

Hellon Canella Machado

**O Uso do Pensamento Computacional como
Recurso para o Desenvolvimento da
Aprendizagem Científica.**

Rio de Janeiro. Brasil

2018

Introdução

Segundo o PISA de 2015¹, apenas 30,6% dos alunos brasileiros alcançaram um patamar de competência básico na área de ciências, necessário para lidar com demandas simples da vida cotidiana. Este resultado coloca o Brasil na posição 63 entre os 70 países onde o teste foi aplicado.

De modo geral, a relação entre estudantes e as disciplinas científicas não tem sido marcada apenas por dificuldades conceituais, mas também por aquelas que envolvem elaboração de estratégias para a resolução de problemas, comuns na atividade científica. Dentre as dificuldades recorrentemente demonstradas por alunos nas aulas ciências, pode-se destacar (POZO; CRESPO, 1998):

- 1. Dificuldade para transpor a contextos novos os conceitos e estratégias de resolução de problemas aprendidos.** Quando o formato do problema muda, os alunos apresentam dificuldades para aplicar à nova situação os conceitos e algoritmos adquiridos.
- 2. Escassez de significado dos resultados obtidos.** Ao lidar com problemas propostos em sala de aula, eles se limitam a buscar uma fórmula para chegar a um resultado numérico. Aplicam, sem exercício crítico, o algoritmo de um modelo de problema, visando uma resposta “correta” e única. Desse modo, a aplicação de uma técnica se impõe à necessidade de compreender uma estratégia e seus potenciais casos de uso. Convertem, assim, o problema em um simples exercício autônomo.
- 3. Ausência de motivação para aprendizagem de temas científicos.**

Se atentarmos com detalhe essas dificuldades, teremos facilidade em perceber que elas têm sua origem na forma como conteúdos científicos são apresentados em sala de aula, e por ela são induzidas.

Em física, por exemplo, é comum a proposição de problemas cuja solução é alcançada por um procedimento ou rotina automatizável (“Qual é velocidade após 32s de um objeto lançado verticalmente com uma velocidade de 20m/s?”).

A ausência de motivação, por sua vez, tem sua origem numa percepção de ausência de significado dos temas apresentados, estimulada também pelos mesmos problemas propostos. Qual é apelo e capacidade mobilizadora de algo “sem significado” e, consequentemente, “sem utilidade”?

¹ PISA, sigla do *Programme for International Student Assessment* – Programa internacional para avaliação de alunos, em tradução livre – é uma proposta de avaliação conduzido pela OCDE (Organização para a Cooperação e Desenvolvimento econômico)

Cabe aqui uma observação de [Pozo e Crespo \(1998\)](#),

Os alunos não aprendem por não estar motivados, mas, por sua vez, não estão motivados porque não aprendem. A motivação não é uma responsabilidade dos alunos, mas também resultado da educação que recebem e, no nosso caso, de como lhes a ciência lhes é ensinada ([POZO; CRESPO, 1998](#)).

Afinal, como o ensino de ciências pode ser desenhado de forma a atacar o problema da motivação? Segundo a bem colocada definição de [Claxton \(1984 apud POZO; CRESPO, 1998\)](#), “motivar é mudar as prioridades de uma pessoa”, tomado como partida os seus próprios interesses para a construção de outros novos. Nesse sentido, o papel do ensino de ciência corresponderia à busca de envolvimento com os problemas pertinentes à realidade do aluno e a seus centros de interesse como meio de estímulo para o seu engajamento – componente essencial para a construção de aprendizagem significativa.

É justamente nessa premissa que esse trabalho se sustentará. Nele apresentamos e desdoblamos um conceito que vem ganhando relevância nos meios educacionais, que tem como ponto de partida um conjunto de práticas de resolução de problemas há décadas utilizado por programadores e cientistas da computação, e largamente empregado em praticamente todos os âmbitos da prática científica, conhecido como **pensamento computacional**.

Em essência, a adoção de um “pensamento computacional” corresponde justamente ao aprendizado e uso de formas de tratar problemas, de qualquer natureza, de modo a tornar possível a sua resolução por computadores.

A incorporação de tais práticas na sala de aula tem permitido que estudantes resolvam problemas científicos reais, ao estimulá-los a construir seus próprios modelos e validá-los utilizando ferramentas computacionais.

A literatura tem demonstrado um surpreendente potencial de engajamento dessas atividades, justamente porque elas trazem consigo um aspecto “mão na massa” que colabora com o aumento da percepção dos alunos de estarem resolvendo problemas científicos autênticos.

Trazer o pensamento computacional para o âmbito escolar equivale também familiarizar os estudantes com a forma que a ciência é praticada profissionalmente, o que os permitirão desenvolver um imagem mais fidedigna dessa atividade.

O desenvolvimento do tema nesse trabalho passará por três etapas. Na primeira, trazemos uma breve discussão sobre o impacto da computação na prática científica, desde o início da sua adoção no século XX até o momento presente, onde a explosão e abundância de dados tem exigido dos pesquisadores o emprego de técnicas baseadas em inteligência artificial.

No capítulo seguinte iremos trabalhar com o conceito de pensamento computacional propriamente dito, desdobrando a definição dada a ele pela literatura, bem como a dos elementos que o compõe, tais como as noções de algoritmo e abstração.

No terceiro e último capítulo, a nossa preocupação recai sobre a implicação e aplicação desse conceito para a aprendizagem científica. Baseando-nos na contribuição de Weintrop et al. (2016), iremos tentar distinguir quais *expertises* científicas buscamos que os alunos exercitem e desenvolvam ao propormos que eles pensem “computacionalmente”. Ao final, iremos apresentar um conjunto de três atividades de fácil execução em sala de aula, cuja finalidade é o exercício de algumas dessas competências.

1 Computadores e a prática científica

A evolução da computação nas últimas décadas, aliada ao seu barateamento, tem produzido impactos notáveis tanto na academia quanto na indústria. A disponibilidade de dispositivos mais baratos e com maior capacidade de processamento e armazenamento tem tornado a computação ubíqua.

Na academia, esse processo tem favorecido o surgimento de novas estratégias para exploração de fenômenos. Até meados do século 20, todo progresso científico foi conduzido apenas por interações entre atividades experimentais e analíticas¹. O surgimento da computação e o seu desenvolvimento desde então trouxeram novas formas de fazer ciência, tais como a simulação e a modelagem computacional, e mais recentemente, a mineração de dados e o aprendizado de máquina, úteis para a análise de grande volume de informação (DJORGOVSKI, 2005; WING, 2006).

O uso de simulações numéricas, por exemplo, se justifica ao permitir que um grande número de fenômenos muito complexos sejam analiticamente tratáveis. Em muitos casos essa é única forma de exploração possível. Mesmo na mecânica newtoniana mais simples, é possível resolver exatamente, apenas, o problema de dois corpos. Para $N \geq 3$, soluções numéricas são necessárias. Da astronomia podemos retirar alguns exemplos, como a formação de estrelas e galáxias e explosão estrelares – de modo geral, qualquer evento envolvendo turbulência (DJORGOVSKI, 2005).

O uso de métodos computacionais, tal como a simulação, tem expandido a abrangência de sistemas não lineares que tem sido explorados por modelos matemáticos e computacionais. Como lembra Weintrop et al. (2016), campos da ciência estão experimentando um renascimento de abordagens experimentais em razão do acesso facilitado a mais poder computacional.

O autor destaca que num passado recente, para muitos pesquisadores apenas o estudo de sistemas determinísticos era viável, tendo o termo ‘não-linear’, praticamente, o sinônimo de ‘insolúvel’. Havia desse modo a propensão à investigação computacional apenas de sistemas lineares. Esse quadro era especialmente verdade para muitas pesquisas em biologia e química.

Esse processo ganha especial relevância ao nos darmos conta da natureza caótica da ampla maioria dos fenômenos físicos. Sistemas lineares e determinísticos são portanto exceções, e não a regra (WEINTROP et al., 2016).

¹ Em sentido mais amplo, quando mencionamos ‘atividade analítica’ estamos nos referindo à utilização de aparato matemático para resolução teórica de problemas. Ao mesmo tempo, o termo análise também pode fazer referência ao emprego da análise matemática, ramo da matemática que lida com conceitos do cálculo diferencial, tais como diferenciação, integração, e séries infinitas.

Outros agentes de transformações da prática científica, embora mais recentes e em fase nascente, são as novas possibilidades trazidas pela abundância e pelo barateamento do armazenamento grandes volumes de dados. Vivemos uma era onde a disponibilidade de dados gerados por câmeras, sensores, execução de simulações e registro de interação humana crescem exponencialmente.

Nesse cenário, como ressalta Djorgovski (2005), o foco de valores tem mudado da propriedade de dados ou de instrumentos para a reunião de *expertise* que tornam possíveis a extração de significados desse volume de informação.

A abundância traz consigo muitos desafios. A taxa com que cientistas e engenheiros têm coletado e produzido dados vem exigindo avanços nas estratégias de análise. O acúmulo chegou a um nível de complexidade que, de certo modo, tem sido impossível fazer qualquer tipo de investigação superficial utilizando técnicas convencionais, tendo como instrumento apenas a percepção humana. Lidar com esse conjunto desestruturado e rico de dados, extraíndo dele significado, tem sido portanto uma das batalhas da revolução científica e industrial em curso (DJORGOVSKI, 2005). Nesse contexto, o emprego de inteligência artificial tem sido essencial.

Em linhas gerais, tal “inteligência”, expressa em técnicas de aprendizagem de máquina (do inglês: *machine learning*), baseia-se no uso algoritmos que instruem computadores como avaliar dados e deles extrair padrões e correlações, permitindo-os, de forma extraordinária, a fazer previsões. E esse processo tem um componente recursivo: quanto mais análises são feitas, mais experiência e competência são adquiridas. Ou seja, mais ‘inteligentes’ essas máquinas se tornam.

Escobar (2017) propõe uma visão simplificada das interações humanas que facilita o entendimento desses algoritmos. Como descreve, ao conhecemos alguém pela primeira vez, baseando-nos em modelos pessoais, somos capazes de dizer nos primeiros minutos se essa pessoa nos transmite boa ou má impressão. Para cada nova pessoa que encontramos, avaliamos algumas de suas características e as registramos. Esse processo nos permite refinar e recompor modelos sociais que irão influenciar outras percepções em interações futuras.

É exatamente nesse princípio recursivo que se fundamenta o aprendizado de máquina: categorizar dados de acordo com suas características com o fim de compor e refinar modelos.

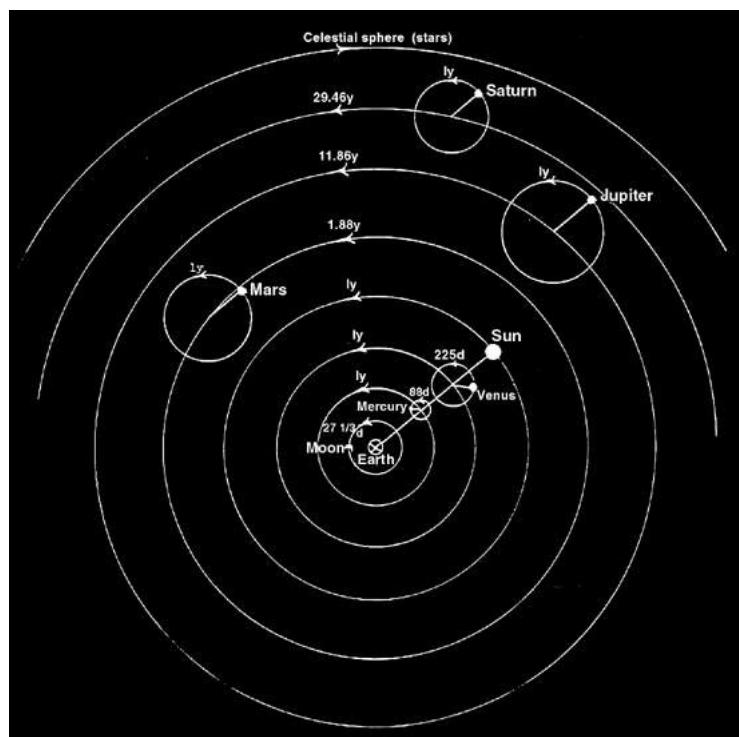
Esse processo tem se provado extremamente eficiente na previsão da configuração de sistemas estocásticos. Tomemos por exemplo a necessidade de prever o tempo, onde há dominância de comportamentos turbulentos. Em um estudo recente (PATHAK et al., 2018 apud VUTHA, 2018), mostrou-se a eficiência do uso de algoritmos de inteligência artificial para previsões ao longo de um período muito maior do que se imaginou possível.

Um ponto digno de nota: o algoritmo utilizado não continha nenhuma informação sobre as equações subjacentes.

Há atualmente questionamentos sobre até que ponto o uso intensivo de ‘robôs oráculos’ levará à perda do interesse dos pesquisadores em descrever fenômenos em termos de equações e princípios unificadores. Esse debate é sintetizado pela confronto das noções de “predição” e “entendimento”.

[Vutha \(2018\)](#) expõe os elementos desse debate com um exemplo histórico. Como destaca, durante mais de um milênio, o movimento dos planetas eram descritos a partir de um modelo elaborado por Ptolomeu, cujos métodos lançavam mão de cálculos misteriosos envolvendo sobreposição de círculos. Apesar de ignorar a teoria da gravidade e de ter a Terra no centro do Universo, essa representação foi extremamente eficiente na sua capacidade preditiva. Ao mesmo tempo, pode-se dizer que ela era incompleta na medida em que não oferecia nenhum entendimento capaz de explicar o seu funcionamento.

Figura 1 – Modelo geocêntrico de Ptolomeu



Fonte: [Hatch \(1998\)](#)

A descrição do movimento dos corpos celestes só foi finalmente compreendida a partir da equações diferenciais descobertas por Isaac Newton. Com elas tem sido possível, desde então, prever a trajetória de todo e qualquer planeta do sistema solar.

A qualidade da proposta de Newton estava na possibilidade de oferecer não apenas

a descrição de trajetórias, mas por permitir também que entendamos o porquê que elas são de uma e não de outra forma. Ao nos trazer equações, ele nos facultou a compreensão do fenômeno do movimento a partir da ótica de princípios unificadores. E é exatamente nesse ponto que reside o poder da descrição matemática. Como analisa [Vutha \(2018\)](#), se formos capazes de extrair de um fenômeno complicado dois ou três princípios, podemos dizer então que o compreendemos.

Porém, dada a dominância de fenômenos complexos na natureza, tem sido extremamente difícil extrair de muitos deles princípios simples. Descrevê-los, portanto, a partir da obtenção de equações universalmente válidas pode ser uma forma um tanto ineficiente de gerar previsões relevantes ([VUTHA, 2018](#)). O emprego de técnicas de *machine learning*, nesse sentido, tem sido essencial.

A inteligência artificial tem ocupados outros espaços além da pesquisa científica. Desde robôs que leem a web para tomar decisão de investimentos ([VERHAGE, 2017](#)) e carros que se auto dirigem, até contextos relativamente mais simples como tradutores automáticos e mecanismo buscas. Já no mundo do trabalho, um sem-número de projeções apontam para a substituição progressiva de quantidade considerável de atividades laborais por robôs.

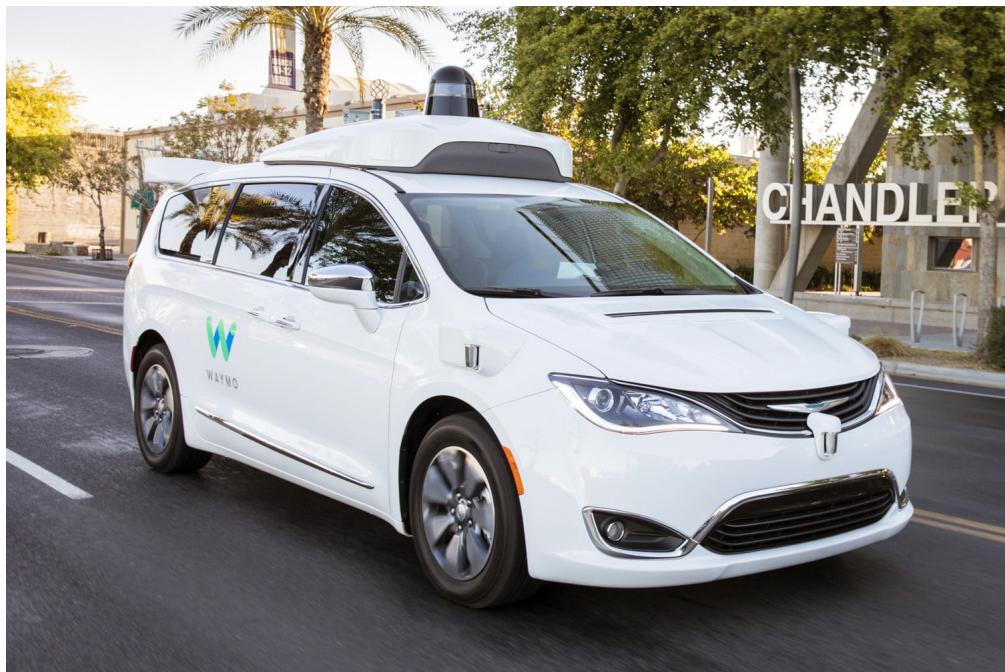
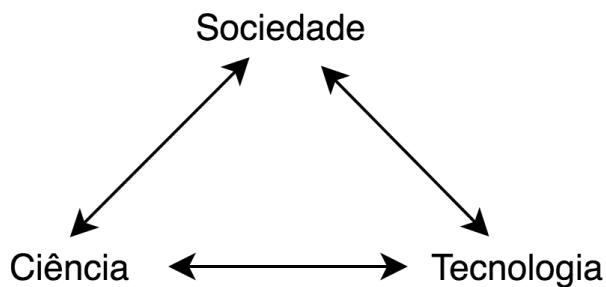


Figura 2 – Carro autônomo da Waymo, subsidiária de veículos da Alphabet (matriz do Google)

A produção estratosférica de dados e a invasão da inteligência artificial nos meios científicos e na indústria é sintomático da interação dinâmica entre a ciência, a tecnologia, e a sociedade em ciclos que se retroalimentam.

Novas descobertas levam a novas perguntas que exigem mais coleta de dados. Quando analisadas, essas informações são utilizadas para refinar e ajustar modelos, levando a novas perguntas e mais produção de dados. Esse processo tende a culminar na criação de novas tecnologias. Tecnologias, por seu turno, inspiram usos sociais criativos que quase sempre criam demandas por mais descobertas científicas e mais tecnologia. Sob esse aspecto, podemos dizer que esses três atores desempenham o papel de forças motrizes da computação ([WING, 2008](#)).

Figura 3 – Forças motrizes da computação



Adaptado de [Wing \(2008\)](#)

O barateamento dos custos de produção científica é outro aspecto relevante dessas transformações. À medida que a internet e dados, consequentemente, se tornam mais acessíveis, qualquer pessoa com boas ideias e bons hábitos de trabalho, de qualquer lugar, pode fazer ciência de primeira linha, comunicar seus resultados e aprender com a comunidade científica. Essa possibilidade é especialmente benéfica para países e instituições que não dispõem de instalações de pesquisa sofisticadas ([DJORGOVSKI, 2005](#)).

Sobre o papel da ciência da computação no desenvolvimento científico, [Djorgovski \(2005\)](#) faz um consideração provocativa:

[...] a ciência da computação aplicada está desempenhando o papel que a matemática fez do século XVII ao século XX: fornecer uma estrutura ordenada e formal e um aparato exploratório para outras ciências. Além de seu aparentemente feliz caso com a teoria das cordas, é difícil dizer o que a matemática está fazendo para outras ciências hoje. A maioria dos cientistas de matemática que utilizamos hoje foi desenvolvida há mais de um século. ([DJORGOVSKI, 2005](#), p. 131. Tradução nossa)

E é do seio dessa reflexão que nasce a noção de um “pensamento computacional”. Tema que desdoblaremos a seguir.

2 Pensamento computacional

A noção de pensamento computacional tem ganhado força desde 2006, ano em que a professora Jeannette Wing, então professora da Universidade Carnegie Mellon, publica um artigo seminal no qual propõe um conjunto de atitudes, ou abordagens, para a resolução de problemas fundamentadas na forma como cientistas da computação e programadores tratam problemas computacionais.

A definição clássica que muitos autores dão para o conceito é extraído do próprio trabalho da autora que estabelece que pensamento computacional consiste “numa abordagem para a solução de problemas, desenho de sistemas e entendimento do comportamento humano que se vale de conceitos fundamentais para ciência da computação” ([WING, 2006](#), Tradução nossa), ou ainda, na capacidade de formular problemas e expressar suas soluções de forma suficientemente clara de tal modo que um computador possa executá-las.

Para o desenvolvimento do tema, faremos a apresentação de três conceitos que consideramos básicos. São eles: os **algoritmos**, o **problema computacional** e a **abstração**. Ao final demonstraremos como esses elementos se articulam para compor o que se entende por pensamento computacional.

2.1 Algoritmos

Em termos simplificados, um algoritmo nada mais é do que uma solução para um problema que satisfaz as seguintes condições ([PIWEK, 2016](#)):

- deve apresentar uma lista sequenciada de passos que levam à solução do problema
- deve ser um processo finito
- deve resolver qualquer instância do problema em questão

Na figura 4 lê-se um conjunto de instruções para somar três números inteiros, extraídos do livro *Practical Arithmetick in Four Books* do autor John Gough, publicado pela primeira vez em 1767. Sob o título ‘General rules’, lemos

Coloque os números de modo que cada algarismo possa ficar diretamente abaixo (ou na mesma linha perpendicular) dos algorismos de mesmo valor, ou seja: unidades em unidades, dezenas em dezenas, centenas em centenas.[...] Em seguida, desenhando uma linha abaixo deles; comece a adição no primeiro lugar (ou unidades) e some todas os algorismos naquele lugar, e se a soma delas for menor que dez, coloque-a abaixo da linha abaixo de seu próprio lugar; mas se a soma for superior a dez,

Figura 4 – Algoritmo de soma de três números inteiros

C H A P. II.
A D D I T I O N.

20. ADDITION is the joining or collecting several numbers into one, or finding a number which shall be equal to any given numbers altogether.

GENERAL RULE.

Let the numbers marked A B C, be given to be added.

1. Place the numbers so that each figure may stand directly underneath (or in the same perpendicular row with) the figures of the same value, that is: units under units, tens 67430 under tens, hundreds under hundreds, &c.

Then drawing a line under them; begin the Addition at the first place (or units) and add together all the figures in that place, and if their sum be under ten, set it down below the line underneath its own place; but if their sum be more than ten, set down only the overplus above the ten (or tens) and so many tens as the sum of these units amount to, carry to the place of tens, adding them and the figures which stand in the place of tens together; then proceed in the same manner to the third place, or hundreds, and so from place to place to the last, and set down the whole sum of the last place.

Application

Fonte: Gough (1813)

estabeleça apenas o excedente acima das dez (ou dezenas) e algumas dezenas, à medida que a soma dessas unidades se eleva, carregue para o local das dezenas, adicionando-as e os algorismos que estão no lugar de dezenas juntos; em seguida, proceda da mesma maneira até o terceiro lugar, ou centenas, e de um lugar para outro até o último, e estabeleça a soma total do último lugar.

Perceba como essas instruções satisfazem as condições elencadas anteriormente: nela vemos um problema (cálculo da soma de três número inteiros) sendo resolvido de forma encadeada ao longo de um processo finito. Nota-se também que os passos acima

facultariam a soma de quaisquer outros três números inteiros, atendendo a exigência de “resolver qualquer instância do problema”.

2.2 Problema Computacional

A tarefa de definir as instâncias de um problema equivale a determinar para quais perguntas um algoritmo deve oferecer respostas. Em problema de divisão, por exemplo, onde é preciso calcular a razão entre a e b , o par $a = 1.5$ e $b = 9.365$ representa uma delas. Já $a = 65.4$ e $b = 77$ compõe outra. Numa única sentença podemos resumir que qualquer par de reais, onde $b \neq 0$, representa uma instância válida.

A decisão de usar um computador para a resolução de qualquer problema exige que definamos tais instâncias. Sob o ponto de vista da ciência computação, ao procedermos dessa forma, distinguindo claramente os contornos do problema, definimos um “problema computacional”.

A definição clara das instâncias do problema (criação de um problema computacional) e dos passos exatos para sua solução¹ (elaboração de um algoritmo) nos permitirá automatizar a sua execução.

2.3 Abstração

A noção de abstração é outro componente que necessita ser compreendido ao tratarmos do assunto. Nas palavras de Wing (2006),

A abstração é usada na definição de padrões, generalização de instâncias e parametrização. É usado para deixar um objeto representar muitos. Ele é usado para capturar propriedades essenciais comuns a um conjunto de objetos enquanto oculta distinções irrelevantes entre eles (WING, 2006, p. 1. Tradução nossa)

Abstrair, como enfatiza o trecho acima, equivale simplesmente a conduzir um processo de generalização, onde incorporamos parte dos detalhes da realidade em um modelo, ao mesmo tempo que descartamos outros. Estabelece-se assim uma relação entre dois níveis: a realidade e sua representação.

A figura 6 ilustra essa discussão. Nela vemos a obra “A Traição das Imagens” por René Magritte (1898-1967), onde lê-se “Ceci n'est pas une pipe” (“Isto não é um cachimbo”). Temos aí provocação que evidencia que um modelo nada mais é do que uma representação, e não a realidade em si.

¹ Ao tratarmos computacionalmente um problema devemos ser capazes também de distinguir quando e porque, eventualmente, ele não tem solução.

Figura 5 – Relação entre a realidade e o seu modelo

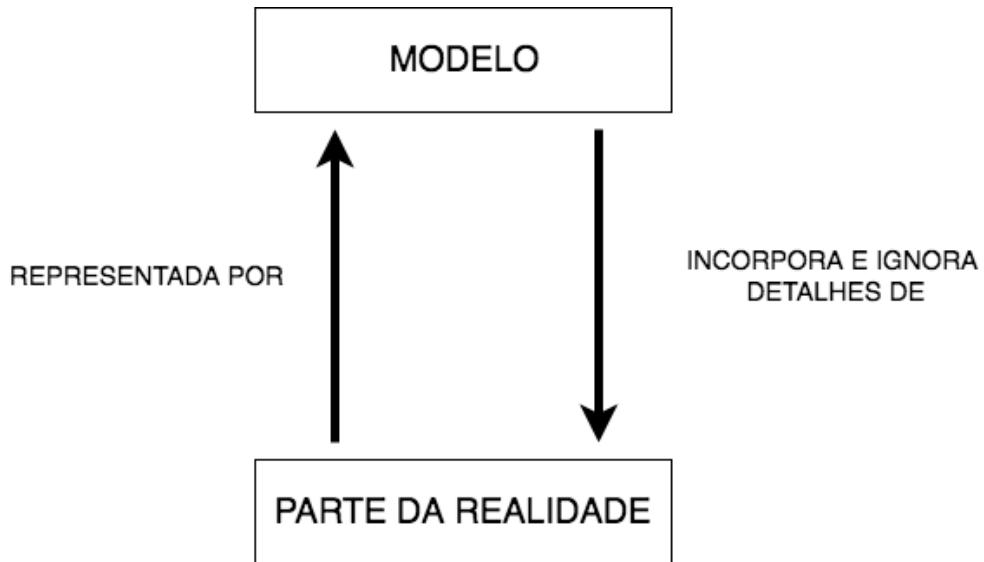
Adaptado de [Piwek \(2016\)](#)

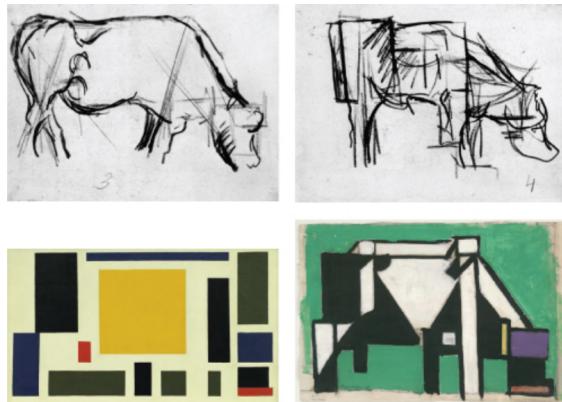
Figura 6 – “A Traição das Imagens” por René Magritte, 1929



O grau de detalhamento do modelo depende das circunstâncias do problema, bem como das necessidades de quem modela. Por exemplo, ao analisarmos a trajetória de lançamento de um foguete, podemos descartar suas dimensões e eliminar possíveis efeitos aerodinâmicos. Essa abordagem é extremamente conveniente para o ensino de primeiras noções de física básica, mas demasiadamente simplória no contexto da condução de um programa aerospatial.

Na figura 7 temos a ilustração dessa ideia: a mesma “realidade” vaca representada por quatro modelos com diferentes graus de detalhes incorporados.

Figura 7 – Quatro representações de uma vaca por Theo van Doesburg, 1917–1918



Fonte: [Piwek \(2016\)](#)

É interessante notar a proximidade do significado para abstração discutido até aqui com aquele proposto pelo filósofo John Locke, segundo o qual formação de uma abstração corresponde a uma transformação durante a qual “ideias tomadas de seres particulares se tornam representantes gerais de todos do mesmo tipo” ([LOCKE, 1690/1979 apud SENGUPTA et al., 2013](#), p. 354. Tradução nossa).

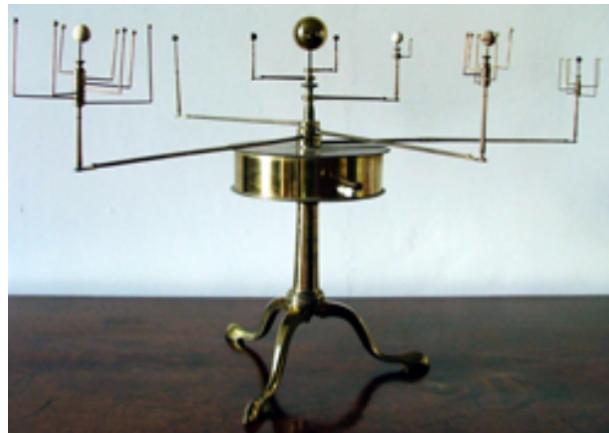
2.3.1 Modelos e módulos: duas possibilidade de abstração

[Piwek \(2016\)](#) distingue dois tipos de abstrações: a “abstração como modelo”, onde detalhes da realidade observada são desconsideradas em favor de outras – o mesmo trabalhado até aqui – e a “abstração como encapsulamento”, processo no qual organizamos nosso modelo em módulos ou cápsulas que permitem a composição de sistemas mais complexos.

Para diferenciarmos esses dois tipos tomemos como ponto de partida o planetário mecânico ilustrado na figura 8. Nele vemos uma simulação do sistema solar, um modelo, que como tal ignora certas aspectos da realidade. Neste caso em específico temos a desconsideração típica de representações astronômicas onde as proporções entre as dimensões dos planetas e a distância relativas entre eles é extremamente grande. A construção de modelos, ignorando e incorporando detalhes, é o que [Piwek \(2016\)](#) chama de “abstração como modelo”.

O mesmo dispositivo incorpora a noção do que o autor chama de “abstração como encapsulamento”. Na figura 9 vemos um sistema de rodas dentadas típico do que podemos encontrar no interior de um planetário mecânico. São elas que permitem o movimento das esferas que representam os planetas. A superfície metálica que vemos na figura 8 cumpre o papel de interface do modelo ao escondê-las, deixando externamente visível apenas o que

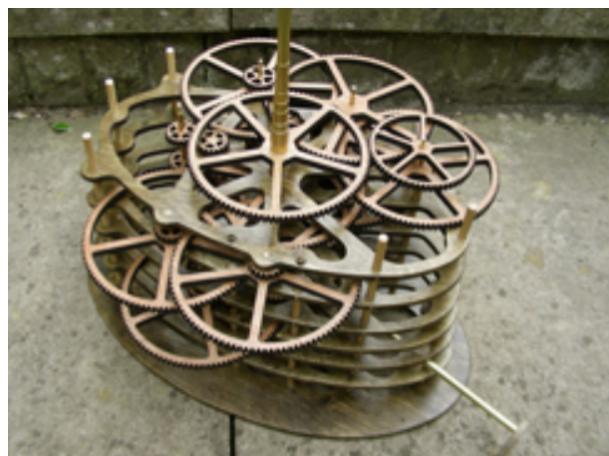
Figura 8 – Planetário mecânico



Fonte: [Piwek \(2016\)](#)

é relevante: o movimento de translação e rotação.

Figura 9 – Engrenagem de um planetário mecânico



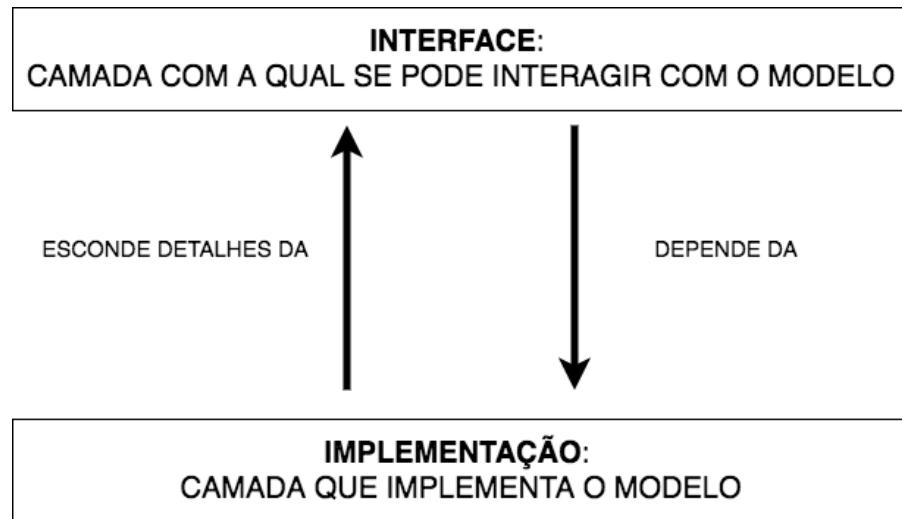
Fonte: [Piwek \(2016\)](#)

Distingue-se assim a formação de duas camadas durante o processo de encapsulamento: a interface com a qual se pode interagir com o modelo (o invólucro metálico do planetário) e a sua implementação propriamente dita (sistema de engrenagens encerradas pela interface). A figura 10 ilustra a relação entre elas.

O uso do encapsulamento é essencial na ciência da computação. Nesse contexto, essa estratégia toma forma em funções e estrutura de dados². Tomemos um exemplo envolvendo funções. Considere a necessidade de calcularmos a seguinte expressão:

² Por estrutura de dados nos referimos às diversas possibilidade de organização e relacionamento de dados que uma linguagem oferece. Dentre as mais comuns estão os vetores, as listas, as pilhas...

Figura 10 – Interface e implementação – dois aspectos de um modelo



Fonte: [Piwek \(2016\)](#)

$$\frac{(2 + 3 + 5 \times 7)}{5}$$

Usando uma linguagem de programação poderíamos reescrever a equação acima do seguinte modo:

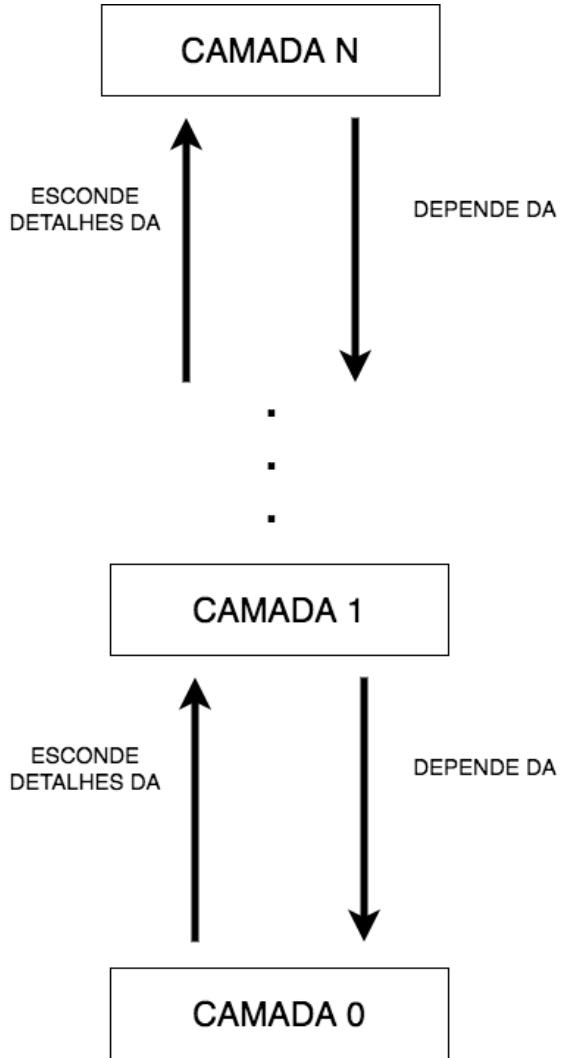
$$razao(soma(2, 3, produto(5, 7)), 5)$$

Perceba a formação de cápsulas. Por exemplo, ao escrevermos $produto(a, b, c\dots)$, estamos modularizando o processo de multiplicação, pois todos os detalhes dessa operação estão invisibilizados atrás de uma interface, cujo nome decidimos chamar como “produto”. Aplica-se o mesmo raciocínio aos casos das funções $soma(a, b, c\dots)$ e $razao(a, b)$.

A decomposição do modelo ou sistema pode ganhar tantas unidades quanto se queira, até atingirmos o nível binário (0 e 1s). É exatamente na possibilidade de aplicação recursiva dessa estratégia que a construção de sistemas complexos se tornam viáveis. Esse aspecto é ilustrado na figura 11.

De fato, ao encapsular, programadores buscam expressar-se apenas em termos de unidades representativas. Essa tática permite a eles raciocinar somente em termos do problema que estão resolvendo, e os protege assim das complexidades próprias do ambiente computacional com que estão trabalhando, tal como a necessidade de administrar diretamente o uso da memória, por exemplo. Ao mesmo tempo, por favorecer a organização e a estruturação lógica do código fonte, como vantagem adicional essa abordagem facilita manutenções futuras do sistema.

Figura 11 – Decomposição em camadas de um sistema

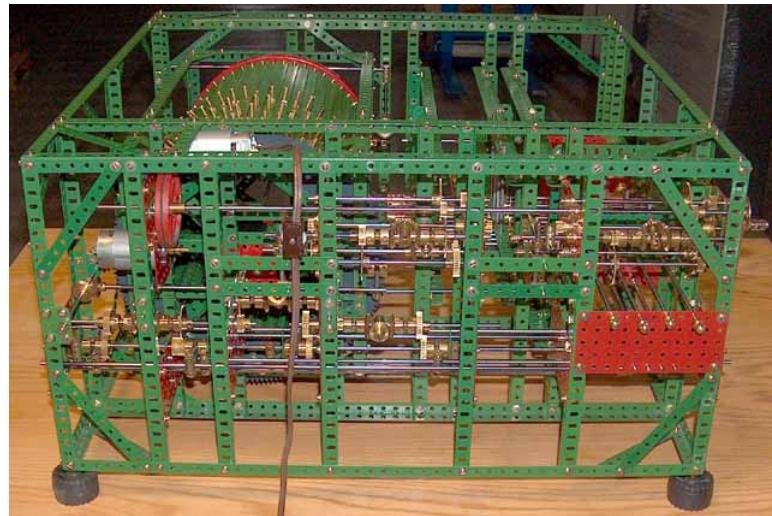


Fonte: [Piwek \(2016\)](#)

Todo o desenvolvimento da computação está assentado na decomposição modular de sistemas. Celulares, computadores, sistemas operacionais, automóveis... Toda e qualquer tecnologia operada por dispositivos eletrônicos. Há exemplos históricos, contudo, onde esse princípio é aplicado apesar de não haver nenhum substrato eletrônico. Na figura 12 vemos o primeiro computador programável desenvolvido com o propósito de executar operações matemáticas. Conhecido como Meccano, ele foi proposto por Charles Babbage (1791–1871). Movido a vapor, na imagem temos um exemplar construído em latão e ferro.

Em suma, a criação de abstrações tanto por modelagem quanto por encapsulamento nos permitem administrar as complexidades do mundo real. Criando modelos, buscamos diminuir os aspectos ruidosos do mundo real em favor de elementos relevantes para a análise do problema, enquanto que ao encapsularmos, como diz [Piwek \(2016\)](#), evitamos

Figura 12 – Meccano – Implementação do primeiro computador programável proposto por Charles Babbage



Fonte: [Piwek \(2016\)](#)

ser vitimizados pelos intrincados detalhes do “mundo dos computadores”.

2.4 Um apanhado geral

Afinal qual é o papel dos algoritmos, dos problemas computacionais e das abstrações na composição da noção de pensamento computacional?

Eis como esses componentes se articulam: os problemas reais, sejam eles de natureza puramente matemática, ou aqueles conectados à realidade física – como os fenômenos estudados pela física e as transações de compra e venda no mercado de ações – podem ser reformulados de modo a se transformarem em um problema computacional. Para tanto, deve-se definir o escopo do problema, ou seja, as perguntas a serem respondidas, bem como seus os casos limites.

Nesta etapa, entra em cena o conceito de modelagem: um processo de identificação de agentes³ e interações, bem como de atribuição de variáveis, cujas quantidades são consideradas relevantes para descrição.

O modelo é, por essência, uma expressão da escolha daquilo que é considerado essencial. Uma redução da realidade. Uma abstração composta por outras menores: seus agentes, as relações entre eles e os processos que os envolvem. E é justamente esse arranjo que nos permite enxergar essas unidades como módulos.

³ entidades físicas ou sociais, sejam elas um grupo de elétrons ou de indivíduos

Para se que possa expressar os aspectos do modelo em termos de problemas computacionais, faz-se necessário estruturar as perguntas a serem respondidas de modo que um computador possa comportá-las, a partir da formulação de um algoritmo. E da implementação desses passos por uma linguagem de programação resulta um programa, uma “máquina” que automatiza a solução buscada.

Nesta etapa, o arranjo em módulos apontado acima repercute na forma como os programa são redigidos. Para a representação de cada das unidades do modelo são escritas funções e utilizadas estrutura de dados apropriadas, como, por exemplo, as classes, árvores e as listas. Camadas de abstração também compõem a comunicação entre o que é expresso por nós seres humanos, utilizando uma linguagem de programação (alto nível), e o que máquinas interpretam (baixo nível) – sequenciamento de conjuntos de dois níveis de tensão, 0s e 1s.

Por sua parte, a automação de modelos nos permite conhecer melhor os aspectos do problema que estamos investigando. Na engenharia, por exemplo, o uso dessa estratégia permite a investigação da evolução de fenômenos em condições limites, o que em alguns casos pode evitar a ocorrência de tragédias. Ao mesmo tempo, por oferecer um contexto que proporciona o ganho de *insights*, ela facilita o desenho de procedimentos para otimização de recursos materiais e humanos.

Esse ganho de conhecimentos introduz um componente recursivo na nossa investigação: de posse de mais informações somos induzidos a formular mais perguntas, e, consequentemente, dar início a novos ciclos de modelagem e automação.

É justamente à esse sequenciamento, que se inicia na colocação de perguntas e na criação de um modelo, e termina na construção de uma rotina computacional para automatizar aspectos do problema ou fenômeno estudado, e que ainda pode se ser reiniciado de acordo com o interesse de aprofundamento, que se designa a expressão “pensamento computacional”.

Dedicaremos o capítulo seguinte à discussão da extensão e das implicações desse conceito para a aprendizagem científica.

3 Implicações para aprendizagem científica

O desenvolvimento da computação e o amadurecimento da noção de pensamento computacional traz consigo várias implicações e oportunidades educacionais.

De partida, pode-se dizer que a efetivação do uso do computador como ferramenta para resolução de problemas exige a formação de capital humano. Normalmente, espera-se que estudantes tenham acesso à universidade para introdução das primeiras noções de programação e ciência da computação. Sabe-se contudo que eles se mostram cada vez familiarizados com *smartphones* e outros dispositivo computacionais. Por isso podemos nos perguntar: por que não antecipar essa instrumentalização?

Esse questionamento se torna ainda mais pertinente em face do percentual reduzido de portadores de grau universitário. A demanda por profissionais com experiência na resolução de problemas computacionais tem crescido numa proporção superior a que academia tem formado. Esse desequilíbrio tem levado grandes empresas de tecnologia como o Google e IBM a dispensar a formação universitária como pre-requisito para contratação (PURTILL, 2018).

A motivação para essa antecipação não é apenas de caráter laboral. Como discutiremos ao longo desse capítulo o uso de computadores e do pensamento computacional oferece excelentes contextos para desenvolvimento de competências científicas. O reverso também é verdadeiro: problemas científicos e matemáticos são domínios excelentes para a aplicação e exercício do pensamento computacional. É nessa tese que esse trabalho se apoia.

No capítulo 2 mostramos o como conceitos de abstração, decomposição de problemas e simulação, aliados a práticas de representação de problemas se estruturam em torno do pensamento computacional. Ao mesmo tempo que são centrais na ciência da computação, estes elementos também são fundamentais para o desenvolvimento de modelos e para a compreensão e resolução de problemas num largo espectro de disciplinas matemáticas e científicas (SENGUPTA et al., 2013).

A literatura tem demonstrado de diversas formas o como a aprendizagem de programação – uma das práticas de representação – em conjunto com conceitos de outros domínios pode ser mais fácil do que aprender cada um desses tópicos individualmente. Essa abordagem é destacadamente diferente da forma como o ensino de computação tem sido trabalhado no ambiente escolar, onde as atividades propostas focam em aspectos apartados da realidade.

Em um estudo citado por Weintrop et al. (2016), descobriu-se que a introdução

de programação no contexto da modelagem de fenômenos físicos e químicos, durante as atividades de alunos de graduação, resultaram no aprendizado mais efetivo de programação, e no aumento do engajamento com a área de domínio das tarefas.

A literatura também nos apresenta outros exemplos de contextualização de atividades computacionais com o ensino de ciências. Dentre algumas propostas práticas, podemos destacar o uso eficiente do software NetLogo ([WILENSKY, 1999](#)) na introdução de noções de probabilidade e estatística, como relatado por [Abrahamson e Wilensky \(2005\)](#), e na modelagem de fenômenos epidemiológicos, tal como reportado em ([LEE et al., 2011](#)).

Vários estudos apontam que o enriquecimento trazido por essa abordagem conjunta, associada com o caráter “mão na massa” de muitas dessas atividades, produz impactos positivos na forma como os alunos percebem as aulas de ciências ([LEE et al., 2011; BARR; STEPHENSON, 2011](#)). A sensação de estar resolvendo um problema autêntico com real aplicabilidade e, em muitos casos, inserido na realidade do qual fazem parte, produz aumentos sensíveis dos níveis de engajamento.

[Weintrop et al. \(2016\)](#) sugere que o aumento da atratividade dessas disciplinas, apresentadas dessa forma, também pode favorecer o aumento da representação de grupos minoritários nos meios científicos, tais como mulheres, negros e transgêneros.

A atenção crescente que educadores têm dado ao tema do pensamento computacional tem sido apoiado também pela expectativa de inverter a relação entre os estudante e a tecnologia. A pergunta que se coloca é: como de meros usuários eles podem vir a tornar criadores?

[Resnick \(2012\)](#) argumenta que apesar de serem vistos como “nativos digitais”¹, a geração nascida durante esse século está familiarizada apenas com o consumo de tecnologia, e não necessariamente com a sua criação ou produção. Para ele, o benefício de dotá-la com essa capacidade criativa não estará apenas nessa ou naquela tecnologia desenvolvida pelo estudante, mas sim nas competências cognitivas adquiridas por ele durante o processo de criação. Dentre algumas delas, o autor destaca:

1. a capacidade de pensar em termos sistêmicos, ou seja, em função do comportamento agregado resultante da interação dos componentes de sistema – “o todo não é a soma das partes”;
2. a habilidade para trabalhar em equipe;
3. a competência para estruturar e planejar as etapas de um projeto;
4. a facilidade para experimentar novas ideias (prototipagem);

¹ A expressão “nativo digital” foi cunhada pelo escritor Marc Prensky para designar aqueles que desde o seu nascimento tiveram a oportunidade de interagir com tecnologias digitais, tais como videogames, computadores, telefone celular, iPhones, iPads...

5. a destreza para decompor ideias complexas em unidades menores (abstrações);
6. a perícia necessária para investigar defeitos ou comportamentos inesperados (deputração);
7. a inteligência emocional necessária para lidar com frustrações e perseverar em face de dificuldades que surgem ao longo de um projeto.

Como ressalta, mesmo que o estudante não venha a se tornar um engenheiro ou um cientista da computação todas essas competências são importantes para todo e qualquer tipo de atividades. Analogamente, mesmo que poucos deles se profissionalizem como escritores, a capacidade para redigir textos é fundamental para qualquer um.

[Resnick \(2012\)](#) direciona o seu argumento para justificar os benefícios do ensino de programação. Veremos, contudo, que o desenvolvimento dessas competências criativas – todas elas associadas ao pensamento computacional – podem passar por caminhos diferentes da habilidade de escrever código, propriamente dita.

Outras oportunidades educacionais estão associadas às respostas que essa abordagem oferece para vários problemas relacionados ao tema da avaliação. O ensino de ciências é dominado por uma tradição explicativa cujo objetivo é a simples transmissão de dados e conceitos pelo professor. Nesse contexto, convencionou-se que o ciclo de ensino e aprendizagem se fecha quando os alunos são capazes de reproduzir essas informações numa prova cronometrada. No ensino de física, em especial, entende-se que o aprendizado se efetiva quando eles se mostram capazes de resolver três ou quatro variações de um problema discutido em sala de aula – quase sempre de natureza algébrica.

O que se observa, portanto, é um modelo avaliativo indutor de um comportamento autônomo do estudante, em prejuízo do desenvolvimento da sua capacidade crítica e analítica.

Por outro lado, o desenvolvimento de atividades sustentadas pelo pensamento computacional tem como foco a resolução de problemas abertos, quase sempre relacionados à construção e não à imposição de modelos (abstrações). Conceitos e princípios científicos são trabalhados sob a perspectiva de que não há modelos “corretos”, mas sim “apropriados”, que melhor respondem a determinadas questões.

Dessa forma, a absorção de tais conteúdos conceituais se dá pela necessidade de articulá-los, por exemplo, na construção de uma animação, ou até mesmo de um jogo. Nesse ambiente, o aluno tem a oportunidade de atestar a validade de um princípio físico pela “qualidade” do caminhar do seu personagem, por quanto bem o seu comportamento reflete a realidade. O aprendizado das três leis de Newton nesse cenário nascerá da análise, da observação, e não da simples “decoreba”.

Os marcos avaliativos introduzidos por essa abordagem estão ancorados na perspectiva de que a aprendizagem de conteúdos científicos devem ser encarados como um meio, um *playground* para o desenvolvimento de competências cognitivas, e não como um fim em si.

Por esse motivo pode-se afirmar que o uso do pensamento computacional estimula comportamentos em sala de aula mais compatíveis com a natureza da investigação científica. E por ela se desenvolver cada vez mais em bases computacionais, como discutido no capítulo 1, apoiar a sua aprendizagem na construção e na análise de modelos computacionais permite que os alunos tenham uma visão mais realista e coerente com a forma que ela é exercida profissionalmente.

Dadas as motivações para exercício do pensamento computacional, ainda permanecem algumas questões de natureza práticas a serem respondidas e conciliadas para que essa abordagem se viabilize em sala de aula. Mesmo que até aqui tenhamos esboçado algumas definições, elas ainda não foram capazes de oferecer respostas precisas sobre como esse conceito pode ser materializado em sala de aula. Essa é a tarefa que desejamos cumprir no capítulo seguinte.

3.1 Significado do pensamento computacional no contexto da sala de aula

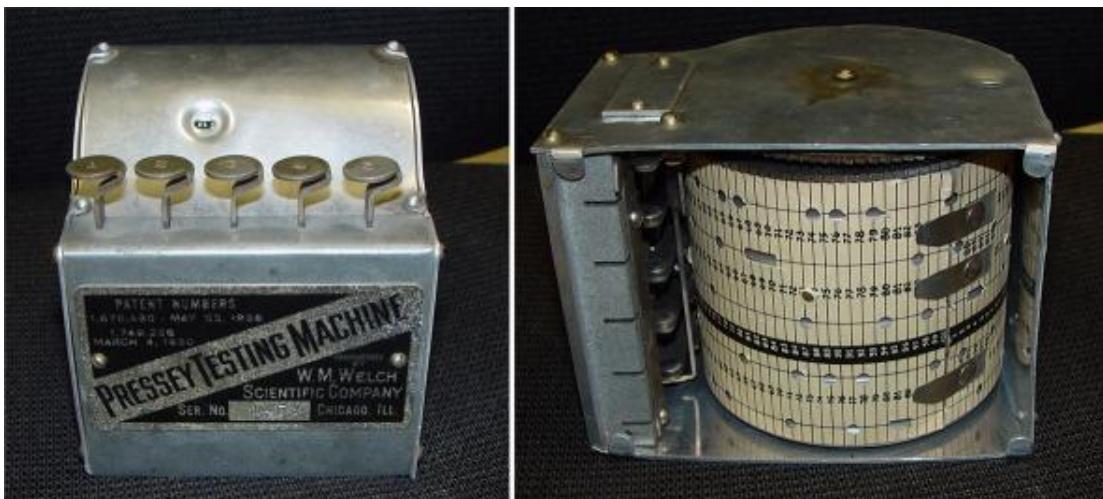
O emprego de dispositivos autômatos como ferramentas educacionais remonta às “máquinas de ensinar”, cujo movimento dependia da seleção da alternativa correta de um dos itens de uma questão múltipla escolha. Esse mecanismo foi proposto e desenvolvido pelo professor de psicologia da Universidade de Ohio, Sidney L. Pressey.

O uso da expressão “pensamento computacional”, contudo, tem uma origem mais recente. O seu primeiro registro é observado no livro *Mindstorms: Children, computers and powerful ideas*, da década de 1980, onde Seymour Papert discorre sobre a oportunidade trazida por computadores para o ensino de matemática.

A popularização da expressão e o consequente interesse da comunidade acadêmica veio apenas com a publicação do artigo por Wing (2006), no qual é proposto “um conjunto de atitudes e habilidades universalmente aplicáveis” assentadas na forma como cientistas da computação e programadores resolvem problemas.

Apesar do intenso debate despertado pelo artigo e do reconhecimento das questões e oportunidades educacionais implicadas no tema, o desenvolvimento de atividades e abordagens para sala de aula tem sido especialmente desafiador em face da parcial ou completa inexistência de um significado consensual para o conceito que seja adequado para este ambiente. A definição para o pensamento computacional resultante da discussão

Figura 13 – “Máquina de ensinar” de Pressey



Fonte: [Oremus \(2015\)](#)

proposta por [Wing \(2006\)](#) foca apenas na contribuição da ciência da computação para as várias áreas da atividade humana, e acaba por não demonstrar como esse conceito deve se realizar no plano educacional.

Essa indefinição tem tornado difícil o estabelecimento de formas mensuráveis de avaliar o progresso dos alunos e tem redundado na inviabilização do desenvolvimento de um currículo, e consequentemente, na oferta de treinamento de professores.

Na tentativa de preencher essa lacuna, um número significativo de conferências e congressos tem sido realizado ao redor do mundo, em especial nos Estados Unidos e na Europa, financiados em boa parte por secretarias de educação e grandes empresas de tecnologia, como a Microsoft. Ao mesmo tempo, desde da publicação de [Wing \(2006\)](#), uma quantidade expressiva de artigos reportando experimentações em sala de aula tem sido publicados.

O ponto de convergência desses debates tem sido a busca de uma definição clara sobre o significado desse conceito para o contexto da sala de aula que ao mesmo tempo dialogue com as particularidades e dificuldades pertinentes a esse ambiente.

Dentre as questões para as quais se tem buscado resposta incluem-se ([WEINTROP et al., 2016](#)):

1. Como o pensamento computacional se relaciona com as atuais disciplinas e práticas correntes em sala de aula?
2. Como se estabeleceria a progressão de assuntos a serem trabalhados?

3. Quais são as práticas educacionais refletidas pelo pensamento computacional, e como preparar professores para que possam empregá-las em sala de aula?
4. Qual é melhor forma de avaliar o uso e o exercício dessas práticas pelos alunos?

Alguns exemplos concretos da formulação desses e de outros questionamentos podem ser encontrados no relatório produzido pelo Conselho Nacional de Pesquisas dos Estados Unidos (NRC), resultante de um encontro realizado no ano de 2010. Nele são listados vinte habilidades e práticas identificadas com o pensamento computacional, tais como a abstração e decomposição de problemas, formulação de soluções heurísticas e de estratégias de busca, bem como conceitos pertinentes à ciência da computação, tais como processamento paralelo e uso de recursão ([NATIONAL RESEARCH COUNCIL, 2010 apud WEINTROP et al., 2016](#)).

Também vale registrar o trabalho de [Barr e Stephenson \(2011\)](#), no qual o esforço de significar o pensamento computacional na perspectiva da sala de aula traduziu-se no mapeamento de algumas competências a disciplinas comuns à educação básica. Esse conteúdo está disponível nas tabelas [1](#) e [2](#).

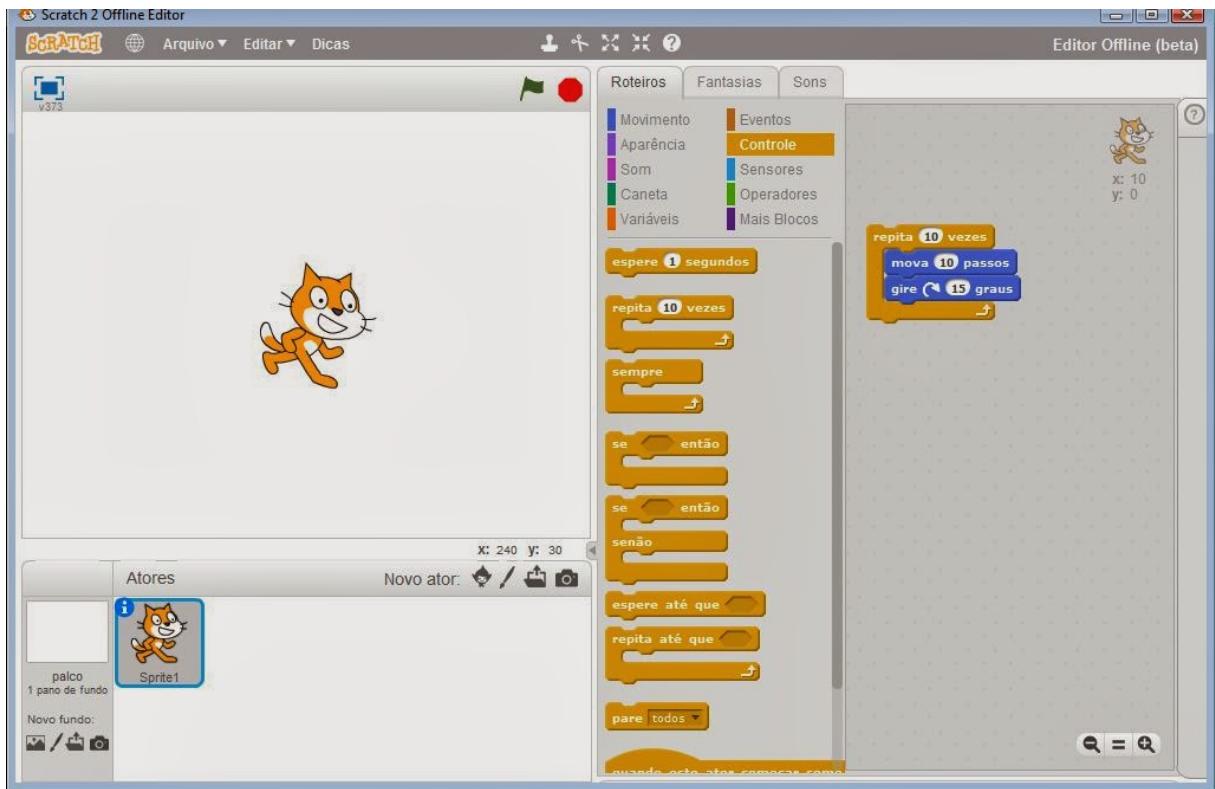
Outro trabalho merecedor de registro é o de [Brennan e Resnick \(2012\)](#). Nele o pensamento computacional é trabalhado sob a perspectiva da aprendizagem de programação e é estruturado em três dimensões:

1. **conceitos computacionais** – conceitos com os quais os estudantes se envolvem ao aprender a programar, tais como iteração e paralelismo.
2. **práticas computacionais** – práticas que os estudantes desenvolvem quando se engajam com esses conceitos.
3. **perspectivas computacionais** – perspectivas que os estudantes adquirem sobre o mundo ao seu redor e sobre si mesmo, resultantes desse aprendizado.

Acompanhando essa proposta, são sugeridas formas de avaliar o progresso dos alunos em cada uma delas. O uso do ambiente *Scratch*, ilustrado na figura [14](#), desenvolvido pelo *MIT Labs*, marca o desenvolvimento do artigo.

Ao longo das páginas seguintes, iremos apoiar as nossas análises na proposição de [Weintrop et al. \(2016\)](#). Entre os trabalhos analisados, esta contribuição é a que melhor responde à nossa necessidade de oferecer uma perspectiva prática do pensamento computacional, coerente com as necessidades do ensino de ciências. Isso porque nele podemos observar a delimitação clara das competências a serem exercitadas pelos alunos e como elas tendem a favorecer a tão almejada aprendizagem científica.

Figura 14 – *Scratch* – linguagem de programação visual desenvolvido pelo *MIT Labs*.



Essa definição é alcançada pela criação de uma taxonomia que conte com as principais práticas identificadas pelo autor que estão relacionadas ao pensamento computacional. Na seção seguinte iremos detalhar-la, mas, previamente, descrever a metodologia adotada. E como definições só podem ser úteis se forem acompanhadas de exemplos práticos, iremos, em seguida, demonstrar como o exercício dessas competências podem ser incorporadas a temas das aulas de ciências, ao mesmo tempo que ilustramos a discussão com a descrição de casos reais.

3.2 A criação de uma taxonomia para o pensamento computacional

A definição de uma taxonomia segue na esteira dos empreendimentos cujo objetivo tem sido uma significação funcional para o pensamento computacional. Nas palavras de Weintrop et al. (2016), o que se busca é

[...] uma abordagem diferente para a definição do pensamento computacional, contando com a aplicação das práticas identificadas acima em contextos distintos da ciência da computação. Ao fazer isso, deixamos de confiar em ideias e práticas descontextualizadas e, em vez disso, recorremos a instâncias reais do pensamento computacional na natureza para fornecer clareza e especificidade sobre o que o termo significa. Isso reforça o argumento de que essas práticas são amplamente aplicáveis, ao

mesmo tempo em que fornecem uma definição concreta e açãoável do pensamento computacional. (WEINTROP et al., 2016, Tradução nossa)

A metodologia adotada envolveu o emprego de três recursos básicos:

1. amostras de atividades educativas envolvendo pensamento computacional tanto em matemáticas como em ciências.
2. levantamento de conceitos existentes sobre o pensamento computacional presentes na literatura.
3. entrevistas com matemáticos e cientistas.

Os passos dados pelos autores foram assim sequenciados:

- 1. Delimitação das rotinas e habilidades relacionadas**

Com o objetivo de identificar as competências e práticas que são associadas repetidamente ao pensamento computacional, uma revisão da literatura foi feita. Dentre o material analisado estiveram tanto documentações oficiais produzidas sobre pensamento computacional, como o já citado National Research Council (2010), comoementas curriculares de programas de engenharia e ciência da computação. Durante essa investigação, buscou-se a aquisição de uma visão panorâmica sobre o tema antes da delimitação do escopo da análise para as disciplinas matemáticas e científicas. Dessa etapa, dez competências básicas foram extraídas (Tabela 3).

- 2. Coleção e classificação de atividades voltadas para a introdução do tema nas aulas de ciências e matemática**

Foram analisadas trinta planos de aulas de uma coleção, permeadas por conteúdos de física, astronomia, química, biologia, ciências da terra, redes e programação. Todo material analisado tinha origem no programa “Reach for Stars”² cujo propósito é a integração entre estudantes de pós-graduação de disciplinas STEM, dependentes do uso de computação nas suas pesquisas, e alunos da educação básica. Tendo como meta a identificação de práticas relacionadas às que foram recortadas do primeiro passo, uma análise foi feita por dois pesquisadores independentes. Dela emergiram um conjunto de 208 facetas do pensamento computacional, das quais uma pequena amostra é exposta na tabela 4.

² Mais informações sobre o programa pode ser encontrado em <http://gk12.ciera.northwestern.edu/>,

Tabela 1 – Práticas identificadas no pensamento computacional, segundo Barr e Stephenson (2011), presentes/possíveis na educação básica

Práticas	Matemática	Ciência	Estudos Sociais	Linguagens
<i>Coleta de dados</i>	Encontrar fonte de dados de um problema, lançando dados ou moedas, por exemplo	Recolhimento de dados de um experimento	Estudar estatísticas de batalha ou outras estatísticas populacionais	Fazer uma análise linguística de sentenças.
<i>Análise de dados</i>	Contagem da ocorrências de lançamentos de moeda ou dados e analisar resultados	Analizar os dados a partir de um experimentar	Identificar tendências nos dados a partir de estatísticas	Identificar padrões para frase de diferentes tipos.
<i>Representação de dados</i>	Use histograma, gráfico circular, gráfico de barras... Usar conjuntos, listas, gráficos, etc.	Resumir dados de um experimento	Resumir e representam tendências	Representar padrões de diferentes tipos de sentença
<i>Decomposição de problemas</i>	Aplicar ordem de operações em uma expressão	Fazer classificação de espécies	-	Escrever um esboço
<i>Abstração</i>	User variáveis em álgebra; estudar funções algébricas em comparação com as funções na programação	Construir um modelo de uma entidade física	Resumir os fatos; deduzir conclusões a partir de factos	-

Fonte: Adaptado de Barr e Stephenson (2011)

Tabela 2 – *Continuação - Práticas identificadas no pensamento computacional, segundo Barr e Stephenson (2011)*, presentes/possíveis na educação básica

Práticas	Matemática	Ciências	Estudos Sociais	Linguagens
<i>Procedimentos algorítmicos</i>	Longa divisão, fatoração	Fazer procedimentos experimentais	-	Escrever instruções
<i>Automação</i>	-	Usar <i>Probware</i> ou <i>Origin</i>	Usar excel	Usar um corretor ortográfico
<i>Paralelização</i>	Resolver sistema lineares; fazer multiplicação de matrizes	Executar simultaneamente experimentos com diferentes parâmetros	-	-
<i>Simulação</i>	Representar graficamente uma função em um plano cartesiano e modificar os valores da variáveis	Simular o movimento do sistema solar	Jogar Age of Empires; Trilha de Oregon	Fazer uma reencenação de uma história

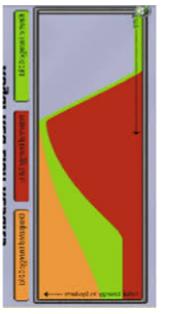
Fonte: Adaptado de Barr e Stephenson (2011)

Tabela 3 – Competências associadas ao pensamento computacional segundo análise da literatura feita por [Weintrop et al. \(2016\)](#)

- | | |
|---|--|
| 1. Capacidade para lidar com problemas abertos | 6. Criar abstrações para aspectos do problema em mãos |
| 3. Persistência para trabalhar com problemas desafiadores | 7. Reestruturar um problema em termos de outros conhecidos |
| 3. Confiança para lidar com a complexidade | 8. Avaliar pontos fortes/fracos de uma representação de dados/sistema representacional |
| 3. Representar ideas the forma computacionalmente significativa | 9. Gerar soluções algorítmicas |
| 3. Quebrar problemas grandes em problemas menores | 10. Reconhecer e lidar com a ambiguidade em algoritmos |
-

Fonte: [Weintrop et al. \(2016\)](#)

Tabela 4 – Facetas do pensamento computacional extraídos de material voltado para implementação do pensamento educacional nas escolas

Atividade	Faceta	Prática Envolvida	Classificação taxonômica da prática
Projeto de construtor de montanha-russa	<p>Os alunos constroem modelos de montanhas-russas que podem ser executadas, gerando dados sobre a energia da montanha-russa</p> 	<p>Obter insight / compreensão de simulações / modelos baseados em computador</p> <p>Construção de modelos computacionais.</p>	<p>Uso de um modelo computacional para entender um conceito</p> <p>Construção de modelos computacionais.</p>
	<p>Os alunos registram medições de energia em quatro pontos da pista e armazenam os dados em uma tabela</p>	<p>Fazer medições efetivas a partir de uma simulação</p>	<p>Coleta de dados</p>
Desenho de uma montanha-russa	<p>São necessárias várias iterações para construir uma montanha russa de sucesso que termine a pista e não trave</p> 	<p>Abordagem iterativa para a construção de uma solução</p>	<p>Uso de modelos computacionais para encontrar e testar uma solução</p> <p>Investigação sistemática das causas de um problema (<i>troubleshooting</i>) e depuração (<i>debugging</i>)</p>
Gráfico das energias cinéticas e potenciais da montanha-russa ao longo do tempo	<p>Traduzindo em gráficos de tela de energia potencial / cinética em forma de tabela</p>	<p>Mensurar a adequação de uma representação avaliando seus pontos fortes e fracos.</p>	<p>Manipulação de dados</p> <p>Visualização de dados</p>

Fonte: Adaptado de Weintrop et al. (2016)

3. Organização temática

A partir da análise feita até então, foi produzida uma lista com vinte e sete práticas, tematicamente organizada em cinco grandes campos: dados e informações, modelagem e simulação, resolução de problemas, pensamento sistêmico.

4. Validação

A taxonomia criada, disponível até a execução dessa etapa, foi apresentada a professores da educação básica. Com esse material em mãos, colaborativamente, eles puderam desenhar atividades para suas turmas.

No geral, de acordo com o autor, os *feedbacks* fornecidos por eles foram bastante positivos, embora algumas preocupações tenham sido evidenciadas em relação às seções da taxonomia com conceitos computacionais, tais como aplicação de lógica condicional e uso de recursão e iteração.

A razão para esse receio era a percepção que tais tópicos estavam muito distantes do conteúdo trabalhado em sala de aula.

Além de profissionais da educação básica, outros tantos *feedbacks* foram colhidos com pesquisadores de alguns setores da indústria, bioquímicos, físicos, engenheiro de materiais, astrofísicos, cientistas da computação e engenheiros biomédicos.

Além da intenção de colher avaliações, o objetivo dessa rodada de entrevistas foi a obtenção de informações e *insights* sobre o como o pensamento computacional integra autênticos ambientes de pesquisas. Ao mesmo tempo, buscou-se observar as semelhanças e diferenças na forma como o uso computação se efetiva na rotina de trabalho em cada uma das áreas desses profissionais.

5. Revisão

A partir dos *feedbacks* e sugestões feita pelos professores e pesquisadores, uma revisão da primeira versão da taxonomia foi feita.

O resultado final desse processo pode ser visto na tabela disponível na figura 15. Uma discussão sobre cada um dos elementos dessa classificação é feita a seguir.

Figura 15 – Taxonomia para práticas do pensamento computacional

Práticas relacionada a dados	Práticas de simulação e modelagem	Práticas de resolução de problemas com o uso de um computador	Práticas de pensamento sistêmico
Coleta de dados	Uso de modelos computacionais para entender um conceito	Preparo de problemas para soluções computacionais	Investigação de um sistema complexo como uma unidade, um todo
Geração de dados	Uso de modelos computacionais para encontrar e testar uma solução	Programação	Entendimento das relações dentro de um sistema
Manipulação de dados	Avaliação de modelos computacionais	Escolha de ferramentas computacionais apropriadas	Escolha de ferramentas computacionais apropriadas
Análise de dados	Desenolvimento de soluções computacionais modulares	Avaliação de diversas abordagens/soluções para um problema	Pensamento em níveis, camadas
Visualização de dados	Construção de modelos computacionais	Criação de abstrações computacionais	Comunicação de informação sobre um sistema
		Investigação sistemática das causas de um problema ou comportamento inesperado (<i>troubleshooting ou debugging</i>)	Definir sistemas e administrar complexidades

Fonte: Weintrop et al. (2016)

3.2.1 Práticas relacionadas a dados

Weintrop et al. (2016) sustenta que a capacidade de analisar e extrair significado de um grande volume de dados desestruturados tem sido uma das características definidoras da prática científica moderna. A maneira como esse empreendimento vem sendo conduzido tem sido impactados significativamente pelos avanços tecnológicos em curso.

Para ilustrar a relevância dessa prática, o autor destaca uma das entrevistas feitas durante a elaboração da taxonomia na qual um cientista de materiais, questionado sobre como era sua pesquisa, responde: “Em quase tudo, o que se tem são dados brutos”. Em seguida ele prossegue explicando como ele define as perguntas a serem respondidas (problemas de pesquisa), e como dados são gerados, processados e organizados, para que então, finalmente, o uso de determinados softwares gerem as visualizações para suas conclusões.

Segundo o autor, abrigar algumas dessas práticas nas atividades escolares deve significar ajudar os alunos a perceberem que inherentemente dados não oferecem respostas imediatas, e que para alcançá-las “ordem deve ser imposta”. Por esse motivo a introdução de noções básicas de estatística e probabilidade se faz necessária.

Cada um dos aspectos relevantes ao uso de dados são desctrinchados a seguir.

1. Coleta de dados

O papel que os computadores cumprem na coleta de dados guarda relação com a necessidade de automatizar protocolos de registro e armazenamento de dados gerados pela observação de um fenômeno.

De acordo com Weintrop et al. (2016), o aluno que dominar esta prática será capaz de propor protocolos de coleta de dados e formas de uso de ferramentas computacionais que automatizam essa tarefa.

2. Geração de dados

Em astronomia, para a compreensão da evolução de uma galáxia, torna-se necessário a geração de dados a partir de simulações computacionais, dado que escala de tempo investigada (bilhões de anos) é incompatível com a dos pesquisadores.

Esse exemplo ilustra um dos aspectos da pesquisa científica onde a necessidade de geração de dados impõe-se em razão de restrições circunstanciais.

Como assinala o autor, o estudante que dominar esta prática será capaz de definir procedimentos computacionais e executar simulações que permitam a criação de dados, cuja utilização pode auxiliá-lo a aprofundar a compreensão em um tópico sob investigação.

3. Manipulação de dados

O uso de computadores torna possíveis a manipulação de dados desestruturados, de tal modo a permitir que significados possam ser extraídos.

Dentre as várias técnicas existentes de manipulação, o autor cita o ordenamento, a filtragem, a limpeza, a normalização e a união de conjunto de dados dissociados.

Como destaca, o aluno que dominar esta prática será capaz de, a partir do uso de ferramentas computacionais, remodelar um conjunto de dados de tal modo a facilitar seu trabalho de investigação.

4. Análise de dados

Em face da abundância de dados existentes, o uso de ferramentas computacionais para análise de dados tem sido um das tarefas mais relevantes da prática científica.

Dentre as várias estratégias existentes, o autor destaca a busca por padrões ou anomalias, a definição de regras para categorização de dados, e a identificação de tendências e correlações.

Conforme sublinha, o aluno que dominar esta prática será capaz de fazer afirmações e extrair conclusões a partir da análise de um conjunto de dados.

5. Visualização de dados

Outros aspectos relevantes e necessários da atividade científica são a comunicação e o compartilhamento de dados.

Nesse sentido, o uso de ferramentas computacionais que tornam possível a projeção de dados, seja na forma de um gráfico simples ou de uma exposição dinâmica que permite a interação do usuário com a informação, desempenha papel preponderante.

Conforme sustenta [Weintrop et al. \(2016\)](#), o aluno que dominar esta prática será capaz de usar com eficiência ferramentas de visualização que permitam a comunicação de informações e conclusões extraídas durante a análise.

3.2.2 Práticas de simulação e modelagem

Outro conjunto práticas que compõe a taxonomia de [Weintrop et al. \(2016\)](#) são aquelas relacionadas ao exercício da representação de fenômenos, também presentes no uso e criação de modelos e simulacros computacionais.

Falando especificamente de modelos, poderíamos dizer que “todos estão errados, mas alguns deles são úteis” ([BOX; DRAPER, 1987 apud WEINTROP et al., 2016](#)), dado o caráter simplificador, mas representativo de alguns – proporcional à capacidade descritiva e preditiva que comportam.

Dentre as várias formas que um modelo pode assumir, o autor destaca os fluxogramas, os diagramas, as equações, as fórmulas químicas, os modelos físicos (como maquetes

ou representações tridimensionais de uma molécula), e por fim os modelos e simulações computacionais – representações não-estáticas de um fenômeno, reproduzidos por um computador ([WEINTROP et al., 2016](#)).

Figura 16 – Modelo físico de uma cadeia de DNA



O trabalho de criação de um modelo computacional envolve a identificação de abstrações apropriadas – regras matemáticas subjacentes ao fenômeno modelado, por exemplo – e seu refino iterativo, por meio de depuração (investigação e correção de comportamentos inesperados) e validação com elementos correspondentes no mundo real ([WING, 2008](#)).

De um ponto de vista pedagógico, a sua utilidade está na possibilidade de tornar a compreensão de um fenômeno mais clara e permitir a investigação de comportamentos cuja condições determinantes são difíceis, senão impossíveis de reproduzir.

O poder didático dessa ferramenta, contudo, não está apenas no seu uso, mas na possibilidade do estudante também criá-la, tornando possível a formulação de suas próprias interpretações da realidade.

Como ressalta [Weintrop et al. \(2016\)](#), os educadores reconhecem que tais práticas

de representação são indissociáveis da aprendizagem científica, mas elas são raramente explicitadas como parte da instrução. E é exatamente para contornar esse problema que novas formulações curriculares tem tornado claro a necessidade de estimular a criação, e não apenas o uso de modelos acabados.

Abaixo listamos as cinco práticas incluídas na taxonomia, referentes à simulação e modelagem.

1. Uso de modelos computacionais para entender um conceito

Segundo o autor, o uso de modelos computacionais que demonstram ideias específicas ou simulam comportamento de fenômenos são poderosas ferramentas didáticas.

Isso porque eles tendem a oferecer suporte a investigações sistemáticas que permitem o exercício do controle sobre parâmetros de comportamento, através de combinações e variações que, normalmente, são difíceis, senão impossíveis, de reproduzir sem o aparato computacional.

Por isso, conclui que o estudante que dominar essa prática será capaz de aprofundar sua compreensão sobre um fenômeno ou conceito sob investigação, utilizando modelos computacionais que os reproduzem.

2. Uso de modelos computacionais para encontrar e testar uma solução

Modelos computacionais podem ser usados para testar hipóteses e descobrir soluções para problemas.

Como discutido anteriormente, eles viabilizam, por exemplo, a exploração de aspectos de fenômenos que seriam muito perigosos, caros, difíceis, ou simplesmente impossíveis de reproduzir em escala ordinária de tempo e de recursos.

Entretanto, com poder computacional e modelos matemáticos adequados pode-se facilmente alcançar uma boa diversificação e combinação do “espaços de parâmetros”³, e assim validar ou testar os limites de uma solução.

Weintrop et al. (2016), sugere que o aluno que dominar esta prática será capaz de propor, testar e assim justificar o uso de uma solução, a partir da exploração de um modelo computacional, e aplicar adequadamente as informações extraídas nesse processo.

3. Avaliação de modelos computacionais

A capacidade de avaliar a relação de um modelo computacional com a realidade representada é um dos aspectos essenciais para medirmos a sua adequabilidade, segundo Weintrop et al. (2016).

³ conjunto de todos os arranjos possíveis dos parâmetros de um modelo matemático

O autor aponta algumas questões que precisam ser analisadas e respondidas quando se tem esse fim em mente. Dentre elas estão as seguintes:

- a) Quais suposições que os criadores do modelo fizeram sobre o mundo e como elas afetam o seu comportamento?
- b) Quais as camadas de abstração foram incorporadas ao modelo e como elas moldam a fidelidade da sua representação do fenômeno?
- c) Quais são os aspectos do modelo que foram fielmente modelados e quais outros foram simplificados ou simplesmente ignorados?

Responder essas questões é tarefa essencial para compreensão dos limites de uma representação. Ao mesmo tempo, tê-las em contas é fundamental para que se possa validá-la ou não.

De acordo com o autor, o aluno que dominar esta prática “será capaz de articular as diferenças entre o modelo e fenômeno representado, o que inclui o levantamento de ameças a sua validade e a identificação das suposições que o sustentam” ([WEINTROP et al., 2016](#)).

4. Desenho de modelos computacionais

O desenho de modelos computacionais podem ser norteados por algumas motivações. Dentre as já discutidas, podemos citar a necessidade de testar hipóteses ou comunicar ideias, e o interesse em compreender melhor um fenômeno,

Durante esse processo somos levados à tomada de algumas decisões, que podem variar de acordo com o problema a ser resolvido. Dentre as mais recorrentes, [Weintrop et al. \(2016\)](#) cita a necessidade de definir os limites do sistema – explicitar aquilo que deve ser incorporado e/ou descartado – e conceitualizar as propriedades e comportamentos dos elementos incluídos. Ao final, deve ser assegurado que o produto resultante atenda aos objetivos que inicialmente motivaram a construção do modelo ([WEINTROP et al., 2016](#)).

Portanto, os estudantes que exercitarem essa prática terão a oportunidade de desenhar modelos computacionais, descrevendo-os e decidindo quais dados serão produzidos, enquanto poderão cultivar o hábito de fazer avaliações críticas, a partir da análise dos limites da representação, das suposições nela incorporadas. Espera-se que essa reflexão faculte a eles o entendimento da extensão das conclusões que podem ser extraídas do modelo.

5. Construção de modelos computacionais

Uma importante ferramenta da atividade de um pesquisador é sua capacidade de criar modelos computacionais ou ampliar outros já existentes, seja dando continuidade ou simplesmente complementando o trabalho realizado por seus pares.

Enquanto o desenho de um modelo, como já discutido, envolve a tomada de decisões essencialmente teóricas – tais como a seleção e descarte de elementos, bem como a descrição e conceptualização de comportamentos e propriedades – a sua construção, por outro lado, exige saber expressá-lo de forma “compreensível” a um computador.

Em muitos casos essa competência se expressa com o uso de uma linguagem de programação, mas em outros com *softwares* que provêm uma interface gráfica ou de linha de comando, tais como o *Wolfram Mathematica*⁴ e o *Matlab*⁵.

Weintrop et al. (2016), sugere que estudantes que dominarem essa prática serão capazes de implementar novos comportamento de modelos, seja por criação ou extensão, utilizando tanto uma linguagem de programação quanto *softwares* prontos, que oferecem suporte à modelagem.

3.2.3 Prática de resolução de problemas computacionais

Nesta etapa de elaboração da taxonomia Weintrop et al. (2016), concentra seus esforços na apresentação das práticas de resolução de problemas trazidas pela ciência da computação. Seguem destrinchadas cada uma delas.

1. Preparo de problemas para admitir soluções computacionais

Normalmente, para que um problema admita solução computacional, faz-se necessário a sua reformulação de modo a mapeá-lo às capacidades de ferramentas existentes – sejam elas pacotes de software ou dispositivos físicos.

Dentre as várias estratégias existentes que possibilitam esse reenquadramento, Weintrop et al. (2016) lista a possibilidade de decomposição de um problema em unidades menores, ou simplesmente a sua reedição em termos de outros, cujas soluções são conhecidas.

O autor sugere que o estudante que dominar esta prática será capaz de empregar estratégias de reenquadramento e tornar mais eficiente a resolução de problemas nos quais estiver trabalhando.

2. Programação

O emprego da codificação tem se mostrado um poderosa ferramenta na resolução de problemas. Dentre as vários possibilidades que essa habilidade tem trazido para pesquisa científica, pode-se destacar a coleta e análise massiva de dados, a formatação e visualização de informações, bem como a construção e extensão de modelos e interfaces com ferramentas computacionais existentes (WEINTROP et al., 2016).

⁴ <http://www.wolfram.com/mathematica/>

⁵ <https://www.mathworks.com/products/matlab.html>

O ensino de programação, contudo, tem se mostrado um tarefa desafiadora em face da pouca ou nenhuma disponibilidade de programas de capacitação para professores e da dificuldade de aprendizado relacionada à sintaxe das linguagens de programação convencionais.

Para contornar estas dificuldades, tem sido desenvolvidas linguagens e plataformas de programação visual, cujo arranjo algorítmico e sintático das instruções dadas são estruturados por elementos gráficos e não textuais. Dos exemplos com esta finalidade, o mais proeminente é o *Scratch*, sobre o qual já falamos na seção 3.1. Mais detalhes a respeito e registros de experiências didáticas envolvendo o programa podem ser encontradas em [Maloney et al. \(2008\)](#) e [Resnick \(2012\)](#).

Figura 17 – Scratch – linguagem e plataforma de programação visual



Quanto às competências a serem exercitadas pelo estudante com o aprendizado de programação, de acordo com a taxonomia de [Weintrop et al. \(2016\)](#), espera-se que eles se tornem capazes de entender, modificar e criar programas e, com essas habilidades, persigam seus “próprios objetivos matemáticos e científicos”.

3. Escolha de ferramentas computacionais apropriadas

Um único problema pode ser resolvido de diversas formas e com o auxílio de diversas ferramentas. A destreza para identificar dentre um conjunto aquelas que melhor auxiliam nesse propósito é uma das principais competências necessárias em um projeto.

Na lista parâmetros que normalmente influenciam essa escolha incluem-se as funcionalidades oferecidas pelo software, as suas possibilidades de personalização, os tipos de dados que podem ser inseridos e produzidos, e para além do campo técnico, por exemplo, a maturidade e grau de atividade da sua comunidade de usuários (WEINTROP et al., 2016). Esse espaço de troca de informação é especialmente importante em situações em que a assistência se faz necessária mas a documentação disponível pelos desenvolvedores da ferramenta são insuficientes.

Weintrop et al. (2016) sugere que os alunos que adquirem essa *expertise* serão capazes de balancear as vantagens e desvantagens oferecidas por ferramenta computacional e, assim, tomar decisões justificáveis e informadas.

4. Avaliação de diversas abordagens/soluções para resolver um problema

Uma outra faceta do processo de resolução de problemas, recorrente na prática científica, é necessidade de escolher uma solução dentre muitas possíveis.

Quando há duas ou mais abordagens para um problema, faz-se necessário a avaliação da adequação de cada uma delas em termos de parâmetros tais como extensibilidade, durabilidade, flexibilidade, reusabilidade e custo, seja em recursos materiais ou em tempo.

Weintrop et al. (2016) avalia que alunos que exercitarem essa prática poderão avaliar diversas abordagens ou soluções para um problema, tendo como referência de julgamento as restrições e requisitos do problema, bem como as funcionalidades das ferramentas disponíveis.

5. Desenvolvimento de soluções computacionais modulares

Ordinariamente, a resolução de um problema complexo sequencia-se em um conjunto de etapas menores e assume forma na solução de outros menores.

Cada uma dessas soluções podem ser desenhadas de modo intrincado, visando apenas o problema em questão, ou podem ser concebidas modularmente, de modo a serem reutilizáveis.

Quando modulares, os elementos de um modelo ou ferramenta computacional podem ser desenvolvidos de maneira incremental e serem testados e depurados de maneira independente, aspecto que facilita sensivelmente sua manutenção.

Quanto ao aspecto escolar, Weintrop et al. (2016) sustenta que os alunos que dominarem essa prática poderão desenvolver soluções cujos componentes atendem tanto as necessidades dos problemas em mãos quanto daqueles que, eventualmente, surgirem adiante.

6. Criação de abstrações computacionais

Como já discutido anteriormente, o trabalho de criação de uma abstração traduz-se na conceitualização de terminadas ideias ou processos, por meio do qual explicita-se determinados aspectos em detrimento de outros, entendidos como menos relevantes para descrição.

Como também já visto, a construção e o emprego dessa entidade conceitual desempenha papel fundamental na atividade matemática e científica, e toma forma na codificação de programas, na geração e visualização de dados, na definição do escopo e escala de um problema, ou na criação de modelos, concebidos para exploração e/ou melhor compreensão de um fenômeno (WEINTROP et al., 2016).

Tratando-se especificamente de abstrações computacionais, Weintrop et al. (2016) sugere alunos que adquirirem a capacitação para criá-las estarão aptos, ao mesmo, para identificá-las e empregá-las em função dos seus próprios objetivos matemáticos e científicos.

7. *Troubleshooting e Debugging*

O uso da expressão *troubleshooting* aplica-se à investigação sistemática e à compreensão do porquê algo não está se comportando como esperado. Em ciência da computação, esse processo é também conhecido como *debugging* (depuração, em tradução livre).

Dentre as estratégias compreendidas pelo termo, incluem-se a definição clara do problema e o teste metódico do sistema sobre o qual recai o erro. Nos dois casos o que se objetiva é o seu isolamento e reprodução.

Nas disciplinas STEM, o estudo do tema desempenha papel preponderante, dado que nelas o surgimento de erros e comportamentos inesperados são uma constante. Contudo, a forma como que eles são depurados varia, sendo possível, portanto, encontrar em cada uma delas um conjunto de técnicas e ferramentas específicas desenhadas com esse propósito.

Weintrop et al. (2016) sugere que o exercício dessa prática tornará o aluno apto a identificar, isolar, reproduzir, e por fim, corrigir comportamentos e erros inesperados em problemas ou modelos com os quais estiver trabalhando.

3.2.4 Práticas de pensamento sistêmico

Múltiplos problemas com quais lidamos atualmente envolvem um conjunto numeroso de componentes que se inter-conectam e inter-relacionam.

A habilidade de concebê-los sob essa perspectiva configura o que se denomina como “pensamento sistêmico”, e se diferencia essencialmente das formas de análise tradicional, com as quais os comportamentos individuais das partes são tomadas como simples agregadores

do todo. Assim, surge a noção de que certas propriedades resultantes não podem ser explicadas sem que seja considerada a atividade integral do sistema.

No contexto de uma sociedade em que a disponibilidade de dados é abundante – onde expressões como *big data* ganham força – emergem métodos de análise suportados pela premissa de que sistemas devem ser estudados, investigados, e entendidos no seu conjunto, e não em função das suas partes.

Essa concepção não é essencialmente nova. Esse fato é exemplificado por alguns modelos e conceitos científicos, existentes há décadas, cuja formulação tem a perspectiva sistêmica como referencial. Alguns bons exemplos incluem a seleção e dinâmica populacional em ecologia, o sistema respiratório humano, e a lei dos gases ideais, originários da física e química (WEINTROP et al., 2016).

Weintrop et al. (2016) sinaliza que embora os aspectos do pensamento computacional apresentados até aqui por sua taxonomia não estejam diretamente associados a noção de pensamento sistêmico, ferramentas computacionais constituem um meio poderoso para tornar essas ideias acessíveis a seus aprendizes.

Adiante apresentamos como o autor estrutura essas ideias no corpo da sua taxonomia.

1. Investigação um sistema complexo como uma unidade

Um sistema pode ser visto como uma entidade compostas por elementos que se inter-relacionam, ou como uma unidade, um caixa-preta, cuja configuração externamente mensurável (*output*) depende do arranjo de condições ao qual é submetido (*input*).

Essa perspectiva é especialmente relevante no contexto da atividade científica, onde fenômenos resultantes de interações complexas em larga escala são sistematicamente analisados.

Weintrop et al. (2016) sugere que alunos que exercitarem essa perspectiva serão capazes de desenhar, conduzir investigações, e por fim interpretar dados coletado de um sistema, modelado como uma entidade única.

2. Entendimento das relações dentro de um sistema

O foco na observação do comportamento integral de um sistema complexo, tal como um fenômeno físico não-linear, enfatizado pela prática anterior, embora ofereça algumas respostas, não é suficiente para descrição e justificativa de algumas de suas propriedades, em muitos casos. Em outros, faz-se necessária a identificação dos diferentes elementos que o compõe, bem como a compreensão e a articulação das relações existentes entre eles.

Weintrop et al. (2016) sublinha que ferramentas computacionais são úteis na investigação e exploração dessas relações por facilitar o controle e o isolamento desses componentes.

Quanto às competências a serem desenvolvidas pelos alunos, o autor sustenta que o desenvolvimento dessas práticas em sala de aula permitirá a eles o exercício de identificação dos elementos constituintes de um sistema complexo, e de articulação das relações existentes entre eles.

3. Pensamento em camadas

A perspectiva de análise de um sistema pode transicionar entre níveis *micro* e *macro*. Um enquadramento cujo referencial são os componentes e atores constituintes permite o desenvolvimento de insights sobre como a interação entre eles favorece a formação de determinados comportamento e propriedades do conjunto.

Por outro lado, outros aspectos são apenas explicáveis se encaramos o sistema como uma caixa-preta, como um agregado.

Weintrop et al. (2016) defende que o estudante conseguir exercitar essa transição do olhar, aprenderá a identificar os níveis distintos dentro de um único sistema, e a articular comportamento de cada um deles com respeito ao todo.

4. Comunicação de informação sobre um sistema

Comunicar o que se aprende em uma investigação é essencial para o exercício da atividade científica.

Essa tarefa pode ser especialmente desafiadora quando o objecto de análise é um sistema complexo, constituído por quantidades expressivas de componentes que se inter-relacionam de modo intrincado.

Quase sempre, para tornar a informação a ser transmitida acessível, faz-se necessário a elaboração de visualizações, tais como infográficos e fluxogramas, que evidenciam os aspectos aprendidos mais relevantes.

Weintrop et al. (2016) lembra que essa habilidade se sustenta em um senso de prioridade que torna clara ao comunicador quais são os elementos proeminentes e quais que podem ser ignorados sem comprometer a integridade da informação.

O autor ainda ressalta que os alunos que tornarem-se experientes nessa prática serão capazes de comunicar o que aprenderam sobre um sistema de maneira a tornar essa informação acessível a outros.

5. Definir sistemas e administrar complexidade

Qualquer coisa pode ser visto como um sistema. Desde um grupo pequeno de funcionários de uma empresa familiar, até um volume de gás contendo bilhões de moléculas.

Essencialmente, o que definirá o seu grau de complexidade é o número absoluto e a diversificação de tipos de elementos presentes. Incluem-se também nessa equação a quantidade, a diversificação, e a frequência de interações existentes.

Weintrop et al. (2016) lembra que a tarefa da definição do contorno de um sistema é essencial pois é ela que determina o seu grau de complexidade, e, por consequência, os tipos de questões ou problemas de pesquisas que podem ser respondidos ou resolvidos.

O autor ainda ressalta que alunos que dominarem essa prática serão capazes de delinear os limites de um sistema de tal modo que eles possam utilizá-lo como um domínio de investigação para uma questão específica.

Concluída a etapa de apresentação da taxonomia, mostraremos na próxima seção alguns exemplos de como trazer o exercícios de algumas competências catalogadas para o “chão” da sala de aula.

3.3 Exemplos de atividades

3.3.1 Pássaros raivosos e a constante gravitacional

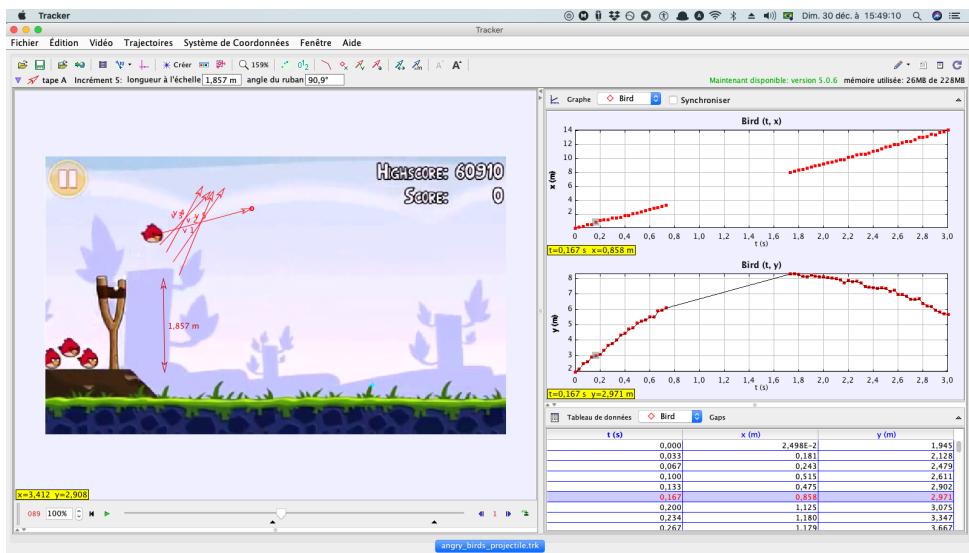
A atividade em questão consiste na gravação pelo aluno de alguns segundos (vinte a trinta) do jogo “Angry birds” (Fig. 18), no qual pássaros são lançados em trajetória parabólica.

Figura 18 – *Print screen* do jogo Angry Birds



O objetivo dessa atividade é a análise de algumas propriedades físicas do ambiente que o jogo simula, que poderá ser feita a partir dos dados gerados por softwares como o *Physics Tracker*⁶, úteis para o traqueamento de propriedades física em vídeos. A fig. 19 ilustra o seu ambiente gráfico.

Figura 19 – Ambiente gráfico do *Physics Tracker*, no qual analisamos alguns segundos da trajetória de um pássaro, personagem do jogo *Angry birds*



Dentre outras possibilidades, os dados registrados pelo *tracker* facultam ao aluno a descoberta das formas matemáticas funcionais das equações de movimento do personagem, o que torna viável, por exemplo, a inspeção do valor da constante gravitacional simulada no jogo. Esses achados, por sua vez, possibilitam uma avaliação do quanto representativo da realidade é o modelo.

A geração de dados pelo modelo físico simulado pelo jogo e a posterior coleta permitida pelo software contemplam duas práticas com os mesmos nomes, catalogadas pela taxonomia como “Prática relacionada a dados” (subseção 3.2.1).

Já o uso do *tracker* para descoberta dos valores de propriedades físicas e das equações de movimento, bem como a análise crítica do modelo em função do que é encontrado inserem-se na categoria “Prática de simulação e modelagem” (subseção 3.2.2), sendo a primeira prática classificada como “Uso de modelos computacionais para encontrar e testar uma solução” e a segunda como “Avaliação de modelos computacionais”.

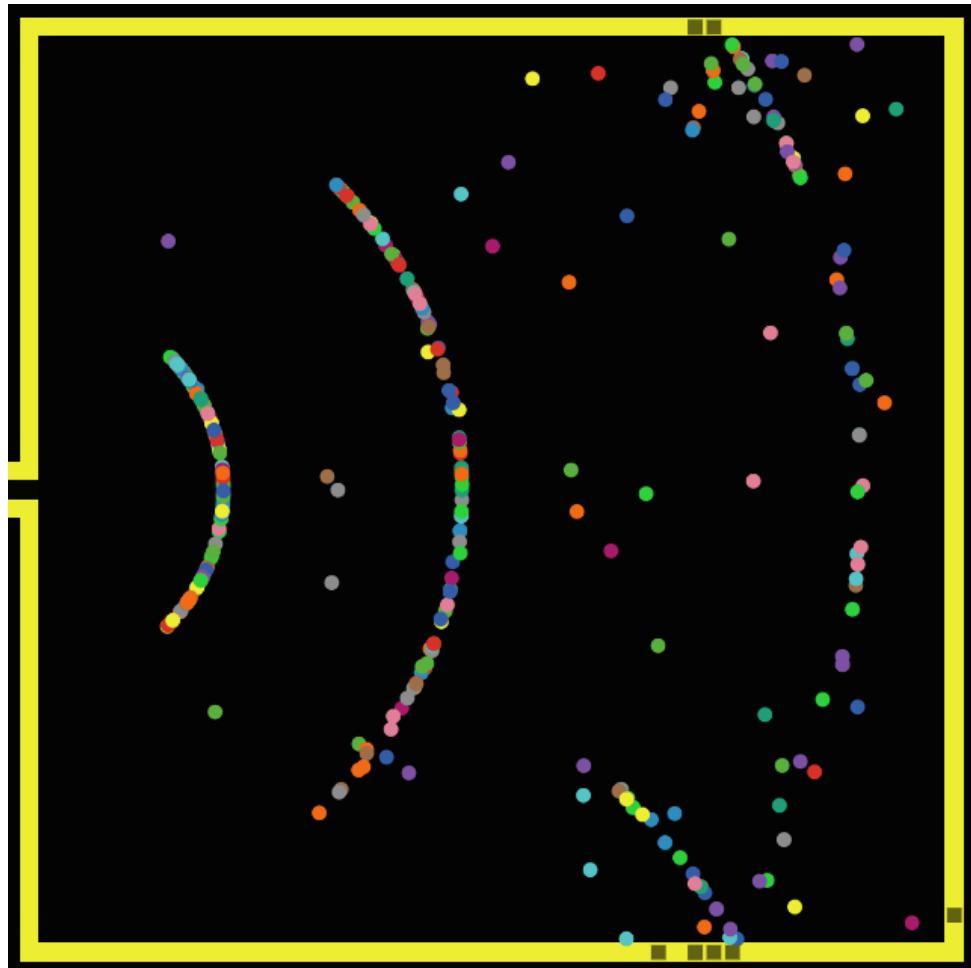
3.3.2 Modelagem do bombeamento em um pneu

O objetivo dessa atividade é a simulação computacional do comportamento das partículas no interior de um pneu bombeado com ar, modelado como um container com

⁶ Disponível gratuitamente para as principais sistemas operacionais em <https://physlets.org/tracker/>

paredes e volume fixos que encerra uma porção de gás ideal. Na figura 20 ilustramos o conjunto.

Figura 20 – Simulação de um pneu bombeado com ar, modelado como um gás ideal



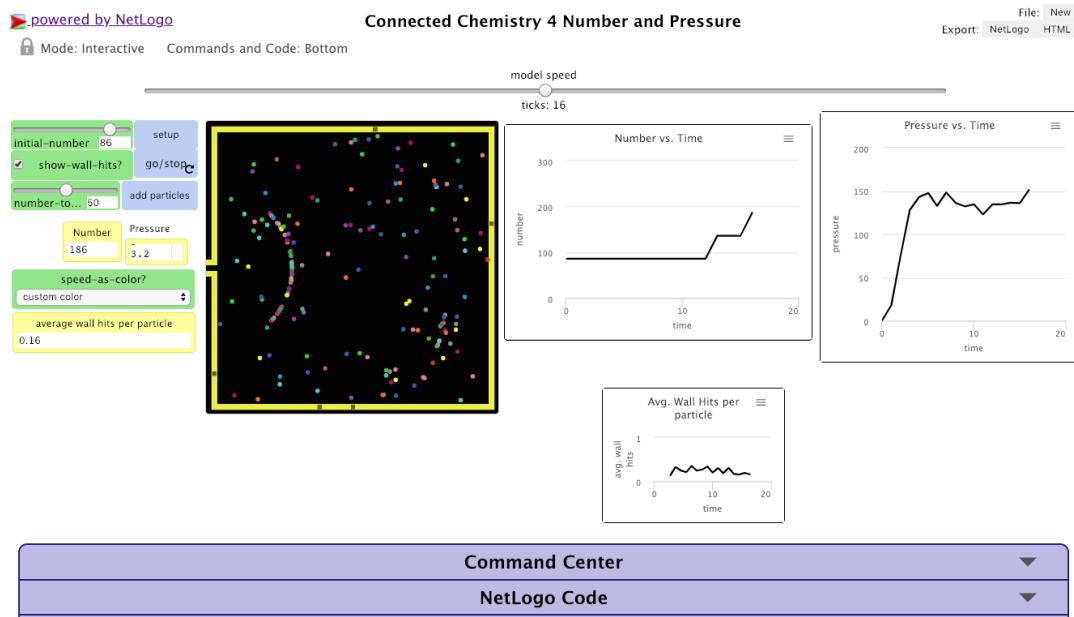
A condição de gás ideal exige que consideremos as colisões nas paredes como elásticas, e que desprezemos qualquer interação à distância (energia potencial) entre suas partículas, que portarão, portanto, apenas energia cinética.

Com esse modelo esperamos que os alunos possam estudar fatores que levam à variação da pressão no interior do container, bem como a relação entre essa grandeza com o número de partículas lá encerradas.

Como suporte à atividade pode-se utilizar um modelo programado com o auxílio do software NetLogo ([WILENSKY, 1999](#)) disponível para download por [Wilensky \(2004\)](#). Na figura 21, ilustramos o ambiente de exploração do modelo a ser utilizado nessa atividade.

Um dos elementos que tornam interessante esta atividade é o painel de controle com o qual pode-se variar os parâmetros do modelo, tais como o número inicial de partículas, a quantidade a ser adicionada e a velocidade de simulação. Outro aspecto

Figura 21 – Modelo de gás ideal e ambiente de suporte à sua exploração a ser utilizado nessa atividade



relevante é a exposição gráfica dos dados gerados, e a possibilidade de exportá-los para formatos compatíveis com softwares populares de tabelamento, como o *Excel*. Na figura 22, demonstramos como fazê-lo.

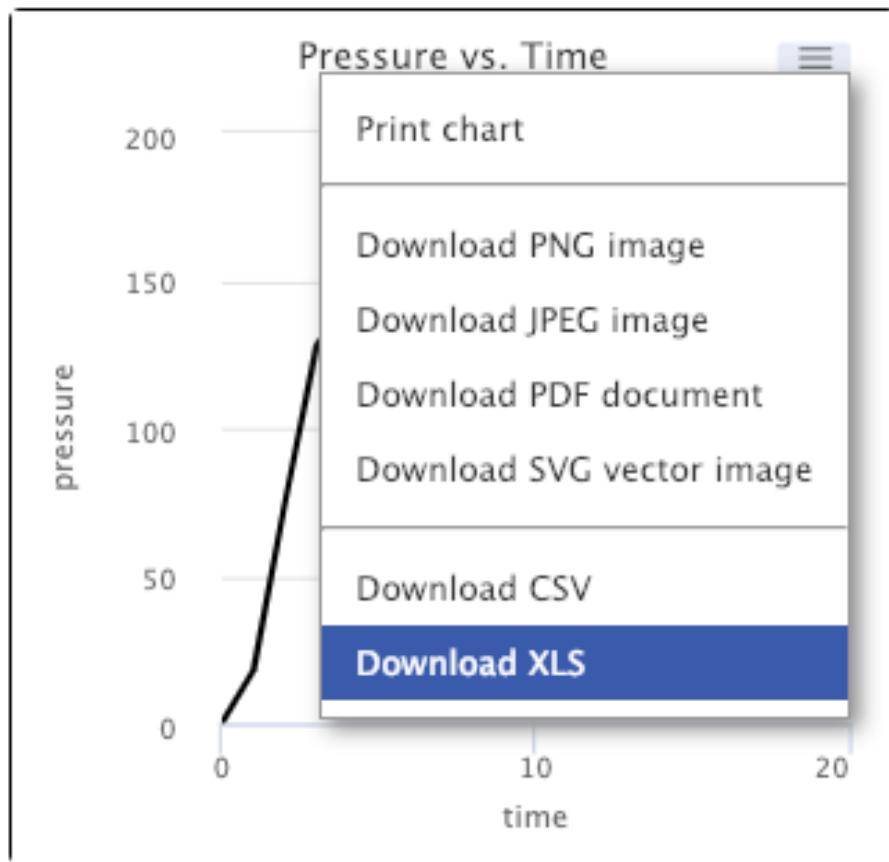
De porte dos dados para o “número de partículas *vs* tempo” e a “pressão *vs* tempo”, os alunos podem traçar um gráfico relacionando a pressão e o número de partículas contidos no container. O que se espera é que eles descubram, por conta própria, a relação linear entre essas duas grandezas.

Essa atividade é proposta por Novak (2016), e compõe um grupo de experimentos computacionais desenhados com a finalidade de tornar explorável a influência do ambiente externo nas propriedades físicas de um gás ideal e o como elas se relacionam matematicamente. Com essa investigação, espera-se que o aluno possa de modo autônomo derivar a equação dos gases ideais a partir dos dados experimentais, gerados pelas simulações.

Além disso, o autor acredita que os *insights* proporcionados por essas tarefas favorecem a formação de um “senso profundo e intuitivo” do comportamento de um sistema de partículas, facilitando, por exemplo, o desenvolvimento de previsões sobre outros comportamentos, mesmo estando ele em estado sólido ou líquido.

Quanto à taxonomia, podemos notar nesta atividade a possibilidade de exercício das seguintes competências:

Figura 22 – Exportação dos dados da simulação para formatos compatíveis com *Excel*



1. Práticas de simulação e modelagem (subseção 3.2.2)

- O uso de um modelo computacional para a compreensão de um conceito – nesse caso a noção de pressão.

2. Prática de dados (subseção 3.2.1)

- Geração de dados por um modelo computacional – exibidos nos gráficos “número de partículas vs tempo” e “pressão vs tempo” (fig. 21), cuja exportação pode ser feita conforme a figura 22.
- Análise e visualização gráfica de dados – necessárias para a exploração das relações entre o número de partículas e a pressão.

3. Práticas de pensamento sistêmico (subseção 3.2.4)

- Investigando o sistema como um todo – o uso do conceito de pressão ao longo de toda a atividade exige que o estudante enxergue o sistema de partículas na sua totalidade.

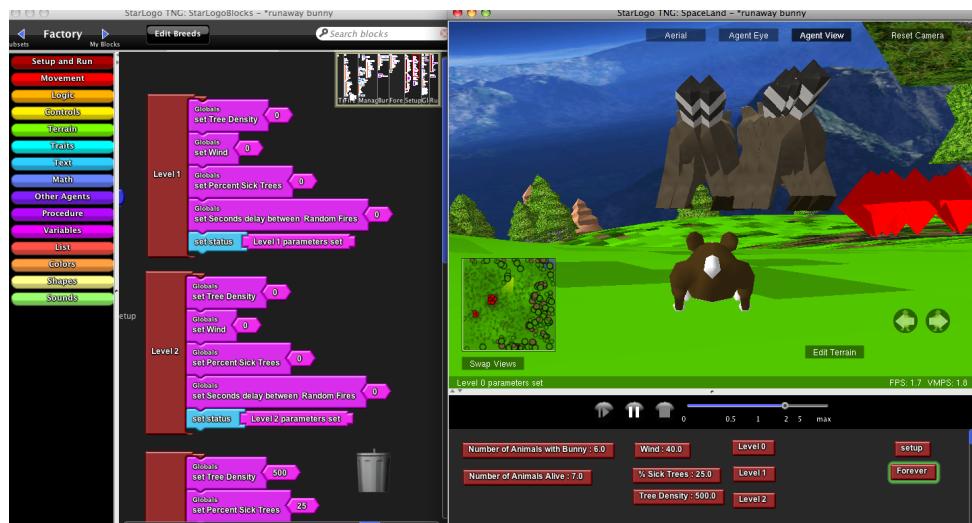
- b) *Pensamento em camadas* – necessário para estabelecer relação entre o comportamento individual das partículas (variação de momento decorrente da colisão com paredes) e emergência de propriedade coletiva (pressão).

3.3.3 Alunos faltosos como contexto de um modelo epidemiológico

A atividade em questão foi reportada por Lee et al. (2011), na qual um grupo de estudantes elaboram um modelo epidemiológico com o propósito de investigar se uma doença poderia se espalhar entre a população escolar em função do layout da construção, do número e movimento dos estudantes, da virulência da doença e do número de pessoas inicialmente infectados.

Para o mapeamento desses fatores no modelo foi utilizado um software de modelagem baseado em agentes (entidades ou atores com liberdade de movimento espacial, geralmente bidimensional, definida por regras previamente estabelecidas) chamado *StarLogo TNG*⁷ (fig. 24).

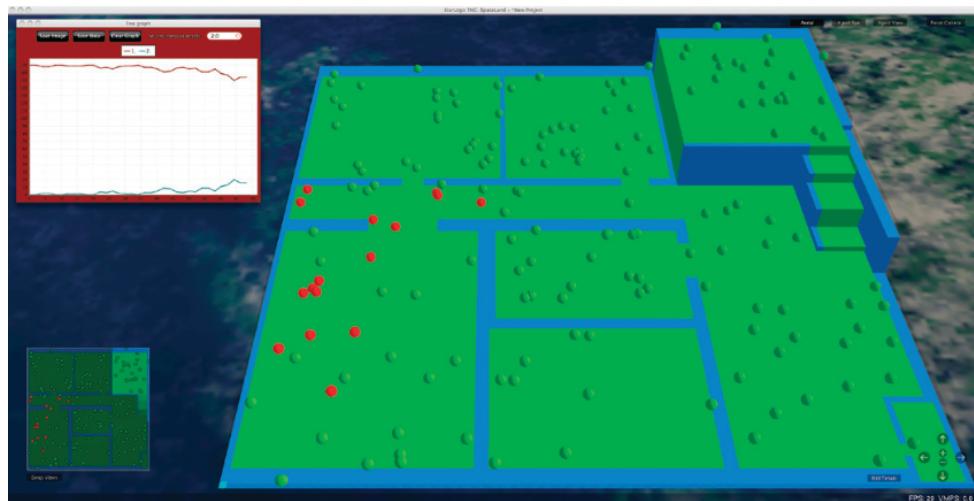
Figura 23 – Imagem do *software* de modelagem *StarLogo*, desenvolvido pelo *MIT Media Lab*



Durante a execução desta atividade, tais agentes foram tomados como uma abstração dos estudantes, e a quantidade deles inserida no modelo correspondeu à exatamente ao de matriculados na escola. A atribuição de movimento às entidades teve por finalidade a criação de uma representação simplificada do movimento dos alunos de uma sala à outra. Já com a delimitação espacial do ambiente, buscou-se a reprodução do espaço físico da instituição. A elaboração deste modelo está ilustrado na figura 24.

⁷ Disponível em <http://web.mit.edu/mitstep/starlogo-tng/download/index.html>

Figura 24 – Construção da representação do ambiente escolar no *StartLogo*. Na mesma imagem é possível também distinguir pontos vermelhos e verdes, representando, respectivamente, alunos doentes e saudáveis.



O controle de parâmetros do modelo foi feito pelo a inserção de *sliders*. Com eles foram possíveis, por exemplo, ajustar o número de alunos ou agentes inicialmente infectados, e controlar o grau de virulência da doença. Além do número inicial de agentes infectados, para a análise da velocidade de espalhamento foi tomada em consideração a frequência de contato entre os agentes.

Já a evolução da simulação foi posta em curso por um *loop* ou, em português, um laço infinito⁸ com o qual o software de modelagem atualizava o estado dos agentes, especificado pela variáveis localização e cor – indicando se o indivíduo está saudável ou doente.

Dada o comportamento randômico de cada agente, a possibilidade de automatizar o modelo, conforme ressalta Lee et al. (2011), permitiu que um conjunto de experimentos fossem feitos com as mesmas condições iniciais, possibilitando o acúmulo de amostras diferentes, e consequentemente a obtenção de probabilidades para certos resultados. Aos primeiros também seguiram as primeiras reavaliações pelos alunos do modelo criado, viabilizadas pela comparação com o número de faltas registrado pela direção da escola durante um surto de gripe suína.

Nesse contexto, entende-se por reavaliar a tarefa de analisar as suposições embutidas no modelo e como elas justificam, ou não, o comportamento do fenômeno analisado, expresso nos dados reais – neste caso, o número de alunos faltosos. Assim, pode-se fechar um primeiro ciclo, onde reflexões sobre a qualidade do modelo subsidiam o seu refinamento em rodadas seguintes.

⁸ Loops ou laços infinitos são construtos computacionais nos quais uma sequência de instruções são executadas infinitamente, ou pelo menos uma até que condição de parada seja satisfeita.

A breve descrição dessa atividade nos permite identificar o exercício das seguintes competências:

1. Prática de dados (subseção 3.2.1)

- a) *Visualização de dados* – competência materializada na diferenciação em cores dos agentes doentes e saudáveis.

2. Práticas de simulação e modelagem (subseção 3.2.2)

- a) *Uso de modelos computacionais para encontrar e testar uma solução* – a criação do modelo proposto pela atividade permite que os alunos proponham e teste nele fatores que influenciam no comportamento do fenômeno de transmissão de gripe.
- b) *Avaliação de modelos computacionais* – o exercício dessa competência se faz necessária na etapa da reavaliação, onde alunos comparam os dados gerados pela simulação e refletem sobre as suposições embutidas no modelo. Espera-se que essa atitude os ajude a refinar as suas hipóteses.
- c) *Desenho de modelos computacionais* – competência implicada na tomada de todas as decisões sobre o que incluir no modelo. Desde a diferenciação em cores dos agentes saudáveis e doentes até o desenho do layout da escola, passando também pela definição do número daqueles inicialmente infectados.
- d) *Construção de modelos computacionais* – enquanto o desenho de um modelo computacional envolve a tomada de decisões sobre o que nele incluir, a sua construção requer saber expressar os aspectos planejados, usando as ferramentas computacionais disponíveis. No caso desta atividade, o estudante tem a oportunidade de exercitar esta competência ao explorar todas possibilidades oferecidas pelo software de modelagem para execução do modelo elaborado.

3. Prática de resolução de problemas computacionais (subseção 3.2.3)

- a) *Construção de abstrações computacionais* – o exercício dessa competência se expressa na criação de representações, tais como o movimento dos agentes simbolizando a dos alunos indo de um ambiente à outro da escola e o uso de cores para expressar os estados “saudável” e “doente”.

Conclusão

A pesquisa necessária para o desenvolvimento deste trabalho, tanto quanto a sua produção, foi de especial importância para o autor refletir e ampliar seus conhecimentos sobre os diversos aspectos que envolvem a aprendizagem científica.

A relevância do tema para a formação educacional da sociedade e, em última análise, para seu progresso material exigem pesquisas, reflexões e debates contínuos, principalmente em face dos problemas de baixo de desempenho e falta de motivação dos estudantes brasileiros com relação a temas científicos, evidentes aos olhos de professores veteranos, e continuamente quantificados por testes nacionais e internacionais.

A construção desse trabalho foi orientada pela expectativa de trazer um conjunto de reflexões e de propostas que tornem o aprendizado de temas científicos, e das metodologias que lhe servem de apoio, mais atrativo ao estudante.

Com esse propósito, trouxemos à discussão o conceito de pensamento computacional e vimos como a sua adoção em sala de aula favorece a construção de um contexto ótimo para o desenvolvimento de competências científicas. Como explicado, esse fato se justifica pelos elementos que compartilha com um largo espectro de disciplinas científicas, tal como as noções de abstração e decomposição de problemas.

Como indicado pela literatura, a percepção de estar resolvendo um problema científico autêntico e que, em alguns casos, dialogue com sua realidade, tem se mostrado capaz de aumentar os níveis de engajamento e interesse dos alunos – fatores cruciais para a aprendizagem significativa – e assim lidar com o problema da motivação com o qual os professores estão habituados.

Além disso, o estímulo à construção de modelos e o convite à reflexão crítica sobre os seus limites de validade, trazidos por essa prática, propicia a formação de um novo tipo de relação entre o estudante e as disciplinas científicas, que substitui o seu papel de apenas de reproduzir o que lhe foi transmitido, durante uma avaliação.

Como pudemos discutir, apesar de haver consenso sobre os princípios e elementos que compõe essa conceituação, o mesmo não ocorre com relação às diretrizes e estratégias para sua aplicação em sala de aula. Mesmo que o seu potencial pedagógico tenha sido identificado, podemos ainda observar na literatura, por exemplo, percepções difusas sobre quais competências cognitivas e científicas a serem desenvolvidas quando se pretende fazer com que os alunos pensem “computacionalmente”.

A contribuição a ser dada, objetivada nessa monografia, é a repercussão do trabalho de [Weintrop et al. \(2016\)](#) que define com clareza tais competências e a demonstração

do como a taxonomia que as incorpora pode ser útil na construção de atividades com conteúdo cientificamente relevante.

Referências

ABRAHAMSON, D.; WILENSKY, U. Problab goes to school: design, teaching, and learning of probability with multi-agent interactive computer models. 2005. Disponível em: <https://ccl.northwestern.edu/2005/Abr+Wil_CERME4.pdf>. Citado na página 20.

BARR, V.; STEPHENSON, C. Bringing computational thinking to K-12. *ACM Inroads*, v. 2, n. 1, p. 48, 2011. ISSN 21532184. Disponível em: <<http://dl.acm.org/citation.cfm?doid=1929887.1929905>>. Citado 4 vezes nas páginas 20, 24, 27 e 28.

BOX, G.; DRAPER, N. *Empirical model-building and response surfaces*. Wiley, 1987. (Wiley series in probability and mathematical statistics: Applied probability and statistics). ISBN 9780471810339. Disponível em: <<https://books.google.com.br/books?id=QO2dDRufJEAC>>. Citado na página 34.

BRENNAN, K.; RESNICK, M. New frameworks for studying and assessing the development of computational thinking. *annual American Educational Research Association meeting, Vancouver, BC, Canada*, p. 1–25, 2012. Disponível em: <http://web.media.mit.edu/{~}kbrennan/files/Brennan{_\}Resnick{_\}AERA201>. Citado na página 24.

CLAXTON, G. *Live and Learn: An Introduction to the Psychology of Growth and Change in Everyday Life*. Harper & Row, 1984. ISBN 9780063182776. Disponível em: <<https://books.google.ae/books?id=ht0NAAAAMAAJ>>. Citado na página 2.

DJORGOVSKI, S. Virtual Astronomy, Information Technology, and the New Scientific Methodology. *Seventh International Workshop on Computer Architecture for Machine Perception (CAMP'05)*, p. 125–132, 2005. Disponível em: <<http://ieeexplore.ieee.org/document/1508175/>>. Citado 3 vezes nas páginas 4, 5 e 8.

ESCOBAR, M. *Artificial intelligence: here's what you need to know to understand how machines learn*. 2017. Disponível em: <<https://theconversation.com/artificial-intelligence-heres-what-you-need-to-know-to-understand-how-machines-learn-72004>>. Acesso em: 10 de Setembro de 2018. Citado na página 5.

GOUGH, W. A. J. *Practical Arithmetic: In Four Books*. [S.l.: s.n.], 1813. Citado na página 10.

HATCH, R. A. *Ptolemy's planetary models*. 1998. Disponível em: <<http://users.clas.ufl.edu/ufhatch/pages/03-Sci-Rev/SCI-REV-Home/resource-ref-read/chief-systems/08-0PTOL3-WSYS.html>>. Acesso em: 11 de Setembro de 2018. Citado na página 6.

LEE, I. et al. Computational thinking for youth in practice. *ACM Inroads*, v. 2, n. 1, p. 32, 2011. ISSN 21532184. Disponível em: <<http://dl.acm.org/citation.cfm?doid=1929887.1929902>>. Citado 3 vezes nas páginas 20, 49 e 50.

LOCKE, J. *An essay concerning human understanding*. New York, NY, USA: Oxford Univerty Press, 1690/1979. Citado na página 13.

- MALONEY, J. H. et al. Programming by choice: urban youth learning programming with scratch. ACM, p. 367–371, 2008. Disponível em: <<http://dblp.uni-trier.de/db/conf/sigcse/sigcse2008.html#MaloneyPKRR08>>. Citado na página 39.
- NATIONAL RESEARCH COUNCIL. *Report of a workshop on the scope and nature of computational thinking*. Washington, DC: The National Academies Press, 2010. Citado 2 vezes nas páginas 24 e 26.
- NOVAK, M. Connected chemistry – gas laws. Center for Connected Learning and Computer-based Modeling at Northwestern University, 2016. Disponível em: <<https://ccl.northwestern.edu/classroomresources/connectedchem/index.shtml>>. Acesso em: 05 de Janeiro de 2019. Citado na página 47.
- OREMUS, W. *The Fascinating, Mostly Failed History of “Teaching Machines”*. 2015. Disponível em: <http://www.slate.com/articles/slate_plus/technology/2015/10/the_history_of_learning_machines_from_sidney_presser_and_b_f_skinner_to.html>. Acesso em: 03 de Novembro de 2018. Citado na página 23.
- PATHAK, J. et al. Model-free prediction of large spatiotemporally chaotic systems from data: A reservoir computing approach. *Phys. Rev. Lett.*, American Physical Society, v. 120, p. 024102, Jan 2018. Disponível em: <<https://link.aps.org/doi/10.1103/PhysRevLett.120.024102>>. Citado na página 5.
- PIWEK, P. Introduction to computational thinking. 2016. Disponível em: <<http://doer.col.org/handle/123456789/6196>>. Citado 7 vezes nas páginas 9, 12, 13, 14, 15, 16 e 17.
- POZO, J.; CRESPO, M. ángel G. *Aprender y enseñar ciencia : del conocimiento cotidiano al conocimiento científico*. [S.l.: s.n.], 1998. Citado 2 vezes nas páginas 1 e 2.
- PURTILL, C. *Apple, IBM, and Google don't care anymore if you went to college*. 2018. Disponível em: <<https://qz.com/work/1367191/apple-ibm-and-google-dont-require-a-college-degree/>>. Acesso em: 12 de Outubro de 2018. Citado na página 19.
- RESNICK, M. *Let's teach kids to code*. 2012. Disponível em: <https://www.ted.com/talks/mitch_resnick_let_s_teach_kids_to_code?language=en>. Acesso em: 14 de Outubro de 2018. Citado 3 vezes nas páginas 20, 21 e 39.
- SENGUPTA, P. et al. Integrating computational thinking with K-12 science education using agent-based computation: A theoretical framework. *Education and Information Technologies*, v. 18, n. 2, p. 351–380, 2013. ISSN 13602357. Citado 2 vezes nas páginas 13 e 19.
- VERHAGE, J. *This Robot Said to Sell Facebook. Next Time It May Be Right*. 2017. Disponível em: <<https://www.bloomberg.com/news/articles/2017-11-21/this-robot-said-to-sell-facebook-next-time-it-may-be-right>>. Acesso em: 11 de Setembro de 2018. Citado na página 7.
- VUTHA, A. *Could machine learning mean the end of understanding in science?* 2018. Disponível em: <<https://theconversation.com/could-machine-learning-mean-the-end-of-understanding-in-science-98995>>. Acesso em: 10 de Setembro de 2018. Citado 3 vezes nas páginas 5, 6 e 7.

- WEINTROP, D. et al. Defining Computational Thinking for Mathematics and Science Classrooms. *Journal of Science Education and Technology*, Springer Netherlands, v. 25, n. 1, p. 127–147, 2016. ISSN 15731839. Citado 24 vezes nas páginas 3, 4, 19, 20, 23, 24, 25, 26, 29, 30, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44 e 52.
- WILENSKY, U. *Netlogo*. 1999. [Http://ccl.northwestern.edu/netlogo/](http://ccl.northwestern.edu/netlogo/). Citado 2 vezes nas páginas 20 e 46.
- WILENSKY, U. NetLogo Connected Chemistry 4 Number and Pressure model. Center for Connected Learning and Computer-Based Modeling, Northwestern University, 2004. Disponível em: <<http://ccl.northwestern.edu/netlogo/models/ConnectedChemistry4NumberandPressure>>. Acesso em: 10 de Janeiro de 2019. Citado na página 46.
- WING, J. Computational Thinking. *Commun. ACM*, ACM, New York, NY, USA, v. 49, n. 3, p. 33–35, 2006. Citado 5 vezes nas páginas 4, 9, 11, 22 e 23.
- WING, J. Computational thinking and thinking about computing. v. 366, p. 3717–25, 11 2008. Citado 2 vezes nas páginas 8 e 35.