interface de problemas de Parson no ambiente *Runestone*. (ZHI et al., 2019).

O artigo de Karavirta, Helminen e Ihantola (2012) apresenta o aplicativo MobileParsons, desenvolvido sobre a biblioteca *js-parsons*. O aplicativo oferece suporte a problemas de Parson com *feedback* automático e redesenha a interface de usuário para ser funcional em dispositivos móveis. O uso de aplicativos consegue transferir os ganhos de aprendizagem de problemas de Parson para o ambiente móvel.

O artigo de Zhi et al. (2019) apresenta a inserção de problemas de Parson no ambiente de programação baseado em blocos chamado *Snap!*, ilustrada na Figura 16. Segundo os resultados, os problemas de Parson ajudam os alunos a evitar erros de sintaxe ao reduzir drasticamente o espaço de solução de programação.

O trabalho de Ericson, Guzdial e Morrison (2015) apresenta uma pesquisa sobre a inserção de recursos interativos em livros virtuais ou *e-books*. Além de textos, imagens, vídeos e áudios, é possível adaptar os exercícios práticos de seleção e múltipla escolha como os problemas de Parson. Neste trabalho é relatado que dentre todos os novos recursos adicionados em um *e-book* de teste, alguns foram mais usados do que outros, sendo os problemas

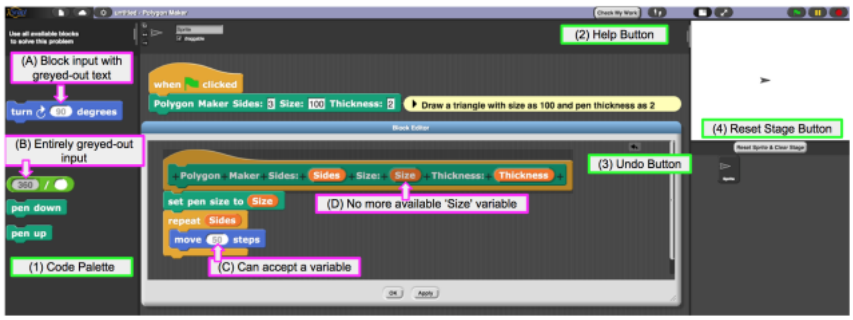


Figura 16 – Interface de problemas de Parson no ambiente *Snap!*. (RUNESTONE, 2018).

de Parson um destes recursos. Utilizar *e-books* interativos pode auxiliar na detecção das dificuldades apresentadas pelos alunos, assim como prestar auxílio dinamicamente com outros materiais ativos disponíveis, conforme é estudado no trabalho de Ericson, Maeda e Dhillon (2022).

4.2.2 Problemas de Parson e escrita de código

O estudo realizado por Ericson, Margulieux e Rick (2017) comparou as atividades de problemas de Parson com as tradicionais atividades de correção e escrita de código. Os resultados mostraram que os alunos do grupo de problemas de Parson obtiveram melhor desempenho em testes de aplicação prática e compreensão dos conceitos do que o grupo de alunos que realizou as atividades de correção e escrita de código.

De maneira similar, o estudo realizado por Haynes e Ericson (2021) identificou que os alunos eram mais eficientes ao solucionarem problemas de Parson adaptativos do que escrever o código equivalente, exceto quando a solução apresentada fosse incomum. Os alunos relataram que gostariam que existisse um recurso similar em problemas de escrita de código, destacando os erros ou fornecendo dicas e recursos para solucionar as dificuldades.

4.2.3 Uso de distratores

O trabalho realizado por Harms, Chen e Kelleher (2016) comparou a eficiência dos problemas de Parson com e sem distratores em um grupo de 92 alunos de 10 a 15 anos de idade. Durante uma única sessão de 2 horas, os alunos foram divididos entre o grupo sem distratores, com 47 pessoas, e o grupo com distratores, 45 pessoas.

Os grupos foram submetidos a uma fase de treinamento com o objetivo de se familiarizar com as tarefas e, posteriormente, submetidos à fase de provas. Os resultados mostram que, na fase de treinamento, todos do grupo de distratores utilizaram blocos distratores em pelo menos 2 das 6 atividades, e 84% usaram em 4 ou mais das mesmas 6 atividades. Na fase de treinamento, os blocos distratores diminuíram a taxa de sucesso das atividades em 26% e aumentou o tempo de resposta em 14%. Na fase de provas, o estudo mostrou que os dois grupos gastaram aproximadamente a mesma quantidade para finalizar os exercícios e que não houve diferença significativa entre os desempenhos, com a quantidade similar de tarefas corretas (HARMS; CHEN; KELLEHER, 2016).

O artigo de Smith e Zilles (2023) descreve a criação de uma ferramenta que gera distratores automaticamente. Um conjunto de modelos de distratores foi construído a partir da análise de respostas erradas obtidas em um questionário de tarefas simples. O questionário solicitava a escrita de uma linha de código para realizar uma simples tarefa, por exemplo inserir um elemento em uma lista. Após a análise de desempenho de um exame final com distratores não pareados realizado em um curso de introdução à programação em *Python* com 494 alunos, os resultados sugerem que não houve impacto significativo na pontuação obtida pelos alunos, mas que houve aumento significativo do tempo de duração total do exame.

4.2.4 Uso de problemas de Parson incompletos

O artigo de Fromont, Jayamanne e Denny (2023) descreve uma nova ferramenta para resolver problemas de Parson chamada (*Unnamed Parsons Problem Tool*). O artigo também apresenta um estudo realizado com 915 alunos de um curso de introdução à programação em C, na qual foram submetidos a 4 exercícios com diferentes estratégias de *fading* para problemas de Parson incompletos: condicional, variável, operador e um exercício padrão sem *fading*. Os resultados mostraram que o exercício sem *fading* levou menos tempo para ser resolvido quando comparado com as demais estratégias e, portanto, sendo considerado como mais fácil que os demais. Sendo assim, os resultados de tempo mostraram que a estratégia condicional foi a mais difícil dentre as três, seguida pelas estratégias de *fading* de variável e operador, consecutivamente.

4.2.5 Uso de problemas de Parson Adaptativos

O artigo de Ericson, Foley e Rick (2018) comparou a eficiência e a eficácia dos problemas de Parson adaptativos com relação aos problemas não adaptativos e problemas de escrita de código. A pesquisa dividiu 126 estudantes de introdução à programação em quatro grupos de análise: (1) grupo de problemas adaptativos, (2) não adaptativos, (3) escrita de código e (4) grupo de controle com problemas adaptativos fora da tarefa.

Os resultados mostraram que o tempo gasto pelos grupos 1 e 2 são aproximadamente iguais e são significativamente menores do que o tempo gasto pelo grupo 3 para escrever os códigos. A pesquisa realizou um pré-teste e um pós-teste imediato, e notou que houve melhora significativa na pontuação dos dois testes, mas que não houve tanta diferença ao comparar a pontuação entre as três condições de tarefa. Portanto, a pesquisa indica que os ganhos de aprendizagem ao resolver problemas de Parson adaptativos e não adaptativos são equivalentes aos ganhos da escrita de código.

O estudo observacional de Ericson, McCall e Cunningham (2019) foi

realizado com 11 professores do ensino médio que possuem pouca experiência com programação e com idades entre 26 e 59 anos. A pesquisa identificou que 9 dos 11 professores tiveram preferência pelos problemas de Parson adaptativos a não adaptativos. Os professores relataram que utilizar problemas de Parson os ajudou a corrigir e escrever códigos semelhantes, reconhecendo erros comuns de sintaxe. Apesar de que as respostas condizem com estudos anteriores, os resultados obtidos podem ter sido influenciados pelo tipo de estudo ser observacional.

O artigo de Haynes-Magyar e Ericson (2022) realizou um experimento com 95 alunos da graduação para averiguar se utilizar problemas de Parson com soluções de escrita mais comuns tornaria o problema de escrita de código mais eficiente de resolver. Todos os alunos foram submetidos a avaliações com problemas de Parson e escrita de código. Os resultados mostraram que os alunos foram mais eficientes na resolução de problemas de Parson que utilizaram a solução mais comum dos alunos em comparação com os problemas de escrita de código. Além disso, o tempo médio para responder aos problemas de Parson foram significativamente menores do que escrever o código equivalente.

Segundo relatos dos alunos da pesquisa, o esforço mental realizado para responder aos problemas de Parson com solução comum foi menor do que escrever o código equivalente (HAYNES-MAGYAR; ERICSON, 2022). A pesquisa também mostrou que os alunos tendem a utilizar as respostas dos problemas de Parson nos problemas de escrita de código equivalente e, portanto, o que implica que a melhor maneira de gerar problemas de Parson é agrupando as soluções comumente utilizadas pelos alunos.

O trabalho realizado por Ericson e Haynes-Magyar (2022) relata sobre o uso de problemas de Parson adaptativos como exercícios de aprendizagem ativa durante palestras. Conforme relatado por outros estudos, os problemas de Parson costumam levar menos tempo para serem resolvidos do que escrever um programa do zero, sendo assim uma ótima alternativa para ambientes de aprendizagem com tempo limitado, como por exemplo uma palestra. Este

estudo comparou o uso de problemas de Parsons adaptativos com escrita de código durante uma palestra com 152 alunos e relatou que o tempo médio para solucionar os problemas de Parson foi menor para 8 dos 10 exercícios passados, mas a diferença foi significativa em apenas cinco. Cerca de 78% dos alunos consideraram os problemas de Parson úteis para o aprendizado, mas cerca de 36% dos alunos preferem escrever os códigos sozinhos. Cerca de 80% dos alunos consideraram útil a adaptação intra-problemas, na qual o exercício vai sendo solucionado automaticamente caso o aluno esteja lutando para fazer sozinho. O estudo não avaliou os ganhos de aprendizagem e sugere que novas pesquisas sejam feitas para investigar o motivo de dois problemas de Parson levarem mais tempo para serem resolvidos do que escrever o código equivalente.

De maneira geral, as pesquisas mostram que os problemas de Parson são mais rápidos de serem respondidos do que escrever um código do zero, mas os dois tipos de exercício não possuem diferenças de ganhos de aprendizagem significativos. Os problemas de Parson se mostram eficazes em alunos que possuem baixa carga cognitiva em programação, alunos iniciantes, que preferem organizar a solução entregue, tendo acesso a dicas e outros auxílios. Esse exercício de conclusão auxilia os alunos ao apresentar diferentes exemplos de desenvolvimento de programas e, conforme o progresso, os alunos tendem a preferir exercícios mais desafiadores, no caso escrever o próprio programa. No próximo capítulo serão apresentadas as considerações finais e discutidos alguns pontos de pesquisa.

5 Considerações finais

Neste trabalho foi realizado um mapeamento sistemático da literatura de problemas de Parson como ferramenta de ensino de programação, mais especificamente do ensino de introdução à programação. Após a análise dos trabalhos obtidos na condução da revisão, foi possível responder às questões de pesquisa estabelecidas, identificando os vários tipos de problemas de Parson que são empregadas até o momento e verificando a influência que este tipo de exercício proporciona.

Os problemas de Parson são uma excelente alternativa de exercício para os alunos novatos em programação. Quando os alunos passam por muita dificuldade, a adaptação intra-problema altera o exercício ao mostrar o caminho até a solução. Conforme o aluno vai aprendendo, a adaptação inter-problema aumenta a dificuldade dos exercícios para mostrar exemplos incorretos e problemas mais complexos.

Por conta de suas características, os problemas de Parson se tornaram a preferência dos alunos novatos. Mesmo sendo mais simples, um problema de Parson não substitui a longo prazo um exercício de escrita de código, mas auxilia na absorção dos conteúdos de programação para que sejam utilizados durante o desenvolvimento de uma solução própria.

Apesar da crescente quantidade de trabalhos relacionados a problemas de Parson ao longo dos anos, poucos trabalhos conseguiram mostrar a eficácia da aprendizagem deste exercício. Como sugestão para trabalhos futuros, é interessante comparar os diferentes tipos de problemas de Parson com outros tipos de exercícios, além de escrita de código, e entender em quais os momentos mais indicados para serem utilizados em um curso de introdução à programação.

Referências

BENDER, J. et al. Learning computational thinking efficiently: How parsons programming puzzles within scratch might help. In: *Proceedings of the 24th Australasian Computing Education Conference*. New York, NY, USA: Association for Computing Machinery, 2022. (ACE '22), p. 66–75. ISBN 9781450396431. Disponível em: <https://doi.org/10.1145/3511861.3511869>. Citado na página 24.

BENNEDSEN, J.; CASPERSEN, M. Failure rates in introductory programming. *SIGCSE Bulletin*, v. 39, p. 32–36, 06 2007. Citado na página 14.

BENNEDSEN, J.; CASPERSEN, M. Failure rates in introductory programming: 12 years later. *ACM Inroads*, v. 10, p. 30–36, 04 2019. Citado na página 14.

DENNY, P.; LUXTON-REILLY, A.; SIMON, B. Evaluating a new exam question: Parsons problems. In: *Proceedings of the Fourth International Workshop on Computing Education Research*. New York, NY, USA: Association for Computing Machinery, 2008. (ICER '08), p. 113–124. ISBN 9781605582160. Disponível em: <https://doi.org/10.1145/1404520.1404532>. Citado na página 24.

DICHEVA, D.; HODGE, A. Active learning through game play in a data structures course. In: *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*. New York, NY, USA: Association for Computing Machinery, 2018. (SIGCSE '18), p. 834–839. ISBN 9781450351034. Disponível em: <https://doi.org/10.1145/3159450.3159605>. Citado na página 24.

DU, Y.; LUXTON-REILLY, A.; DENNY, P. A review of research on parsons problems. In: *Proceedings of the Twenty-Second Australasian Computing Education Conference*. New York, NY, USA: Association for Computing Machinery, 2020. (ACE'20), p. 195–202. ISBN 9781450376860. Disponível em: <https://doi.org/10.1145/3373165.3373187>. Citado 2 vezes nas páginas 20 e 24.

ERICSON, B.; HAYNES-MAGYAR, C. Adaptive parsons problems as active learning activities during lecture. In: *Proceedings of the 27th ACM Conference on on Innovation and Technology in Computer Science Education Vol. 1*. New York, NY, USA: Association for Computing Machinery, 2022. (ITiCSE '22), p. 290–296. ISBN 9781450392013. Disponível em: <https://doi.org/10.1145/3502718.3524808>. Citado 2 vezes nas páginas 24 e 36.

ERICSON, B.; MCCALL, A.; CUNNINGHAM, K. Investigating the affect and effect of adaptive parsons problems. In: *Proceedings of the 19th Koli Calling International Conference on Computing Education Research*. New York, NY, USA: Association for Computing Machinery, 2019. (Koli Calling '19). ISBN 9781450377157. Disponível em: <https://doi.org/10.1145/3364510.3364524>. Citado 4 vezes nas páginas 7, 24, 26 e 35.

ERICSON, B. J. et al. Parsons problems and beyond: Systematic literature review and empirical study designs. In: *Proceedings of the 2022 Working Group Reports on Innovation and Technology in Computer Science Education*. New York, NY, USA: Association for Computing Machinery, 2022. (ITiCSE-WGR '22), p. 191–234. ISBN 9798400700101. Disponível em: <https://doi.org/10.1145/3571785.3574127>. Citado 3 vezes nas páginas 7, 24 e 27.

ERICSON, B. J.; FOLEY, J. D.; RICK, J. Evaluating the efficiency and effectiveness of adaptive parsons problems. In: *Proceedings of the 2018 ACM Conference on International Computing Education Research*. New York, NY, USA: Association for Computing Machinery, 2018. (ICER '18), p. 60–68. ISBN 9781450356282. Disponível em: <https://doi.org/10.1145/3230977.3231000>. Citado 3 vezes nas páginas 24, 29 e 35.

ERICSON, B. J.; GUZDIAL, M. J.; MORRISON, B. B. Analysis of interactive features designed to enhance learning in an ebook. In: *Proceedings of the Eleventh Annual International Conference on International Computing Education Research*. New York, NY, USA: Association for Computing Machinery, 2015. (ICER '15), p. 169–178. ISBN 9781450336307. Disponível em: <https://doi.org/10.1145/2787622.2787731>. Citado 2 vezes nas páginas 24 e 32.

ERICSON, B. J.; MAEDA, H.; DHILLON, P. S. Detecting struggling students from interactive ebook data: A case study using csawesome. In: *Proceedings of the 53rd ACM Technical Symposium on Computer Science Education - Volume 1*. New York, NY, USA: Association for Computing Machinery, 2022. (SIGCSE 2022), p. 418–424. ISBN 9781450390705. Disponível em: <https://doi.org/10.1145/3478431.3499354>. Citado 2 vezes nas páginas 24 e 33.

ERICSON, B. J.; MARGULIEUX, L. E.; RICK, J. Solving parsons problems versus fixing and writing code. In: *Proceedings of the 17th Koli Calling International Conference on Computing Education Research*. New York, NY, USA: Association for Computing Machinery, 2017. (Koli Calling '17), p. 20–29. ISBN 9781450353014. Disponível em: <https://doi.org/10.1145/3141880.3141895>. Citado 3 vezes nas páginas 14, 24 e 33.

ERICSON, B. J.; MILLER, B. N. Runestone: A platform for free, on-line, and interactive ebooks. In: *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*. New York, NY, USA: Association for Computing Machinery, 2020. (SIGCSE '20), p. 1012–1018. ISBN 9781450367936. Disponível em: <https://doi.org/10.1145/3328778.3366950>. Citado na página 24.

FROMONT, F.; JAYAMANNE, H.; DENNY, P. Exploring the difficulty of faded parsons problems for programming education. In: *Proceedings of the 25th Australasian Computing Education Conference*. New York, NY, USA: Association for Computing Machinery, 2023. (ACE '23), p. 113–122. ISBN 9781450399418.

Disponível em: <https://doi.org/10.1145/3576123.3576136>. Citado 4 vezes nas páginas 8, 24, 30 e 35.

GOLETTI, O.; MENS, K.; HERMANS, F. Tutors' experiences in using explicit strategies in a problem-based learning introductory programming course. In: *Proceedings of the 26th ACM Conference on Innovation and Technology in Computer Science Education V. 1*. New York, NY, USA: Association for Computing Machinery, 2021. (ITiCSE '21), p. 157–163. ISBN 9781450382144. Disponível em: <https://doi.org/10.1145/3430665.3456348>. Citado na página 24.

HARMS, K. J.; CHEN, J.; KELLEHER, C. L. Distractors in parsons problems decrease learning efficiency for young novice programmers. In: *Proceedings of the 2016 ACM Conference on International Computing Education Research*. New York, NY, USA: Association for Computing Machinery, 2016. (ICER '16), p. 241–250. ISBN 9781450344494. Disponível em: <https://doi.org/10.1145/2960310.2960314>. Citado 3 vezes nas páginas 24, 27 e 34.

HAYATPUR, D. et al. Structuring collaboration in programming through personal-spaces. In: *Extended Abstracts of the 2023 CHI Conference on Human Factors in Computing Systems*. New York, NY, USA: Association for Computing Machinery, 2023. (CHI EA '23). ISBN 9781450394222. Disponível em: <https://doi.org/10.1145/3544549.3585630>. Citado na página 24.

HAYNES, C. C. Toward ability-based design for novice programmers with learning (dis)abilities. In: *Proceedings of the 2020 ACM Conference on International Computing Education Research*. New York, NY, USA: Association for Computing Machinery, 2020. (ICER '20), p. 336–337. ISBN 9781450370929. Disponível em: <https://doi.org/10.1145/3372782.3407104>. Citado na página 24.

HAYNES, C. C.; ERICSON, B. J. Problem-solving efficiency and cognitive load for adaptive parsons problems vs. writing the equivalent code. In: *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. New York, NY, USA: Association for Computing Machinery, 2021. (CHI '21). ISBN 9781450380966. Disponível em: <https://doi.org/10.1145/3411764.3445292>. Citado 2 vezes nas páginas 24 e 33.

HAYNES-MAGYAR, C.; ERICSON, B. The impact of solving adaptive parsons problems with common and uncommon solutions. In: *Proceedings of the 22nd Koli Calling International Conference on Computing Education Research*. New York, NY, USA: Association for Computing Machinery, 2022. (Koli Calling '22). ISBN 9781450396165. Disponível em: <https://doi.org/10.1145/3564721.3564736>. Citado 2 vezes nas páginas 24 e 36.

HELMINEN, J. et al. 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*. New York, NY, USA: Association for Computing Machinery,

2012. (ICER '12), p. 119–126. ISBN 9781450316040. Disponível em: <https://doi.org/10.1145/2361276.2361300>. Citado na página 24.

HELMINEN, J. et al. How do students solve parsons programming problems? – execution-based vs. line-based feedback. In: *2013 Learning and Teaching in Computing and Engineering*. [S.l.: s.n.], 2013. p. 55–61. Citado na página 24.

IHANTOLA, P.; HELMINEN, J.; KARAVIRTA, V. How to study programming on mobile touch devices: Interactive python code exercises. In: *Proceedings of the 13th Koli Calling International Conference on Computing Education Research*. New York, NY, USA: Association for Computing Machinery, 2013. (Koli Calling '13), p. 51–58. ISBN 9781450324823. Disponível em: <https://doi.org/10.1145/2526968.2526974>. Citado na página 24.

IHANTOLA, P.; KARAVIRTA, V. Open source widget for parson's puzzles. In: *Proceedings of the Fifteenth Annual Conference on Innovation and Technology in Computer Science Education*. New York, NY, USA: Association for Computing Machinery, 2010. (ITiCSE '10), p. 302. ISBN 9781605588209. Disponível em: <https://doi.org/10.1145/1822090.1822178>. Citado 2 vezes nas páginas 24 e 32.

KARAVIRTA, V. et al. Interactive learning content for introductory computer science course using the ville exercise framework. In: *2015 International Conference on Learning and Teaching in Computing and Engineering*. [S.l.: s.n.], 2015. p. 9–16. Citado na página 24.

KARAVIRTA, V.; HELMINEN, J.; IHANTOLA, P. A mobile learning application for parsons problems with automatic feedback. In: *Proceedings of the 12th Koli Calling International Conference on Computing Education Research*. New York, NY, USA: Association for Computing Machinery, 2012. (Koli Calling '12), p. 11–18. ISBN 9781450317955. Disponível em: <https://doi.org/10.1145/2401796.2401798>. Citado 2 vezes nas páginas 24 e 32.

KUMAR, A. N. Epplets: A tool for solving parsons puzzles. In: *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*. New York, NY, USA: Association for Computing Machinery, 2018. (SIGCSE '18), p. 527–532. ISBN 9781450351034. Disponível em: <https://doi.org/10.1145/3159450.3159576>. Citado na página 24.

KUMAR, A. N. Helping students solve parsons puzzles better. In: *Proceedings of the 2019 ACM Conference on Innovation and Technology in Computer Science Education*. New York, NY, USA: Association for Computing Machinery, 2019. (ITiCSE '19), p. 65–70. ISBN 9781450368957. Disponível em: <https://doi.org/10.1145/3304221.3319735>. Citado na página 24.

KUMAR, A. N. Mnemonic variable names in parsons puzzles. In: *Proceedings of the ACM Conference on Global Computing Education*. New York, NY, USA: Association for Computing Machinery, 2019. (CompEd '19), p. 120–126. ISBN 9781450362597. Disponível em: <https://doi.org/10.1145/3300115.3309509>. Citado na página 24.

LOJO, N.; FOX, A. Teaching test-writing as a variably-scaffolded programming pattern. In: *Proceedings of the 27th ACM Conference on on Innovation and Technology in Computer Science Education Vol. 1*. New York, NY, USA: Association for Computing Machinery, 2022. (ITiCSE '22), p. 498–504. ISBN 9781450392013. Disponível em: <https://doi.org/10.1145/3502718.3524789>. Citado na página 24.

LOPEZ, M. et al. Relationships between reading, tracing and writing skills in introductory programming. In: *Proceedings of the Fourth International Workshop on Computing Education Research*. New York, NY, USA: Association for Computing Machinery, 2008. (ICER '08), p. 101–112. ISBN 9781605582160. Disponível em: <https://doi.org/10.1145/1404520.1404531>. Citado na página 24.

MERRIENBOER, J. J. G. V. Strategies for programming instruction in high school: Program completion vs. program generation. *Journal of Educational Computing Research*, v. 6, p. 265–285, 01 1990. Citado na página 15.

MORRISON, B. B. et al. Subgoals help students solve parsons problems. In: *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*. New York, NY, USA: Association for Computing Machinery, 2016. (SIGCSE '16), p. 42–47. ISBN 9781450336857. Disponível em: <https://doi.org/10.1145/2839509.2844617>. Citado na página 24.

MULDNER, K.; JENNINGS, J.; CHIARELLI, V. A review of worked examples in programming activities. *ACM Trans. Comput. Educ.*, Association for Computing Machinery, New York, NY, USA, v. 23, n. 1, dec 2022. Disponível em: <https://doi.org/10.1145/3560266>. Citado na página 24.

OYELERE, S. S.; SUHONEN, J.; LAINE, T. H. Integrating parson's programming puzzles into a game-based mobile learning application. In: *Proceedings of the 17th Koli Calling International Conference on Computing Education Research*. New York, NY, USA: Association for Computing Machinery, 2017. (Koli Calling '17), p. 158–162. ISBN 9781450353014. Disponível em: <https://doi.org/10.1145/3141880.3141882>. Citado na página 24.

PARSONS, D.; HADEN, P. Parson's programming puzzles: A fun and effective learning tool for first programming courses. In: *Proceedings of the 8th Australasian Conference on Computing Education - Volume 52*. AUS: Australian Computer Society, Inc., 2006. (ACE '06), p. 157–163. ISBN 1920682341. Citado 5 vezes nas páginas 12, 15, 16, 24 e 32.

PETERSEN, K. et al. Systematic mapping studies in software engineering. *Proceedings of the 12th International Conference on Evaluation and Assessment in Software Engineering*, v. 17, 06 2008. Citado na página 19.

RUNESTONE. *Parsons Problems - Mixed Up Blocks*. 2018. Disponível em: <https://runestone.academy/ns/books/published/overview/Assessments/parsons.html>. Acesso em: 10 jun. 2023. Citado 7 vezes nas páginas 7, 17, 26, 28, 30, 31 e 33.

SIRKIä, T. Combining parson's problems with program visualization in cs1 context. In: *Proceedings of the 16th Koli Calling International Conference on Computing Education Research*. New York, NY, USA: Association for Computing Machinery, 2016. (Koli Calling '16), p. 155–159. ISBN 9781450347709. Disponível em: <https://doi.org/10.1145/2999541.2999558>. Citado 2 vezes nas páginas 24 e 29.

SMITH, D. H.; ZILLES, C. Discovering, autogenerating, and evaluating distractors for python parsons problems in cs1. In: *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1*. New York, NY, USA: Association for Computing Machinery, 2023. (SIGCSE 2023), p. 924–930. ISBN 9781450394314. Disponível em: <https://doi.org/10.1145/3545945.3569801>. Citado 2 vezes nas páginas 24 e 34.

STEPHENSON, B.; MANGAT, G. Using a computer to score parsons problems answered on paper. In: *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education*. New York, NY, USA: Association for Computing Machinery, 2021. (SIGCSE '21), p. 1069–1075. ISBN 9781450380621. Disponível em: <https://doi.org/10.1145/3408877.3432467>. Citado na página 24.

SWELLER, J. Cognitive load during problem solving: Effects on learning. *Cognitive Science*, v. 12, n. 2, p. 257–285, 1988. ISSN 0364-0213. Disponível em: <https://www.sciencedirect.com/science/article/pii/0364021388900237>. Citado na página 15.

THOMAS, A. et al. Stochastic tree-based generation of program-tracing practice questions. In: *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*. New York, NY, USA: Association for Computing Machinery, 2019. (SIGCSE '19), p. 91–97. ISBN 9781450358903. Disponível em: <https://doi.org/10.1145/3287324.3287492>. Citado na página 24.

TINKERCAD. *Official Guide to Tinkercad Circuits*. 2021. Disponível em: <https://www.tinkercad.com/blog/official-guide-to-tinkercad-circuits>. Acesso em: 10 jun. 2023. Citado 2 vezes nas páginas 7 e 18.

WAITE, J. et al. An online platform for teaching upper secondary school computer science. In: *Proceedings of the 2021 Conference on United Kingdom & Ireland Computing Education Research*. New York, NY, USA: Association for Computing Machinery, 2021. (UKICER '21). ISBN 9781450385688. Disponível em: <https://doi.org/10.1145/3481282.3481287>. Citado na página 24.

WANG, W. et al. Crescendo: Engaging students to self-paced programming practices. In: *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*. New York, NY, USA: Association for Computing Machinery, 2020. (SIGCSE '20), p. 859–865. ISBN 9781450367936. Disponível em: <https://doi.org/10.1145/3328778.3366919>. Citado na página 24.

WEINMAN, N.; FOX, A.; HEARST, M. Exploring challenging variations of parsons problems. In: *Proceedings of the 51st ACM Technical Symposium on*

Computer Science Education. New York, NY, USA: Association for Computing Machinery, 2020. (SIGCSE '20), p. 1349. ISBN 9781450367936. Disponível em: <https://doi.org/10.1145/3328778.3372639>. Citado na página 24.

WEINMAN, N.; FOX, A.; HEARST, M. A. Improving instruction of programming patterns with faded parsons problems. In: *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. New York, NY, USA: Association for Computing Machinery, 2021. (CHI '21). ISBN 9781450380966. Disponível em: <https://doi.org/10.1145/3411764.3445228>. Citado 4 vezes nas páginas 7, 24, 29 e 30.

ZHI, R. et al. Evaluating the effectiveness of parsons problems for block-based programming. In: *Proceedings of the 2019 ACM Conference on International Computing Education Research*. New York, NY, USA: Association for Computing Machinery, 2019. (ICER '19), p. 51–59. ISBN 9781450361859. Disponível em: <https://doi.org/10.1145/3291279.3339419>. Citado 3 vezes nas páginas 7, 24 e 32.