

DOI: 10.47456/rjn84x61

Resolução Computacional de Problemas Matemáticos Auxiliada pela Inteligência Artificial

Computational Solving of Mathematical Problems Assisted by
Artificial Intelligence

Lúcio S. Fassarella

Resumo: Neste artigo apresentamos os resultados de uma pesquisa básica, qualitativa e exploratória, desenvolvida com o objetivo de avaliar o uso da inteligência artificial para programar algoritmos-soluções elaborados em resoluções computacionais de problemas matemáticos. Fundamentamos teoricamente o trabalho nos conceitos de resolução computacional de problemas e pensamento computacional, com este sendo usado como recurso heurístico. Os procedimentos metodológicos consistem em três etapas: (1) escolha de problemas matemáticos adequados para resolução computacional; (2) elaboração de algoritmos-soluções para os problemas escolhidos; (3) uso de uma inteligência artificial para programar os algoritmos-soluções elaborados e validação dos resultados. Os programas foram escritos na linguagem Python pela inteligência artificial Gemini, versão 2.5 Flash. Para ilustrar a proposta, apresentamos neste artigo a resolução computacional de um problema matemático e a programação do algoritmo-solução pela inteligência artificial. Ao final, concluímos que o uso da inteligência artificial para programar algoritmos-soluções de problemas matemáticos é uma abordagem didática promissora e simples de implementar.

Palavras chave: Pensamento Computacional; Algoritmo-solução; Python; Gemini.

Abstract: This article presents the results of a basic, qualitative, and exploratory research project developed to evaluate the use of artificial intelligence to program algorithmic solutions for computational solving of mathematical problems. The theoretical framework for this work is based on the concepts of computational problem-solving and computational thinking, with the latter being used as a heuristic resource. The methodological procedures consist of three steps: (1) selection of suitable mathematical problems for computational solving; (2) development of algorithmic solutions for the chosen problems; (3) use of artificial intelligence to program the developed algorithmic solutions and validation of the results. The programs were written in the Python language using the Gemini artificial intelligence, version 2.5 Flash. To illustrate the proposal, this article presents the computational solution of a mathematical problem and the programming of the algorithmic solution by artificial intelligence. In conclusion, we found that using artificial intelligence to program algorithmic solutions for mathematical problems is a promising and simple-to-implement teaching approach.

Keywords: Computational Thinking; Algorithm-solution; Python; Gemini.

Introdução

O computador é até mais revolucionário como ideia, do que como um dispositivo prático que modifica a sociedade – e todos nós

sabemos o quanto ele mudou nossas vidas. Por que digo isso? Bem, o computador mudou a epistemologia, mudou o significado do “entender”. Para mim, você entende algo se você puder programá-lo. (Gregory Chaitin)

O uso dos computadores e da programação no ensino da Matemática tem sido objeto de estudo acadêmico desde antes do surgimento dos computadores pessoais, na década de 1970. Um dos pioneiros nesse campo foi o Prof. Seymour Papert, que participou do desenvolvimento do LOGO, uma linguagem de programação para uso educacional, e posteriormente concebeu o Construcionismo, uma teoria da aprendizagem baseada na ideia de que *as pessoas aprendem melhor por meio da construção de objetos significativos para elas*. Nessa concepção, o computador é um “objeto-para-pensar-com” que tem o potencial de revolucionar não só o modo como pensamos, mas também como aprendemos e resolvemos problemas (Papert, 1980; 1985; 2008; Valente, 1998). Entre outras coisas, o computador permite aos estudantes (aprendizes) manipular conceitos de um modo análogo à interação com objetos reais, favorecendo diretamente seu desenvolvimento mental:

O uso do computador requer certas ações que são bastante efetivas no processo de construção do conhecimento. Quando o aprendiz está interagindo com o computador ele está manipulando conceitos e isso contribui para o seu desenvolvimento mental. Ele está adquirindo conceitos da mesma maneira que adquire conceitos quando interage com objetos do mundo, como observou Piaget (Valente, 2002, p.40).

Pensando no ensino e aprendizagem da Matemática sob tais perspectivas, focamos nossa atenção na resolução computacional de problemas.

Resolver computacionalmente um problema matemático significa elaborar um algoritmo-solução: um algoritmo (i.e., sequência de instruções) cuja execução produz a solução do problema ou uma aproximação satisfatória da solução (Fassarella, 2015; 2022).

Os algoritmos podem ser concebidos e expressos de diversas formas: por meio da língua natural (conforme usada nas comunicações hodiernas), como fluxogramas ou como

*pseudocódigos*¹, entre outras possibilidades menos comuns. Enfim, podemos transferir a execução de um algoritmo escrito na língua natural ou na forma de pseudocódigo para um computador mediante a programação (codificação) do algoritmo em uma *linguagem de programação*.²

Na resolução computacional de problemas matemáticos, o computador assume um duplo papel: recurso para resolução de problemas e ambiente para representação e manipulação de objetos matemáticos mediante a elaboração e programação de algoritmos (Valente, 2002; Fassarella, 2022). Entretanto, apesar das contribuições potenciais da programação para o aprendizado da Matemática³, o uso didático da programação eventualmente requer dos estudantes o domínio de alguma linguagem de programação, o que pode dificultar sua aplicação em sala de aula tanto por somar mais um tópico ao currículo, quanto por ser um tema pouco atrativo e que requer metodologias de ensino específicas (Arimoto; Oliveira, 2019; Souza; Falcão; Mello, 2021). Para contornar essas dificuldades, propomos delegar a programação de algoritmos a uma inteligência artificial treinada especificamente para isso. Didaticamente, essa proposta é promissora porque elimina o trabalho de programar preservando o papel ativo dos estudantes na resolução de problemas. Ademais, ela não conflita com uma eventual intenção de ensinar programação para os estudantes – antes, serve como motivação por evidenciar seu valor prático.

1 Pseudocódigos são “(...) descrições sumárias de algoritmos na língua natural, eventualmente acrescida de alguns símbolos e regras [típicos da programação]” (Fassarella, 2022, p.3). Podemos considerar um pseudocódigo de um algoritmo como uma representação intermediária entre a expressão do algoritmo na língua natural e um programa de computador escrito em uma linguagem de programação específica. As diferenças cruciais entre um pseudocódigo e o correspondente programa de computador estão nas regras de sintaxe e no conjunto de funções pré-programadas da linguagem de programação.

2 Uma linguagem de programação é uma linguagem artificial desenvolvida especificamente para ser interpretada por computadores. Existem inúmeras linguagens de programação, cada uma com suas vantagens e desvantagens relativas.

3 Além da resolução computacional de problemas favorecer o desenvolvimento das habilidades relacionadas ao pensamento matemático (Valente, 2005), os algoritmos elaborados para resolver problemas matemáticos constituem registros de representação semiótica cuja mobilização contribui diretamente para a aprendizagem da Matemática devido a fatores cognitivos que podem ser analisados por meio da Teoria dos Registros de Representação Semiótica (Fassarella, 2022).

Partindo dessa proposta, desenvolvemos uma pesquisa básica, qualitativa e exploratória, com o objetivo de avaliar a viabilidade de usar uma inteligência artificial para programar algoritmos-soluções. Metodologicamente, a pesquisa consistiu nas seguintes etapas:

1. escolha de problemas matemáticos adequados para resolução computacional;
2. elaboração de algoritmos-soluções para os problemas escolhidos;
3. uso de uma inteligência artificial para programar os algoritmos-soluções elaborados e validação dos resultados.

Cabe enfatizar que nossa proposta para aplicação da inteligência artificial no ensino da Matemática visa potencializar a aprendizagem sem comprometer o desenvolvimento de habilidades cognitivas superiores dos estudantes. Isso porque a proposta não terceiriza a resolução de problemas para uma inteligência artificial, como se faz quando a ela é empregada para produzir soluções para os problemas diretamente a partir dos enunciados – uma abordagem que pode até ser bem-sucedida, gerando resoluções discursivas detalhadas e corretas (Cazadine; Cardoso, 2025), mas que também pode alijar o raciocínio dos estudantes e comprometer sua aprendizagem. Esse aspecto é significativo porque pesquisas recentes mostram que o uso de inteligências artificiais podem melhorar a performance dos estudantes em tarefas didáticas sem, todavia, promover “[...] o processamento cognitivo profundo e metacognitivo necessário para uma aprendizagem de alta qualidade” (Yan *et al.*, 2025, p.435, tradução nossa). O aparente paradoxo tem uma explicação simples: à medida que as inteligências artificiais substituem o trabalho cognitivo dos estudantes nas tarefas didáticas, elas suprem suas deficiências de conhecimento e competências intelectuais (melhorando o desempenho em algumas avaliações) ao mesmo tempo em que reduzem os estímulos para o desenvolvimento de habilidades cognitivas e uso de faculdades intelectuais superiores, como memorização, criatividade e criticidade (comprometendo o aprendizado). Esse entendimento é corroborado pela pesquisa

quantitativa realizada por Kosmyna *et al.* (2025) sobre o uso de *modelos de linguagem de grande escala* [*large language models* – *LLMs*] como assistentes para a escrita de ensaios:⁴

Embora os modelos de linguagem de grande porte (LLMs) ofereçam conveniência imediata, nossas descobertas destacam potenciais custos cognitivos. Ao longo de quatro meses, os usuários de LLMs apresentaram desempenho consistentemente inferior nos níveis neural, linguístico e comportamental. Esses resultados levantam preocupações sobre as implicações educacionais a longo prazo da dependência de LLMs [modelos de linguagem de grande porte] e ressaltam a necessidade de uma investigação mais aprofundada sobre o papel da IA na aprendizagem (Kosmyna *et al.*, 2025, excerto do resumo, tradução nossa).

O artigo tem a seguinte estrutura. Na primeira seção, discutiremos brevemente sobre a resolução computacional de problemas matemáticos, introduzindo alguns conceitos e terminologia, particularmente o conceito de pensamento computacional como recurso heurístico. Na segunda seção, descrevemos o percurso metodológico, explicitando fontes e critérios de seleção de problemas para a pesquisa, bem como nossas escolhas para a linguagem de programação e inteligência artificial empregados. A terceira seção traz a resolução computacional de um problema matemático para ilustrar a aplicação do pensamento computacional e o uso da inteligência artificial na programação de algoritmos-soluções. Na quarta seção, apresentamos as conclusões da pesquisa, levando em conta todos os problemas investigados. Nas Considerações Finais, resumimos o trabalho, acrescentamos alguns comentários e apontamos questões para pesquisas futuras.

Resolução computacional de problemas matemáticos

4 Modelos de linguagem de grande escala são um tipo específico de inteligência artificial baseada em redes neurais profundas e cuja base de dados de treinamento é constituída principalmente por textos escritos por pessoas. O objetivo desses modelos é produzir textos coerentes em linguagem humana a partir de instruções dadas também em linguagem humana. Atualmente, há inúmeros modelos desse tipo disponíveis para acesso pela Internet, entre eles o ChatGPT (<https://chatgpt.com/>), o Claude (<https://claude.com/>), o Grok (<https://grok.com/>) e o Gemini (<https://gemini.google.com>).

Para orientar a resolução computacional de um problema matemático no contexto didático, propomos o seguinte protocolo estruturado em quatro etapas:⁵

1. Interpretação do problema.
2. Estabelecimento de um plano de resolução. Consiste no processo de formulação ou descoberta de um caminho para resolver o problema a partir das informações disponíveis. Característica essencial das resoluções computacionais é que o plano de resolução culmine na elaboração de um algoritmo-solução, possivelmente escrito na língua natural ou como pseudocódigo.
3. Programação do algoritmo-solução. Caso seja necessário transferir a execução do algoritmo-solução para um computador, essa etapa consiste em escrever (ou codificar) o algoritmo-solução em uma linguagem de programação. Aqui, propomos que essa tarefa possa ser delegada a uma inteligência artificial capaz de receber o algoritmo-solução escrito na língua natural ou como pseudocódigo e retornar a codificação desse algoritmo na linguagem de programação escolhida pelo estudante.
4. Execução do algoritmo-solução e validação do resultado. A execução do algoritmo-solução (pelo estudante ou por um computador) gera um resultado que deve corresponder à solução do problema inicial (ou a uma aproximação satisfatória dela). Esse resultado precisa ser verificado devido à possibilidade de ocorrência de erros nas etapas anteriores. Essa tarefa pode ser realizada de três formas, pelo menos:
 - I. Verificação direta: avaliação da correção/incorreção da resposta pela inspeção de um conjunto finito de critérios, como satisfazer uma equação algébrica. Os critérios podem ser exatos ou aproximados, a depender do problema e da estratégia de resolução empregada.

5 O leitor pode notar que este protocolo constitui um amálgama das concepções de Polya (1978) e Valente (2002), cujas semelhanças estão destacadas em Moraes (2020).

- II. Verificação indireta: avaliação da correção/ incorreção da resposta por meio de uma análise rigorosa do algoritmo, visando garantir que sua execução precisa gera a solução do problema. Se realizada com perfeito rigor matemático, eventualmente equivale a uma demonstração de que o resultado do algoritmo é certo ou, alternativamente, errado.
- III. Testes: consiste no emprego do algoritmo elaborado (com eventuais adaptações) para resolver versões reduzidas, especializadas ou simplificadas do problema, para as quais conhecemos as soluções por vias alternativas. Nesse caso, presumimos que o fato de o algoritmo produzir respostas certas para as variações testadas do problema significa que também deve produzir a resposta certa para o problema original. Embora essa estratégia de validação não garanta a correção do resultado do algoritmo, ela pode ser a única alternativa viável quando não for factível verificar direta ou indiretamente o resultado do algoritmo-solução, seja pela falta de critérios acessíveis que caracterizem a solução, seja pela complexidade do próprio algoritmo. Enfim, os testes podem gerar confiança subjetiva nos resultados, a depender da sua quantidade e qualidade.⁶

Na resolução computacional de problemas matemáticos, podemos proficuamente utilizar como recurso heurístico o pensamento computacional, entendido como a “[...] capacidade de compreender, definir, modelar, comparar, solucionar, automatizar e analisar problemas (e soluções) de forma metódica e sistemática, através da construção de algoritmos” (Vicari et al., 2018, p.15).⁷ O

⁶ A verificação por testes é exemplificada na resolução do problema que apresentamos adiante. Fassarella (2009) discute um exemplo mais sofisticado e detalhado desse tipo de validação no contexto da resolução algébrica de um problema de probabilidade no nível do Ensino Médio.

⁷ Esse entendimento exprime a concepção da Sociedade Brasileira de Computação e foi incorporado à Lei N° 14.533, de 11 de janeiro de 2023, que institui a Política Nacional de Educação Digital e especifica que o pensamento computacional “[...] se refere à capacidade de compreender, analisar, definir, modelar, resolver, comparar e automatizar problemas e suas

termo foi utilizado originalmente por Papert (1980) e defendido por Jeannette Wing (2016) com o significado amplo de “(...) uma coleção de diversas habilidades humanas para resolver problemas obtidas do estudo da natureza da computação” (Curzon; McOwan, 2017, p.3, tradução nossa). Seguindo Brackmann (2017), consideramos o pensamento computacional como uma forma de abordar problemas matemáticos estruturada em quatro pilares:

- *Decomposição*: dividir problemas complexos em problemas mais simples;
- *Abstração*: identificar nos problemas as informações que são relevantes para resolução, descartando aspectos irrelevantes;
- *Reconhecimento de padrões*: identificar nos problemas aspectos estruturais que possam ser descritos como repetições ou recorrências, bem como descobrir semelhanças entre problemas diferentes que permitam aplicar estratégias conhecidas em situações novas;
- *Elaboração de algoritmos*: definir sequências detalhadas de ações para resolver problemas.

Geralmente, os pilares do pensamento computacional são mobilizados sinergicamente na resolução de um problema. No contexto do ensino da Matemática, eles podem ser propostos como diretrizes para os alunos elaborarem seus planos de resolução (Tavares, 2024). Com a prática, a expectativa é de que os estudantes internalizem os pilares e passem a mobilizá-los automaticamente.

Mesmo que o plano de resolução estabelecido tenha a forma de um algoritmo-solução, não é necessário que ele seja programado em um computador no caso em que for possível ser executado manualmente – situação típica de atividades *desplugadas*, relacionadas ao desenvolvimento ou aplicação do pensamento

soluções de forma metódica e sistemática, por meio do desenvolvimento da capacidade de criar e adaptar algoritmos [...]” (Brasil, 2023). Cabe destacar que a literatura apresenta variações na conceituação do pensamento computacional, mas essa discussão não nos é pertinente aqui; aos leitores interessados em aprofundamentos, indicamos Wing (2016), Valente (2016), Brackmann (2017), Vicari et al. (2018), Curzon e McOwan (2017).

computacional; veja os exemplos dados por Crema (2020) e Tavares (2024).

Quando verificamos que o resultado de um algoritmo não é uma solução do problema proposto, precisamos realizar sua depuração (*debugging*): avaliar o processo de resolução para identificar e corrigir eventuais erros, os quais podem ter ocorrido em qualquer etapa da resolução. As contribuições didáticas da depuração para a aprendizagem de conceitos matemáticos são discutidas extensivamente por Valente (2002; 2005).⁸

Percurso Metodológico da Pesquisa

Alguns esclarecimentos sobre o percurso metodológico da pesquisa, especificamente sobre as escolhas dos problemas abordados, da linguagem de programação adotada e da inteligência artificial empregada para programar.

Os problemas investigados ao longo da pesquisa foram elaborados pelo autor ou obtidos de fontes variadas, como Devlin (1984), Kosova *et al.* (2024) e o *Project Euler* (2026).⁹ Para selecionar os problemas, usamos dois critérios de inclusão: (i) a possibilidade de resolução computacional e (ii) envolver apenas conceitos básicos de Matemática (até o nível do Ensino Médio). Investigamos uma parte da miríade de problemas possíveis e quinze deles foram apresentados em uma *oficina de extensão*¹⁰ realizada em 2025 para o público-alvo de estudantes de graduação, licenciandos e professores de Matemática da Educação Básica. Esses problemas estão disponíveis para acesso livre na Internet¹¹ e dentre eles escolhemos um para ilustrar a proposta da pesquisa.

8 Corroborando teses de Valente (2002; 2005), Moraes (2020) e Kosova *et al.* (2024) evidenciam as contribuições da programação para avaliação de conhecimentos matemáticos de estudantes.

9 "O Projeto Euler é uma série de problemas desafiadores de matemática/programação que exigirão mais do que apenas conhecimentos matemáticos para serem resolvidos. Embora a matemática ajude a encontrar métodos elegantes e eficientes, o uso de um computador e habilidades de programação serão necessários para resolver a maioria dos problemas" (Project Euler, 2026, tradução nossa).

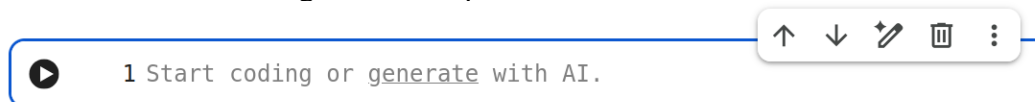
10 Disponível em: <https://fassarella.online/ensino/programath/2025/>. Acesso em: 4 ago. 2026.

11 Disponível em: <https://github.com/LucioFassarella/Programath/blob/main/Programath.ipynb>. Acesso em: 4 ago. 2026.

Os algoritmos-soluções dos problemas investigados foram programados na linguagem *Python*¹² pela inteligência artificial *Gemini*¹³, na plataforma de programação *Colab*¹⁴ do Google. O Python foi escolhido por ser uma linguagem de programação que se aproxima de uma língua natural, o Inglês; além disso, ele possui muitas bibliotecas e uma ampla comunidade de usuários, o que facilita sua aprendizagem e aplicação em diversas áreas. O Gemini foi a inteligência artificial empregada para programar os algoritmos-soluções porque ele possui treinamento para programação e está disponível para uso gratuito na plataforma Colab.¹⁵ Já o Colab é uma plataforma gratuita de programação na nuvem que podemos acessar por navegadores de Internet, dispensando a necessidade de instalação de quaisquer softwares específicos.

No Colab, o *prompt* de comando (Figura 1) nos permite escrever o código Python de um algoritmo ou abrir uma nova janela para instrução do *Gemini*, bastando clicarmos em “generate with AI” (gerar com IA):

Figura 1: *Prompt* de comando do Colab



Fonte: <https://colab.research.google.com>. Acesso em 19 jan. 2026.

No *prompt* de comando do Gemini (Figura 2), podemos instruir a inteligência artificial para programar um algoritmo em Python inserindo o algoritmo em língua natural, como pseudocódigo ou descrevendo o que o programa ele deve realizar.

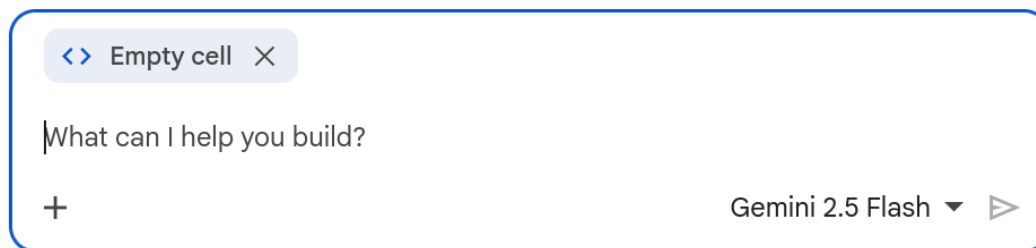
Figura 2: *Prompt* de comando do Gemini

¹² “O Python é uma linguagem de programação de alto nível, interpretada e de propósito geral, que se tornou uma das mais populares do mundo. [...] sua filosofia central foca na legibilidade do código e na produtividade do programador” (Google, 2026a).

¹³ “O Gemini é a inteligência artificial generativa e multimodal desenvolvida pelo Google. Em sua versão mais atual (família Gemini 3), ele se destaca por não ser apenas um chatbot, mas um assistente pessoal proativo capaz de processar e criar diferentes tipos de informação simultaneamente” (Google, 2026b).

¹⁴ “O Google Colab (abreviação de Colaboratory) é um serviço de nuvem gratuito, baseado no Jupyter Notebook, que permite escrever e executar código Python diretamente no seu navegador, sem a necessidade de qualquer configuração local” (Google, 2026c).

¹⁵ Atualmente, o Colab disponibiliza duas versões do Gemini: versão 2.5 *Flash* (gratuita) ou a versão 3 (exclusiva para assinantes). Na pesquisa, utilizamos apenas a versão 2.5 *Flash*.



Fonte: O autor.

Em geral, os programas gerados pelo Gemini contêm comentários que esclarecem os elementos do código, facilitando a compreensão da programação. Conforme podemos ver na Figura 4, os comentários são os trechos precedidos pelo símbolo “#”, exibidos em cor verde na tela e estão escritos em inglês.

O uso do Gemini no Colab é simples, prático e foi suficiente para programar os algoritmos-soluções dos problemas investigados. Essa ferramenta também pode ser usada para depurar algoritmos com erros de sintaxe e indicar correções, mas não usamos esse recurso na pesquisa.

Conforme antecipado, apresentamos ilustrativamente na próxima seção a resolução computacional de um problema matemático em que aplicamos a inteligência artificial Gemini para programar o algoritmo-solução elaborado.

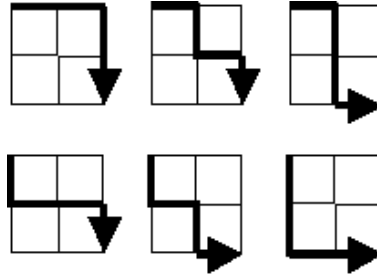
Resolução computacional com programação por inteligência artificial

Nesta seção, discutimos a resolução computacional de um problema matemático específico com uso da inteligência artificial para programar o algoritmo-solução. O leitor poderá perceber a mobilização de conhecimentos matemáticos e dos pilares do pensamento computacional, evidenciando que a proposta preserva o valor didático da atividade.

Problema: Numa grade de pontos 21x21, existem quantos caminhos ligando o canto superior esquerdo ao canto inferior direito, se forem permitidos apenas movimentos para a direita e para baixo?

Para exemplificar, os caminhos admissíveis numa grade 3x3 são dados na Figura 3:

Figura 3: grade 3x3 e seus seis caminhos admissíveis



Fonte: <https://projecteuler.net/problem=15>.
Acesso em: 19 jan. 2026.¹⁶

O problema pode ser resolvido por meio de uma recursão que exprima o número de caminhos permitidos em uma grade com dimensões $n \times m$ em termos dos números de caminhos permitidos em grades com dimensões menores, sendo n o inteiro positivo que corresponde ao número de pontos em uma (qualquer) linha vertical e m o inteiro positivo que corresponde ao número de pontos em uma (qualquer) linha horizontal. O pilar abstração do pensamento computacional se manifesta no reconhecimento de que o problema essencialmente não muda quando admitimos grades com dimensões n e m diferentes de 21; o pilar decomposição está na estratégia de reduzir o problema de dimensões 21×21 a problemas de dimensões menores; já o reconhecimento de padrões corresponde à percepção de que a mesma estratégia de resolução se aplica a problemas para grades com quaisquer dimensões.¹⁷

A partir dessas constatações, nosso plano de resolução consiste em definir uma função real recursiva $S(n, m)$, com variáveis inteiras positivas n e m , cujo valor seja o número de caminhos admissíveis numa grade $n \times m$; depois, usamos essa função para calcular a solução do problema original, ou seja $S(21, 21)$. Para definir a função $S(n, m)$, observamos que, nos casos de grades com dimensões $n = 1$ ou $m = 1$, é

¹⁶ Destacamos que o problema que enunciamos é ligeiramente diferente do problema original do Project Euler, mas as resoluções são análogas e as soluções, iguais. Fizemos essa modificação apenas para facilitar a descrição da resolução neste artigo.

¹⁷ Admitir grades com dimensões n e m diferentes é uma abstração do problema que facilita a resolução por tornar a relação de recorrência mais simples, o que eventualmente se percebe somente ao se tentar explicitar essa relação.

fácil ver que o número de caminhos admissíveis é exatamente 1. No caso em que as dimensões n e m são maiores ou iguais a 2, o primeiro segmento de um caminho admissível na grade deve ligar o vértice superior esquerdo ao vértice imediatamente à direita ou ao vértice imediatamente abaixo, implicando que o valor de $S(n, m)$ corresponde à soma dos valores $S(n - 1, m)$ e $S(n, m - 1)$ – ou seja, o problema de dimensões $n \times m$ se reduz aos problemas de dimensões $(n - 1) \times m$ e $n \times (m - 1)$. Em síntese, $S(n, m)$ é uma função a duas variáveis inteiras positivas satisfazendo as seguintes condições iniciais e relação de recorrência:

$$S(n, 1) = 1; S(1, m) = 1; S(n, m) = S(n - 1, m) + S(n, m - 1), \text{ se } n > 1 \text{ e } m > 1.$$

Podemos exprimir a definição de $S(n, m)$ e instruir o cálculo de $S(21, 21)$ na forma de pseudocódigo:

Algoritmo-solução

Função $S(n, m)$:

1. se $n = 1$ ou $m = 1$, retorne 1;
2. se $n > 1$ e $m > 1$, retorne $S(n-1, m) + S(n, m-1)$.

Calcule $S(21, 21)$.

Inserindo esse algoritmo-solução no Gemini, obtemos o programa Python mostrado na Figura 4:

Figura 4: o algoritmo-solução do problema em Python, programado pelo Gemini

```

1 memo = {}
2
3 def S(n, m):
4     # Check if the result is already in memoization table
5     if (n, m) in memo:
6         return memo[(n, m)]
7
8     # Base case: if n=1 or m=1, there's only one path
9     if n == 1 or m == 1:
10        result = 1
11    # Recursive case: sum of paths from (n-1, m) and (n, m-1)
12    else:
13        result = S(n - 1, m) + S(n, m - 1)
14
15    # Store the result in the memoization table before returning
16    memo[(n, m)] = result
17    return result
18
19 # Calculate S(21, 21)
20 result_21_21 = S(21, 21)
21 print(f"0 número de caminhos para uma grade 21x21 é: {result_21_21}")

```

... 0 número de caminhos para uma grade 21x21 é: 137846528820

Fonte:

<https://colab.research.google.com/drive/18qurOhbTnyhJBLdT1z3i69c4fw3W2XY1>.

Acesso em: 19 jan. 2026.

A execução da função $S(n, m)$ com as variáveis $n = 21$ e $m = 21$ gerou o resultado $S(21, 21) = 137846528820$. Não é possível validar esse resultado por verificação direta, mas podemos testar o algoritmo-solução para grades de pequenas dimensões, comparando os valores da função $S(n, m)$ com os valores que obtemos por enumeração explícita dos caminhos admissíveis. Fazendo isso para cinco grades, verificamos que os resultados do algoritmo estão corretos, o que finaliza a resolução do problema.¹⁸

$$S(1, 21) = 1; S(21, 1) = 1; S(2, 2) = 2; S(2, 3) = 3; S(3, 3) = 6.$$

Na próxima seção, resumiremos as conclusões de caráter geral sobre a resolução computacional de problemas com auxílio da inteligência artificial, obtidas com base nos problemas investigados na pesquisa.

¹⁸ Omitimos os desenhos das malhas e caminhos admissíveis, mas o leitor pode produzi-los facilmente e verificar a correção dos valores da função $S(n, m)$ para os valores de n e m indicados.

Conclusões

Primeiramente, observamos que os problemas passíveis de resolução computacional envolvem o cálculo de valores de funções reais, a avaliação de expressões numéricas ou sentenças lógicas em conjuntos finitos, ou podem ser resolvidos por uma estratégia que essencialmente se reduz a um processo recursivo, como no caso do problema discutido na seção anterior. Problemas geométricos podem ser tratados, como em Moraes (2020), mas em geral dependemos da aplicação de fórmulas e os algoritmos-soluções não fornecem vantagem performática em relação ao uso de calculadoras.¹⁹

Percebemos que as resoluções computacionais requerem um conhecimento matemático mínimo que vai além do necessário para interpretação do enunciado e elaboração do plano de resolução, pois a validade dos algoritmos-soluções dependem de os objetos e conceitos matemáticos serem corretamente empregados e consistentemente relacionados. Em geral, a resolução computacional mobiliza mais conhecimentos e habilidades do que a mera aplicação de fórmulas ou execução de procedimentos memorizados, embora possa eventualmente se utilizar disso.

Para muitos problemas, uma resolução computacional pode ser implementada sem depender de conhecimento teórico aprofundado ou habilidade operacional excepcional. Por um lado, isso pode não favorecer o aprofundamento teórico dos estudantes; por outro, amplia a gama de problemas que podem ser abordados em sala de aula, permitindo que os professores e estudantes possam tratar situações mais interessantes ou realistas, possivelmente demasiado tediosas para resolver manualmente. Vamos ilustrar essa situação com dois exemplos. No problema da seção anterior, tivemos apenas que definir recursivamente a função $S(n,m)$ para construir o algoritmo-solução que nos permitiu calcular $S(21,21)$ pelo computador (que efetivamente realizou a recorrência envolvida); não sendo praticável realizar manualmente a recorrência envolvida no cálculo de

19 Aqui não estamos considerando a possibilidade didática de se usar a atividade para provocar a reflexão dos estudantes ou avaliar seus conhecimentos, como preconizado por Valente (2002; 2005), Moraes (2020) e Kosova et al. (2024).

$S(21,21)$, para obter esse valor sem uso do computador precisaríamos resolver o problema por uma via alternativa ou encontrar uma fórmula fechada para a função $S(n, m)$ a partir da sua relação de recorrência, o que nos requer um certo nível de conhecimento das funções recursivas ou uma aplicação sagaz da *indução finita*.²⁰ Como segundo exemplo, considere a tarefa de encontrar o máximo divisor comum (mdc) de um par de números inteiros. Ela pode ser realizada computacionalmente por alguém que meramente conheça a definição desse conceito, ignorando completamente tópicos avançados como o *algoritmo de Euclides* ou a *fatoração prima* (Fassarella, 2021b).

Eventualmente, não é possível negligenciar a complexidade computacional de um algoritmo-solução, o que está relacionado ao tempo necessário para o algoritmo ser executado pelo computador. Não é incomum que problemas relativamente simples tenham algoritmos-soluções cuja execução requer um tempo de execução impraticável. Essa possibilidade mostra que as resoluções computacionais têm limitações práticas mesmo quando são teoricamente realizáveis, sendo um tópico que precisa ser considerado pelo professor na escolha dos problemas didáticos e que pode ser discutido em um contexto apropriado.

Retomando o objetivo da pesquisa, considerando que os algoritmos-soluções de todos os problemas investigados foram programados sem dificuldades pelo Gemini e que a execução desses programas produziram sempre respostas válidas para os problemas, concluímos que o uso da inteligência artificial para programar algoritmos-soluções para problemas matemáticos é uma abordagem didática promissora e simples de implementar.

Considerações Finais

Discutimos o uso de uma inteligência artificial para programar algoritmos-soluções na perspectiva de tornar mais acessível aplicar a resolução computacional de problemas matemáticos no contexto da

20 Para os leitores interessados: $S(n, m) = \frac{(n+m-2)!}{(n-1)! \times (m-1)!}$.

Educação Básica. Mais especificamente, apresentamos os fundamentos dessa proposta, delineamos os pilares do pensamento computacional como recurso heurístico para resolução computacional de problemas e exemplificamos a abordagem com a resolução computacional de um problema específico.

Entendemos que o emprego da inteligência artificial para programar algoritmos-soluções não se contrapõe ao ensino da programação na Educação Básica. Pelo contrário, esse pode ser um meio para motivar a aprendizagem da programação e também um recurso útil no ensinar a programar: a prática da resolução computacional de problemas desvinculada das tecnicidades das linguagens de programação mostra de forma direta a eficácia dessa abordagem ao mesmo tempo em que favorece o desenvolvimento do pensamento computacional e permite aos estudantes se familiarizarem contextualmente com a linguagem de programação. Em particular, professores ou estudantes podem usar a inteligência artificial nas atividades didáticas de programação para corrigir erros, para simplificar algoritmos produzidos ou mesmo para produzir programas alternativos.

Destacamos duas limitações da nossa pesquisa. Primeiro, usamos apenas o modelo de inteligência artificial Gemini. Outros modelos de inteligência artificial podem produzir resultados diferentes e mesmo o Gemini pode ter desempenhos variados frente a problemas mais complexos dos que tratamos. Pesquisas futuras podem investigar essas possibilidades e estabelecer comparações úteis.

Em segundo lugar, nossa pesquisa sobre o uso da inteligência artificial para programar algoritmos-soluções tem caráter teórico e qualitativo. Esperamos que as possíveis contribuições didáticas que discutimos, bem como outras, possam ser investigadas por pesquisas de campo junto a estudantes da Educação Básica.

Referências

ARIMOTO, Maurício; OLIVEIRA, Weldrey. Dificuldades no processo de aprendizagem de programação de computadores: um survey com

estudantes de cursos da área de computação. In: WORKSHOP SOBRE EDUCAÇÃO EM COMPUTAÇÃO (WEI), 27., 2019, Belém. **Anais [...]**. Porto Alegre: Sociedade Brasileira de Computação, 2019. p. 244-254. ISSN 2595-6175. DOI: <https://doi.org/10.5753/wei.2019.6633>. Acesso em: 14 jan. 2026.

BRACKMANN, Christian P. **Desenvolvimento do pensamento computacional através de atividades desplugadas na educação básica**. 2017. 226 p. Tese (Doutorado em Informática na Educação) – Universidade Federal do Rio Grande do Sul, Porto Alegre, 2017. Orientador: Dante Augusto Couto Barone. Disponível em: <https://lume.ufrgs.br/handle/10183/172208>. Acesso em: 12 dez. 2022.

BRASIL, **Lei Nº 14.533**, de 11 de janeiro de 2023. Art.3º, Inciso I. Disponível em: https://www.planalto.gov.br/ccivil_03/_ato2023-2026/2023/lei/114533.htm. Acesso em: 3 jan. 2026.

CAZADINE, Amanda F.; CARDOSO, Valdinei C. Explorando o uso do chatgpt em situações-problema de proporcionalidade. **Kiri-Keré: Pesquisa em Ensino**, v.1, n.24, p.67-87, 2025. DOI: <https://doi.org/10.47456/5h8p2s12>. Acesso em: 12 jan. 2026.

CREMA, Cristiani. **Computação desplugada para estudantes do Ensino Médio**: concepção, execução e avaliação de atividades. 2020. 155p. Dissertação (Mestrado em Ensino de Ciências, Matemática e Tecnologias) – Universidade do Estado de Santa Catarina, Joinville, 2020. Disponível em: https://sucupira-legado.capes.gov.br/sucupira/public/consultas/coleta/trabalhoConclusao/viewTrabalhoConclusao.jsf?popup=true&id_trabalho=9917752. Acesso em: 14 jan. 2026.

CURZON, Paul; MCOWAN, Peter W. **The Power of Computational Thinking**: Games, Magic and Puzzles to Help You Become a Computational Thinker. Singapore: World Scientific, 2017. DOI: <https://doi.org/10.1142/q0054>. Acesso em: 12 jan. 2026.

FASSARELLA, Lúcio S. **A resposta é plausível?** Comentários baseados em um cálculo de probabilidades. **Revista do Professor de Matemática**, n.70, p.1-5, 2009. Disponível em: <http://www.rpm.org.br/cdrpm/70/10.html>. Acesso em: 5 jan. 2026.

FASSARELLA, Lúcio S. Resolução Computacional de Problemas de Probabilidade. **Proceedings Series of the Brazilian Society of Computational and Applied Mathematics**, v.3, n.2, p.020129 [6 pages], 2015. DOI: <https://doi.org/10.5540/03.2015.003.02.0129>. Acesso em: 5 jan. 2026.

FASSARELLA, Lúcio S. Estimando probabilidades por simulações computacionais. **Professor de Matemática Online**, Goiânia, v. 9, n. 2, p. 240-251, 2021a. DOI: <https://doi.org/10.21711/2319023x2021/pmo917>. Acesso em: 12 jan. 2026.

FASSARELLA, Lúcio S. Desconexão procedimental e programação no ensino-aprendizagem da matemática: considerações a partir da teoria

dos registros de representação semiótica. **Revemat**, v.15, n.2, p.1-24, 2021b. DOI: <https://doi.org/10.5007/1981-1322.2020.e76610>. Acesso em: 19 jan. 2026.

FASSARELLA, Lúcio S. Aspectos semióticos da programação da aprendizagem da matemática. In: ENCONTRO REGIONAL DE MATEMÁTICA APLICADA E COMPUTACIONAL, 2022, São Mateus. **Anais** [...]. São Mateus: Universidade Federal do Espírito Santo, 2022. p. 62-68. ISBN 978-65-00-72846-0. Disponível em: <https://drive.google.com/file/d/1jqoFF2rNbLi-uSBY1Fmy335G6ATHL0Jg/view>. Acesso em: 6 jan. 2026.

GOOGLE. **Breve descrição da linguagem de programação Python**. Gemini, versão 3, 19 jan. 2026a. Disponível em: <https://gemini.google.com/share/d9fa314ec93>. Acesso em: 19 jan. 2026.

GOOGLE. **Breve descrição da inteligência artificial Gemini**. Gemini, versão 3, 19 jan. 2026b. Disponível em: <https://gemini.google.com/share/d55d80aec5f3>. Acesso em: 19 jan. 2026.

GOOGLE. **Breve descrição do Google Colab**. Gemini, versão 3, 19 de jan. 2026c. Disponível em: <https://gemini.google.com/share/8ab0d0f7cab8>. Acesso em: 19 jan. 2026.

KOSMYNA, Nataliya *et al.* Your brain on ChatGPT: accumulation of cognitive debt when using an AI assistant for essay writing task. **arXiv preprint**, arXiv:2506.08872, 2025. Disponível em: <https://arxiv.org/abs/2506.08872>. Acesso em: 14 jan. 2025.

KOSOVA, Robert; SINA, Zenel; KAPÇIU, Rinela; KOSOVA, Anna M.; KODRA, Asad. Solving math problems with computer programs: the benefits for the high school students. In: INTERNATIONAL CONFERENCE ON ENGINEERING, NATURAL AND SOCIAL SCIENCES, 3., 2024, Konya. **Proceedings** [...]. Konya: [s. n.], 2024. p. 703-710. Disponível em: https://drive.google.com/file/d/1847SEy_Tfx7ke1qnO6Pepq873WnlkFkV/view. Acesso em: 17 jan. 2026.

MORAES, Fábio A. C. **A Programação como ferramenta didático-pedagógica para o ensino de Geometria na Educação Básica em um ambiente construcionista**. 2020. 247f. Dissertação (Mestrado em Ensino na Educação Básica) – Programa de Pós-Graduação em Ensino na Educação Básica, Universidade Federal do Espírito Santo, Centro Universitário Norte do Espírito Santo, São Mateus-ES, 2020. Disponível em: <https://ensinonaeducacaobasica.ufes.br/pt-br/pos-graduacao/PPGEEB/detalhes-da-tese?id=14489>. Acesso em: 14 jan. 2026.

PAPERT, Seymour. **Mindstorms**: Children, Computers, And Powerful Ideas. New York: Basic Books, 1980.

PAPERT, Seymour. **Logo**: Computadores e Educação. Brasiliense, São Paulo, 1985 (1980).

PAPERT, Seymour. **A máquina das crianças**: repensando a escola na era da informática. Ed. revisada. Tradução: Sandra Costa. Porto Alegre: Artmed, 2008 (1993).

PROJECT EULER. **Project Euler**. Página inicial. Disponível em: <https://projecteuler.net/>. Acesso em: 17 jan. 2026.

POLYA, Georg. **A arte de resolver problemas**. Rio de Janeiro: Interciência, 1978.

SOUZA, Franciely Alves de; FALCÃO, Taciana Pontual; MELLO, Rafael Ferreira. O Ensino de Programação na Educação Básica: Uma Revisão da Literatura. In: SIMPÓSIO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO (SBIE), 32. , 2021, Online. *Anais [...]*. Porto Alegre: Sociedade Brasileira de Computação, 2021 . p. 1265-1275. DOI: <https://doi.org/10.5753/sbie.2021.218461>. Acesso em: 14 jan. 2026.

TAVARES, Lucas C. **O pensamento computacional na resolução de problemas discretos**: seu valor didático à luz da Teoria dos Registros de Representação Semiótica. Dissertação (Mestrado em Ensino na Educação Básica) – Universidade Federal do Espírito Santo, Centro Universitário Norte do Espírito Santo. 2024. 136f. São Mateus, ES, 2024. Disponível em: <http://repositorio.ufes.br/handle/10/17977>. Acesso em: 5 jan. 2026.

VALENTE, José A. Por quê o computador na educação? In: Valente, José A. (org.). **Computadores e conhecimento**: repensando a educação. 2a. ed. Campinas, SP: Unicamp/Nied, 1998. p.29-53.

VALENTE, José A. A Espiral da aprendizagem e as tecnologias da informação e comunicação: Repensando conceitos. Em M.C. Joly (ed.). **Tecnologia no Ensino**: implicações para a aprendizagem. São Paulo: Casa do Psicólogo Editora, 2002. pp. 15-37.²¹

VALENTE, José A. **A Espiral da Espiral de Aprendizagem**: o processo de compreensão do papel das tecnologias de informação e comunicação na educação. Tese (Livre Docência) – Universidade Estadual de Campinas, Instituto de Artes. Campinas, SP, 2005. Anexo 3, p.209-232. Disponível em: <https://repositorio.unicamp.br/acervo/detalhe/857072>. Acesso em: 9 jan. 2026.

VALENTE, José A. Integração do pensamento computacional no currículo da educação básica: diferentes estratégias usadas e questões de formação de professores e avaliação do aluno. **Revista e-Curriculum**, v.14, n.03, p. 864-897, jul./set. 2016. Disponível em: <https://revistas.pucsp.br/index.php/curriculum/article/view/29051>. Acesso em: 9 jan. 2026.

VICARI, Rosa Maria; MOREIRA, Álvaro; MENEZES, Paulo Blauth; MENEZES, Crediné Silva de; NUNES, Daltro; LIVI, Maria Aparecida C. **Pensamento Computacional**: Revisão Bibliográfica. 2. ed. Porto

²¹ Também disponível como anexo em (Valente, 2005).

Alegre: Desenvolvido no âmbito do Projeto UFRGS/MEC, 2018. 192 p. v. 1. Disponível em: <https://lume.ufrgs.br/handle/10183/197566>. Acesso em 12 dez. 2022.

YAN, Lixiang; GREIFF, Samuel; LODGE, Jason M.; GASEVIC, Dragan. Distinguishing performance gains from learning when using generative AI. **Nat. Rev. Psychol.** v.4, p.435–436, 2025. DOI: <https://doi.org/10.1038/s44159-025-00467-5>. Acesso em: 14 jan. 2026.

WING, Jeannette. Pensamento Computacional – Um conjunto de atitudes e habilidades que todos, não só cientistas da computação, ficaram ansiosos para aprender e usar. **Revista Brasileira de Ensino de Ciência e Tecnologia**. Ponta Grossa, ano 2016, v. 9, n. 2, ed. 1, p. 1-10, 2016. DOI: <http://dx.doi.org/10.3895/rbect.v9n2.4711>. Acesso em: 31 ago. 2022.

Sobre os Autores

Lúcio S. Fassarella

lucio.fassarella@ufes.br

Professor titular do Departamento de Matemática Aplicada da Universidade Federal do Espírito Santo, doutor em Ciências Físicas pelo Centro Brasileiro de Pesquisas Físicas (2002), mestre em Matemática Pura e Aplicada pelo Instituto Nacional de Matemática Pura e Aplicada (1998). Graduado em Física pela Universidade Federal do Espírito Santo (1996).