MEC-SETEC

INSTITUTO FEDERAL DE MINAS GERAIS - *Campus* Formiga

Curso de Ciência da Computação

MODELOS COMPUTACIONAIS: COMPUTABILIDADE E CLASSES DE COMPLEXIDADE

Suena Batista Galoneti

Orientador: Mário Luiz Rodrigues Oliveira

Formiga - MG

2020

Suena Batista Galoneti

MODELOS COMPUTACIONAIS: COMPUTABILIDADE E CLASSES DE COMPLEXIDADE

Trabalho de Conclusão de Curso apresentado ao Instituto Federal Minas Gerais - Campus Formiga, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

Orientador: Mário Luiz Rodrigues Oliveira

Formiga - MG

2020

Galoneti, Suena Batista
G178m Modelos Computacionais: Computabilidade e Classes de Complexidade / Suena Batista Galoneti -- Formiga : IFMG, 2020.
71p. : il.

Orientador: Prof. Mário Luiz Rodrigues Oliveira
Trabalho de Conclusão de Curso – Instituto Federal de Educação, Ciência e Tecnologia de Minas Gerais – *Campus* Formiga.

1. Revisão Sistemática da Literatura. 2. Teoria da Computação. 3. Modelos Computacionais. 4. Computabilidade. 5. Classes de Complexidade. I. Oliveira, Mário Luiz Rodrigues. II. Título.

CDD 004

Ficha catalográfica elaborada pela Bibliotecária Msc. Simoni Júlia da Silveira

SUENA BATISTA GALONETI

Modelos Computacionais: Computabilidade e Classes de Complexidade

Trabalho de Conclusão de Curso apresentado ao Instituto Federal de Minas Gerais-Campus Formiga, como Requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

Aprovado em: 26 de outubro de 2020.

BANCA EXAMINADORA

Prof. MÁRIO LUIZ RODRIGUES OLIVEIRA (Orientador)

Profa. DANIELLE COSTA DE OLIVEIRA (Membro1)

Prof. WALACE DE ALMEIDA RODRIGUES (Membro2)

Documento assinado eletronicamente por **Mario Luiz Rodrigues Oliveira**, **Professor**, em 27/10/2020, às 09:12, conforme art. 1º, III, "b", da Lei 11.419/2006.

Documento assinado eletronicamente por **Danielle Costa de Oliveira**, **Professora**, em 27/10/2020, às 09:19, conforme art. 1º, III, "b", da Lei 11.419/2006.

Documento assinado eletronicamente por **Walace de Almeida Rodrigues**, **Professor**, em 27/10/2020, às 14:26, conforme art. 1º, III, "b", da Lei 11.419/2006.

AGRADECIMENTOS

Primeiramente à Deus, por ser essencial em minha vida, autor de meu destino e meu guia.

Aos meus pais, irmãs e a toda minha família que, com muito carinho, apoio e conselhos, não mediram esforços para que eu chegasse até esta etapa de minha vida.

A todos os professores que me agregaram durante toda a minha vida acadêmica e principalmente nesta última jornada, que além de me formar uma cidadã culta me forma, agora, uma profissional. Aqui quero agradecê-los e representá-los nas pessoas dos professores Mário e Walace, que durante toda a execução desse trabalho me acompanharam e conduziram para que eu pudesse chegar até aqui.

As experiências nessa jornada não seria tão proveitosa e marcante se não fosse compartilhada com as amizades que aqui surgiram, então a eles também, meu muito obrigada (#foreverpessoaldamesinha e #periodofatorial).

À todos que cruzei no IF e fora dele durante esses anos, que me ofertaram um sorriso, uma palavra de sabedoria, um ombro amigo ou até mesmo apenas um bom dia, mas que Assim fizeram parte e melhoram os meus dias ao longo desses anos.

E por último a minha mãe, meus cachorros e paredes que foram ótimos alunos nos momentos de estudos.

"A menos que modifiquemos a nossa maneira de pensar,
não seremos capazes de resolver os problemas
causados pela forma como nos acostumamos a ver o mundo"
(Albert Einstein)

RESUMO

A ciência da computação pode ser considerada uma descoberta recente dentre as demais áreas de estudos da ciência, mas mesmo assim passou por várias inovações, sendo projetada sobre vários diferentes modelos neste curto intervalo de tempo.

Esse trabalho visa fornecer um compilado com informações sobre os modelos computacionais desde o início da computação até as inovações do cenário atual.

Foram elaboradas três perguntas com o intuito de comparar modelos computacionais a fim de conseguir listá-los e classificá-los de acordo com sua complexidade, desempenho e usabilidade:

- Quais são os modelos computacionais existentes?
- Qual modelo computacional mais viável (custo computacional x poder computacional e possibilidade de desenvolvimento em larga escala)?
- Qual o modelo computacional mais didático?

Utilizando a metodologia Revisão Sistemática da Literatura, encontram-se aqui documentado todos os passos aplicados em cada uma das etapas de planejamento, execução e análise, e ao final tem como objetivo fornecer algumas respostas partindo da comparação dos modelos.

Inicialmente, quando proposta essa pesquisa, era esperado encontrar informações sólidas e métricas claras e simétricas sobre cada um dos modelos retornados, o que não se tornou verdade em cem por cento dos casos, fornecendo assim resultados parciais: evidenciando como modelos mais atrativos a computação em DNA e Quântica e como modelo mais didático a Máquina X. Contudo, as demais perguntas não puderam ser satisfatoriamente respondidas.

Todavia, encontram-se aqui importantes modelos computacionais: Rede Neural Recorrente Analógica, Autômato Celular, Computação em DNA, Computação Quântica, Lambda Cálculo, Máquina de Post, Máquina de registradores, Máquina de Turing, Máquina de Turing Probabilística e Máquina X.

Palavras-chave: Modelos Computacionais. Revisão Sistemática da Literatura. Teoria da Computação.

ABSTRACT

Computer science can be considered a recent discovery among other areas of science studies, but even so it has already undergone several innovations, being projected on several different models in this short period of time.

This work aims to provide a compilation with information on computational models from the beginning of computing to the innovations of the current scenario.

Three questions were prepared in order to compare computational models in order to be able to list and classify them according to their complexity, performance and usability:

- What are the existing computational models?
- Which the most viable computational model (computational cost x computational power and possibility of large-scale development)?
- What is the most didactic computational model?

Using the Systematic Literature Review methodology, here are documented all the steps applied in each of the planning, execution and analysis stages, and at the end it aims to provide some answers based on the comparison of the models.

Initially, when this research was proposed, it was expected to find solid and clear and symmetrical metrics about each of the returned models, which did not become true in 100% of the cases, thus providing partial results: evidencing computer computing as the most attractive models. DNA and Quantum and Machine X as the most didactic model. However, the other questions could not be satisfactorily answered.

However, here are important computational models: Recurrent Analog Neural Network, Cellular Automaton, DNA Computing, Quantum Computing, Lambda Calculus, Post Machine, Recorder Machine, Turing Machine, Probabilistic Turing Machine and Machine X.

Keywords: Computational Models. Systematic Literature Review. Computer Theory.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

A	Adenosina
ACM	ACM Digital Library
ARNN	Rede Neural Anagógica Recorrente
BPP	Bounded-Error Probabilistic Polynomial Time
BQP	Bounded-Error Quantum Polynomial Time
C	Citosina
CQ	Circuito Quântico
DNA	Ácido Desoxirribonucleico
FSM	Finit State Machine
G	Guanina
IEEE	IEEEXplore
MT	Máquina de Turing
MTND	Máquina de Turing Não Determinística
MTP	Máquina de Turing Probabilística
MTQ	Máquina de Turing Quântica
NP	Non-Deterministic Polynomial Time
P	Polynomial-Time
PSPACE	Polynomial Space
RSL	Revisão Sistemática da Literatura
RW	Read and Write
SC	Scopus
SD	Science Direct
SP	Spring Link
WoS	Web of Science

LISTA DE SÍMBOLOS

Γ	Letra grega Gama
Λ	Letra grega Lambda maiúscula
λ	Letra grega Lambda minúscula
δ	Letra grega Delta
β	Letra grega beta
ε	Letra grega Epsilon
ψ	Letra grega Psi
Σ	Letra grega Sigma maiúscula
σ	Letra grega Sigma minúscula
$\in$	Pertence
∞	Infinito
$\cup$	União
$\forall$	Para Todo

SUMÁRIO

1 INTRODUÇÃO

Nos dias atuais os computadores se fazem cada vez mais presentes, podendo ser encontrados tanto em grandes laboratórios tecnológicos quanto em relógios de pulso.

Ao se comparar os computadores atuais com os primeiros desenvolvidos pode-se perceber uma enorme evolução, assim como a prevista por Moore em sua lei, que afirma que a velocidade de um computador é dobrada a cada 12 meses. Deste modo, sempre houve um crescimento constante na velocidade de processamento dos computadores. Entretanto, essa evolução tem um certo limite, um ponto no qual não é possível aumentar essa velocidade e então se faz necessário uma revolução significativa na computação (TANENBAUM, 2013).

A ciência se desenvolve de maneira cumulativa: as pesquisas normalmente evoluem com base em resultados obtidos do passado.

A fundamentação da segurança do cientista está em suas pesquisas que o preparam para novas descobertas a partir dos dados obtidos, como evolução histórica do assunto, quais as maiores dificuldades, as soluções encontradas até o presente momento e os problemas ainda pendentes (FILHO, 2007).

Quando comparada a outras áreas da ciência, a computação é muito recente. Nesses poucos anos (aponta-se a Segunda Guerra Mundial como marco inicial dado que foi quando efetivamente construíram-se os primeiros computadores) o seu avanço tem sido exponencial, abrindo-se um grande leque de tecnologias, conceitos e ideias, o que tornou a computação uma ciência complexa, ampla, geradora de novos enfoques, passando a se apresentar como um verdadeiro desafio a quem queira entendê-la e traçar sua evolução (FILHO, 2007).

No entanto, a Teoria da Computação teve origem na lógica formal, muito antes da existência de computadores.

> *"Descartes acreditava no emprego sistemático do cálculo algébrico como um método poderoso e universal para resolver todos os problemas. Esta crença juntou-se à de outros e surgem as primeiras ideias sobre máquinas universais, capazes de resolver todos os problemas. Esta era uma crença de mentes poderosas que deixaram obras respeitáveis na Matemática e nas ciências em geral"* (CARVALHO, 1998).

A Ciência da Computação vem se expandindo em várias direções, fazendo-se necessário desenvolver novos formalismos, que são importantes para ter um sistema bem definido. Devido a essa expansão, se tornou possível distinguir duas áreas na Ciência da Computação: uma aborda comportamento interativo, e a outra aborda comportamento

algorítmico. A Teoria da Computação Clássica enquadra-se no segundo, sendo considerada como base da Ciência da Computação (PAPADIMITRIOU, 1998). Diante dessa evolução da Ciência da Computação, a Teoria da Computação teve seu desenvolvimento impulsionado pelas necessidades originadas pelos avanços tecnológicos, requerendo, assim, uma maior elaboração de novos formalismos e um maior detalhamento dos existentes (PY, 2003).

Como qualquer outra ciência, a computação teve a sua história pautada por perguntas que impulsionaram seu desenvolvimento, dentre essas perguntas algumas se destacaram nos estudos desenvolvidos, como:

- O que pode ser computável?
- Como descrever o que computar?
- Como computar?

Objetivando responder perguntas como estas, a teoria da computação tem como escopo estudar modelos de computação suficientemente genéricos a fim de especificar qualquer função computável e explorar os limites do que pode ser computado (DIVERIO; MENEZES, 2009).

No início do século XX, ocorreu um grande impulsionamento nas pesquisas da área da teoria da computação. Um marco de grande destaque foi o trabalho realizado por David Hilbert, matemático alemão, que propôs um dos grandes desafios para aquele século: o Entscheidungsproblem (DIVERIO; MENEZES, 2009):

> *"encontrar um algoritmo que recebe como entrada a descrição de uma linguagem formal e uma sentença nesta linguagem e tem como saída "verdadeiro"ou "falso" dependendo se a sentença de entrada é verdadeira ou falsa.*

O matemático Hilbert acreditava que todo problema bem definido teria solução e por isso desafiou a comunidade científica em buscar um procedimento efetivo para resolver cada um dos problemas bem definidos. Contudo na década de 30, os matemáticos Alonzo Church e Alan Turing, mostraram que existem problemas para os quais não há procedimento efetivo que os resolva e assim responderam de forma negativa ao desafio proposto por Hilbert (DIVERIO; MENEZES, 2009).

Neste mesmo ano, Turing apresentou um formalismo para a representação de procedimentos efetivos, tornando-se pioneiro na identificação de programas escritos para uma "máquina computacional", e assim noções intuitivas do efetivamente computável. Este trabalho atribui grande importância a Turing, se tratando de um conhecimento muito significativo e de fundamental importância para a ciência da computação (DIVERIO; MENEZES, 2009).

Deste modo, este trabalho foi elaborado preocupando-se com as definições e conceitos estabelecidos desde o início da Computação, chamando a atenção para a importância da Máquina de Turing, Máquina de Turing Quântica, Lambda Cálculo e Computação baseada em DNA, entre outros modelos computacionais. Essas noções se fizeram necessárias para abordar computabilidade na Teoria da Computação.

Nesse contexto, o trabalho desenvolveu a ideia de uma revisão sistemática sobre os modelos de computação, sua computabilidade e classes de complexidade.

1.1 Justificativa

Ao conhecer os modelos de computação o cientista passa a compreender de uma maneira mais completa a Ciência da Computação e seu desenvolvimento atrelado às tecnologias disponíveis.

A compreensão dos modelos de computação permite aos cientistas e desenvolvedores entender os limites e capacidades da resolução de problemas por meio algorítmicos. Assim, ao se ter uma base maior de informações sobre a os diversos modelos de computação e a Teoria de Computabilidade o cientista passa a ter uma nova visão sobre a computação e sobre o que esperar dela no futuro próximo.

Com o constante e acelerado crescimento da quantidade de informação científica, as Revisões Sistemáticas da Literatura (RSL) desempenham um papel de grande importância, pois seu principal objetivo é consolidar as informações em conhecimento, ou seja, os dados disponíveis na literatura sobre um determinado tema são reunidos, organizados, avaliados e interpretados (JESUS, 2015).

Ao viabilizarem, de forma clara e explícita, um resumo de todos os estudos sobre determinado tema, as revisões sistemáticas nos permitem incorporar um espectro maior de resultados relevantes, ao invés de limitar as nossas conclusões á leitura de somente alguns artigos, sendo particularmente úteis na orientação para investigações futuras (CORDEIRO[1] et al., 2007).

Sendo assim, boas revisões sistemáticas são recursos importantes ante o crescimento acelerado da informação científica. Esses estudos ajudam a sintetizar a evidência disponível na literatura sobre uma intervenção, podendo auxiliar pesquisadores no seu cotidiano de trabalho (SAMPAIO; MANCINI, 2007).

Muito embora revisões sistemáticas sejam uma importante ferramenta em um contexto de acelerado crescimento de informações, registra-se que a execução desta RSL supre uma lacuna de conteúdos em literatura de língua portuguesa na área de teoria da computação, visto que realizada uma pesquisa preliminar e não se encontrou nenhum conteúdo similar ao que será apresentado aqui.

1.2 Objetivos

1.2.1 Objetivo Geral

Este trabalho tem como objetivo o desenvolvimento de uma Revisão Sistemática da Literatura sobre os Modelos Computacionais, sua computabilidade e classes de complexidade.

1.2.2 Objetivos Específicos

Além de contribuir com um compilado de informações sobre os Modelos Computacionais, pretende-se responder as seguintes perguntas:

- Quais são os modelos computacionais existentes?
- Qual modelo computacional mais viável (custo computacional x poder computacional e possibilidade de desenvolvimento em larga escala)?
- Qual o modelo computacional mais didático?

1.3 Estrutura do trabalho

Este trabalho é composto por seis capítulos, sendo que este é o primeiro, onde fez-se as apresentações das ideias iniciais do projeto, com os objetivos e a justificativa.

No Capítulo 2, é apresentado a fundamentação teórica trazendo os principais conceitos utilizados nesse trabalho.

No Capítulo 3 é abordado a metodologia de Revisão Sistemática da Literatura.

No Capítulo 4 contempla o desenvolvimento descrevendo o passo a passo da execução da Revisão.

No Capítulo 5 são explanados os modelos encontrados como resultados da Revisão Sistemática da Literatura.

Por fim, no Capítulo 6 são apresentadas as considerações finais, discussões sobre este trabalho e propostas para trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 Computabilidade

Uma das finalidades do estudo da computabilidade é determinar a solucionabilidade de problemas, em outras palavras, a existência de algoritmos e, desse modo, explorar os limites do que pode ser implementado em um computador evitando a pesquisa de soluções inexistentes (DIVERIO; MENEZES, 2009).

O estudo da computabilidade baseia-se nos problemas de decisão. Tais problemas caracterizam por terem como respostas duas opções: sim ou não. Essa abordagem foi adotada por permitir o tratamento dos problemas como linguagens. Um importante resultado da teoria da computação é como provar a não existência de procedimento efetivo para solucionar problemas de decisão. Em síntese, a teoria da computação explora os limites da computabilidade e por conseguinte os limites do que pode efetivamente ser implementado em um computador. Um exemplo clássico é a prova da não existência de procedimento efetivo para o problema da parada (DIVERIO; MENEZES, 2009).

O aparecimento dos computadores no final da década de 1930 e início da década de 1940 destacou, gradualmente, uma diferença entre os problemas decidíveis. Essa diferença baseia-se no fato de que muitos dos problemas aparentam ser mais difíceis do que outros, o que ocorre devido à somente se ter conhecimento de algoritmos extremamente lentos que os resolvam. Surgiu, então, a necessidade de se explicitar essa distinção refinando a teoria da computabilidade (CARDONHA; SILVA; FERNANDES, 2004).

A fim de aprofundar o estudo de tais questões, consolidou-se, na década de 1960, a teoria de complexidade, formalizando a ideia de algoritmo eficiente e a propondo reduções eficientes entre problemas. Também nessa época foram apresentadas as definições de classes de complexidade P e NP , assim como o conceito de NP-completude, delimitando os problemas de acordo com a dificuldade de se conseguir algoritmos eficientes para resolvê-los (CARDONHA; SILVA; FERNANDES, 2004).

2.2 Classes de Complexidade

Com a utilização de computadores deixando de ser limitada a instituições científicas e assim ampliando seu uso os programadores foram capazes de perceber que a existência de um algoritmo para uma determinada tarefa não era suficiente para que ela pudesse ser resolvida por um computador. A partir disso foram identificados uma grande quantidade de problemas para os quais até mesmo os melhores algoritmos conhecidos apresentavam

um execução com tempos tão exorbitantes que inviabilizavam a busca de resposta através de meios computacionais. Restava apenas estabelecer se essas observações adivinham da incapacidade humana de implementar um algoritmo realmente eficiente ou se tratava da dificuldade inerente do problema em consideração (OLIVEIRA, 2010).

A teoria de computabilidade fornece métodos para classificar quais problemas são solucionáveis por algoritmos, particionando-os em diversas classes de acordo com a quantidade de recursos computacionais necessários e suficientes para resolver cada um deles (OLIVEIRA, 2010).

Com apenas 40 anos de exploração, essa área, atualmente, abrange grande parte das pesquisas dentro da ciência/teoria da computação, sendo responsável por buscar e determinar as razões que tornam certos problemas decidíveis difíceis de serem resolvidos por computadores (OLIVEIRA, 2010).

A área de pesquisa da complexidade computacional é ramificada em dois vieses: o estudo da dificuldade de um problema computacional específico e a investigação da relação entre classes computacionais e recursos computacionais. Apesar da existência dessa divisão, pode-se reduzir o estudo de classes de problemas ao estudo de problema individuais que capturam propriedades essenciais da classe (OLIVEIRA, 2010).

A classe de problemas P (*polynomial-time*) denota o conjunto de problemas que podem ser resolvidos em tempo polinomial, no tamanho da entrada, por uma máquina de *Turing* determinística. A classe NP (*nondeterministic polynomial-time*) denota o conjunto de problemas que podem ser resolvidos em tempo polinomial, no tamanho da entrada, por uma máquina de *Turing* não determinística. (OLIVEIRA, 2010)(CORMEN et al., 2009).

Rapidamente, os pesquisadores puderam observar uma característica importante/interessante presente em diversos problemas computacionais difíceis: dada uma possível solução para uma instância do problema a sua verificação como uma das soluções buscadas ocorre eficientemente. A partir deste fato, denominou-se a classe desses problemas como NP (OLIVEIRA, 2010).

Uma das possibilidades de solução para problemas pertencentes à classe NP pode ser escrita como a concatenação de dois métodos: geração de todas as possíveis soluções para uma dada instância do problema e seguindo da verificação das soluções encontradas. Apesar da definição de classe NP garantir que a verificação ocorre eficientemente, esse algoritmo como um todo não possui essa propriedade, pois a primeira etapa, conhecida como busca exaustiva, requer um número de passos computacionais proporcional ao número de possibilidades no espaço de busca, o que é, de modo geral, uma função de crescimento exponencial no tamanho das instâncias do problema (OLIVEIRA, 2010).

Apesar da existência de algoritmos cuja a complexidade seja exponencial para determinado problema não significa necessariamente que a complexidade real desse pro-

blema seja exponencial. Com a crescente descoberta e implementação de novos algoritmos tornou-se amplamente aceita a definição sugerida por Cobham e Edmonds, a qual apresenta a seguinte propriedade: um algoritmo é dito eficiente quando seu número de passos é limitado por uma função algébrica, ou seja, por um polinômio (OLIVEIRA, 2010).

Na atualidade problemas como criptografia moderna que garante a segurança da Internet e de grande parte das transações financeiras caracterizam alguns dos problemas que pertencem à classe NP, e nesses casos algumas de suas implementações baseiam-se em um conhecido obstáculo dessa classe: a dificuldade de se fatorar números inteiros muito grandes (OLIVEIRA, 2010).

Outra definição importante dentro da complexidade computacional é a de NP-completude, a qual afirma que se qualquer problema pertencente à classe NP pode ser reduzido de forma eficiente a um determinado problema também NP, este último é então um problema NP-completo (CARDONHA; SILVA; FERNANDES, 2004).

Quando se fala de complexidade computacional não se pode deixar de mencionar a mais famosa questão da área de teoria da área da computação: se P é ou não igual a NP; e a forma de se solucionar tal impasse seria demonstrando que algum problema NP-completo está em P, e assim implicar que P = NP (CARDONHA; SILVA; FERNANDES, 2004).

Com o advento da computação quântica foi necessário formalizar a ideia de complexidade também para esse novo modelo computacional. Dentro desses modelos quânticos há a classe denominada BQP (*Bounded Error Quantum Polynomial*) quem contém os problemas de decisão resolvidos por modelos de computação quânticos com probabilidade de erro de até 1/3 (GRILO, 2010).

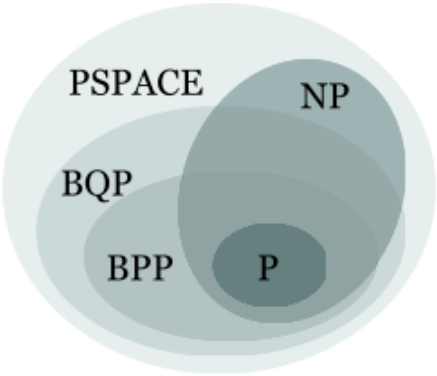
Também é importante citar modelos probabilísticos, tal como a classe BPP *Bounded-Error Probabilistic Polynomial Time* que contém os problemas de decisão resolvidos por uma máquina de *Turing* probabilística em tempo polinomial com uma probabilidade de erro de até 1/3 (GRILO, 2010).

Ressalta-se ainda a classe de problemas PSPACE *Polynomial Space*. Essa classe agrupa os problemas que podem ser resolvidos utilizando-se uma quantidade de memória de tamanho polinomial em relação ao tamanho da entrada do problema em uma Máquina de *Turing* (GRILO, 2010).

A relação entre as classe BQP, BPP e PSPACE é conforme a Figura 1.

A computação quântica trouxe consigo uma dúvida: se um computador quântico pode ou não resolver problemas NP para os quais não se sabe se podem ser resolvidos de maneira eficiente no modelo computacional clássico. Essa questão agrega grande importância pelo fato da classe NP envolver, entre outros problemas, os modelos criptográficos utilizados atualmente para garantir a segurança da Internet e de grande parte das transações financeiras (GRILO, 2010).

Figura 1 – Relação Entre as classes de Complexidade



Fonte: (GRILO, 2010)

Tabela 1 – Tabela Resumo das Classes de Complexidade

Classe de Complexidade	Descrição
P	Conjunto de problemas que podem ser resolvidos em tempo polinomial, no tamanho da entrada, por uma máquina de *Turing* determinística.
NP	Conjunto de problemas que podem ser resolvidos em tempo polinomial, no tamanho da entrada, por uma máquina de *Turing* não determinística.
PSPACE	Conjunto de problemas que podem ser resolvidos utilizando-se uma quantidade de memória de tamanho polinomial em relação ao tamanho da entrada do problema em uma Máquina de *Turing*.
BPP	Conjunto dos problemas de decisão resolvidos por uma máquina de *Turing* probabilística em tempo polinomial com uma probabilidade de erro até 1/3.
BQP	Conjunto dos problemas de decisão resolvidos por modelos de computação quânticos com probabilidade de erro de até 1/3.

Fonte: Elaborado pela Autora

3 MATERIAIS E MÉTODOS

3.1 Revisão Sistemática da Literatura

A Revisão Sistemática da Literatura(RSL) é uma técnica de pesquisa científica objetiva que visa avaliar e interpretar trabalhos disponíveis e relevantes para uma determinada questão de pesquisa, tendo como objetivo, ao final, ter um conjunto organizado de documentos selecionados que respondam a questão proposta (JESUS, 2015).

Seguindo com a explicação da ferramenta escolhido, a parte sistêmica pode ser dita como a parte principal, sendo a qual o estrutura, e tendo como características base a presença de métodos bem definidos e ser reprodutível (JESUS, 2015).

Antes de se iniciar uma RSL deve-se primeiro pesquisar e embasar a necessidade de fazê-la, justificando assim a razão do seu trabalho. Todavia, este presente trabalho contém uma seção própria - Seção Justificativa- na qual se encontra devidamente defendido seu mérito.

O processo de desenvolvimento de uma RSL pode ter suas atividades divididas em três categorias: planejamento, execução e análise de resultados (JESUS, 2015).

Na primeira etapa, o planejamento, é onde o protocolo é especificado, e esse tem por objetivo minimizar a possibilidade de ocorrência de vieses. Sendo o elemento essencial para a execução de uma RSL, sua qualidade tem um impacto direto sobre ela, dessa forma, o protocolo deve ser avaliado antes que se prossiga com a revisão (SENOCAK, 2019).

As fases que compõem esta metodologia devem ser explicitamente estabelecidas no protocolo, assim como a(s) questão(ões) de pesquisa, a estratégia que será utilizada para conduzir a RSL, as fontes consideradas para busca, os critérios de para a inclusão, exclusão e definição de qualidade dos estudos, extração dos dados relevantes e sumarização dos mesmos (SENOCAK, 2019).

É também nesse primeiro estágio que geralmente são descritos os objetivos, a amostra, o método e o cronograma sobre os quais será executada a revisão (JESUS, 2015).

Com a fase de planejamento consolidada e aprovada, segue-se então para a próximo estágio da revisão: a execução.

Durante esta etapa são realizados os passos de identificação, seleção e coleta de dados.

Dentro da primeira atividade os estudos são identificados a partir de uma estratégia de busca ampla, utilizando diversas bases de dados, assim aumentando a capacidade de

retorno de todos os trabalhos disponíveis pertinentes ao escopo da pesquisa (SENOCAK, 2019).

No segundo tópico, uma vez identificados os estudos são analisados de acordo com os critérios de seleção - parâmetros para inclusão ou exclusão - e de qualidade que foram previamente definidos na elaboração do protocolo na fase anterior (SENOCAK, 2019).

Dentro dos tipos de estudos que podem ser retornados/selecionados existem os estudos primários e secundários (MAFRA; TRAVASSOS, 2006):

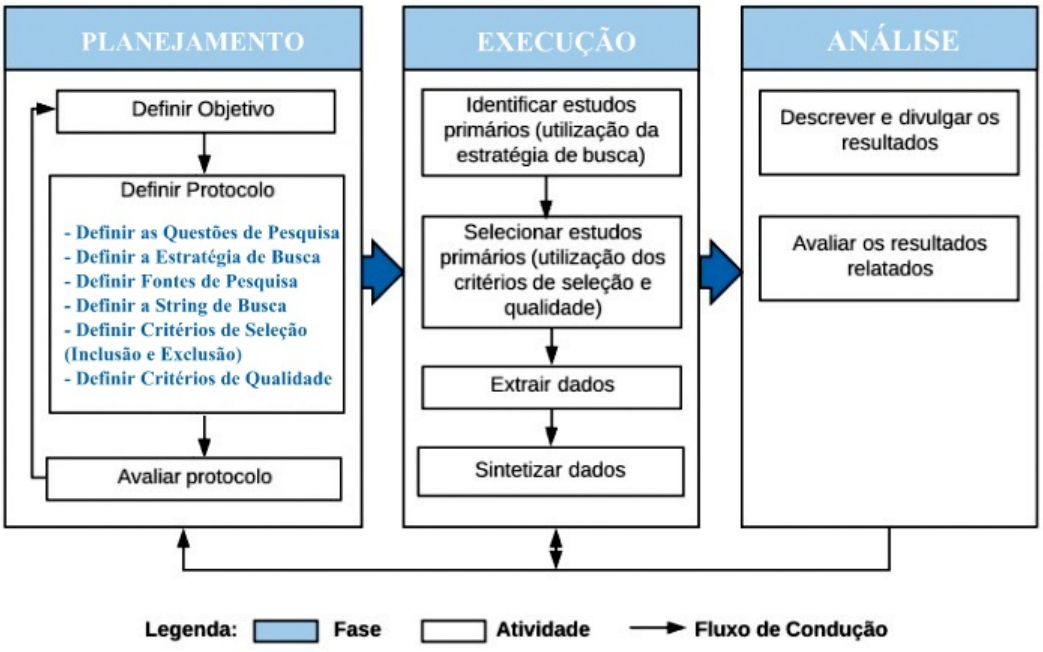
- Estudos Primários: estão dentro desta classe os estudos que correspondem a investigações originais, assim como trabalhos que visam caracterizar uma determinada tecnologia em uso dentro de um contexto específico.
- Estudos Secundários: são estudos que procuram estabelecer conclusões a partir de estudos primários, identificando, avaliando e interpretando seus resultados relevantes relacionados a um determinado tópico de pesquisa, assim sintetizam evidências sobre o mesmo.

Após classificação dos documentos é realizada a extração e síntese de dados.

Finalmente inicia-se a última etapa da RSL: apresentação e interpretação dos resultados. Nesta fase, os resultados obtidos na análise dos dados coletados são apresentados, usualmente são utilizadas tabelas, gráficos e outros artefatos para facilitar a visualização destas informações (JESUS, 2015).

E ao final, de posse de todas as evidências coletadas e devidamente apresentadas, as questões de pesquisa deverão ser respondidas (NEIVA, 2016). A figura 2 mostra, de forma esquemática, toda a metodologia aplicada.

Figura 2 – Etapas e Atividade de uma Revisão Sistemática da Literatura



Fonte: (SENOCAK, 2019)

4 ANÁLISE E APRESENTAÇÃO DOS RESULTADOS

O desenvolvimento desta RSL permitiu que fossem reunidos estudos na área da Teoria da Computação e assim efetuar um levantamento dos modelos computacionais mais estudados por esta área do conhecimento.

Essa revisão sistemática foi organizada sobre fases bem definidas. A primeira fase - o planejamento - consistiu em definir as questões de pesquisa, as strings de buscas, as fontes de buscas e os critérios de inclusão e exclusão. Na segunda fase - a execução - foram acessados as bases de buscas escolhidas e aplicado os critérios de inclusão e exclusão sobre os trabalhos retornados, os selecionando respeitando os critérios pré estabelecidos no protocolo desta revisão. Na terceira fase - a análise - foi realizada a análise individual dos trabalhos preliminares selecionados em busca de responder às questões de pesquisa definidas na primeira fase e expostas na seção a seguir.

4.1 Fase de Planejamento

Essa etapa consiste na especificação das questões de pesquisa, elaboração e evolução do protocolo da RSL que abrange o desenvolvimento da *String* de Busca, a determinação dos critérios de inclusão e exclusão e a escolhas das fontes de busca nas quais serão realizadas as pesquisas (GRANDE, 2014).

O Estabelecimento desse protocolo que controla o processo de execução da RSL torna possível garantir que as pesquisas sejam realizadas com rigor, evitando a ocorrência de vieses que possam interferir nos resultados.

4.1.1 Objetivo e questões de Pesquisa

Foram formuladas, portanto, algumas questões de pesquisa para identificar o cenário dos modelos computacionais existentes e/ou em estudo. Assim este estudo objetiva responder às seguintes questões:

- Q1: Quais são os modelos computacionais existentes?
- Q2: Qual modelo computacional mais viável (custo computacional x poder computacional e possibilidade de desenvolvimento em larga escala)?
- Q3: Qual o modelo computacional mais didático?

4.1.2 String de Busca

Para a obtenção da *string* de busca foi-se utilizado inicialmente o método de concatenar os termos ligados ao tema via o operador lógico AND e acrescentados seus sinônimos através do operador lógico OR. Porém, como estes termos usados tem uma ampla gama de empregabilidade em outras áreas e até mesmo dentro da área da computação com outros significados/objetivos retornou-se um uma quantidade "exagerada" de trabalhos (na casa dos milhões), o que tornava a execução deste inviável.

Em função da natureza iterativa do processo de uma RS, os itens que compõem o protocolo podem ser refinados durante a sua execução. Por exemplo, durante a seleção dos estudos, novos termos não inicialmente considerados na *string* de busca podem ser identificados, levando ao refinamento desta e, posteriormente, à uma nova execução das buscas por estudos primários (SENOCAK, 2019).

- Termos selecionados inicialmente e seus sinônimos:
  - modelo (padrão,paradigma,sistema) = *model,system, machine, standard, pattern, template, paradigm, ism, method, process*
  - computacional = *computational, computer, computation, machine*
  - *programming paradigm*
  - *model* comput**
  - *models of computation*
- Primeira tentativa de concatenação da String de Busca:

  ((“model” OR “system” OR “machine” OR “standard” OR “pattern” OR “template” OR “paradigm” OR “ism” OR “method” OR “process”) AND (“computation” OR “computer” OR “machine”)) OR “models of computation” OR “model comput*”*

- String final:

  "universal machine"or "models of computation"and "computer model”

No decorrer dos teste realizados com as possíveis *string* de buscas, observou-se que a utilização dos termos na forma inicial, em sua maioria substantivos simples, retornou um grande número de falsos positivos, devido, como mencionado anteriormente, a grande possibilidade de usos diferentes dos mesmos. Visto isso, optou-se por utilizar um número menor de termos e de modo mais específico com o intuito de focalizar mais a pesquisa.

Outro ponto importante a ser mencionado é que a fim de conduzir a pesquisa em diferentes repositórios eletrônicos a *string* de busca definida ao final foi adaptada de acordo com as especificações de cada base de dados.

4.1.3 Critérios de Inclusão e Exclusão

Nesta etapa estabeleceu-se os critérios necessários para a inclusão e as características indesejadas que acarretariam na exclusão ou seleção de cada um dos materiais retornados.

Visando restringir os trabalhos incluídos aplicou-se 6 critérios de inclusão e 5 de exclusão.

Os critérios estabelecidos para a execução do presente trabalho foram sumarizados na Tabela 2.

Tabela 2 – Tabela Critérios de Inclusão e Exclusão

Critérios de Inclusão	Critérios de exclusão
Idioma inglês e português	Publicados em idiomas diferentes de Inglês e Português
Publicações do tipo artigos, dissertações, teses	Estudos classificados por tutoriais, pôsteres, painéis, palestras, workshops
Acessíveis através das bases de dados conveniadas ao Instituto ou livres	Sem disponibilidade do conteúdo integral
Documentos classificados dentro da área de trabalho	Duplicados
Publicados em relações ou revistas	Estudos fora do escopo do trabalho
Relacionados com o tema do trabalho: Modelos Computacionais	

Fonte: Elaborado pela Autora

Vale a pena ressaltar que para compor o grupo de estudos incluídos dessa revisão estes deviam se tratar apenas de estudos primários, assim não englobando outras revisões. Outro ponto interessante foi a não inclusão de critério de data, que usualmente é definido, a fim de captar todos os modelos discutidos ao longo da história da computação.

4.1.4 Fontes de Buscas

Foram selecionados sete dos principais repositórios digitais da área científica, sendo que quatro deles também encontram-se disponíveis dentro da CAPES da Instituição.

- CAPES:
  - Scopus[1]
  - ScienceDirect[2]
  - Springer[3]
  - Web of Science[4]
- ACM[5]
- IEEE Xplore[6]
- Google Acadêmico[7]

4.2 Fase de Execução

Assim que concluída a fase de planejamento, passa-se a fase de execução, dando início, segundo (BA; CHARTERS, 2007), as seguintes etapas:

- Busca e identificação de estudos, que deve ser executa em cada uma das bases de pesquisa selecionadas;
- Seleção de estudos retornados, seguindo os critérios previamente definidos no protocolo da RSL;
- Avaliação da qualidade das pesquisas selecionadas, a fim de ponderar a contribuição das pesquisas selecionadas para a RSL;

Essa etapa teve sua execução realizada em um intervalo de 7 meses, Outubro de 2019 a Maio de 2020. Durante esse período foram realizadas as buscas, seleção, classificação, análise dos estudos e por fim a análise dos resultados da pesquisa.

4.2.1 Processo de Buscas

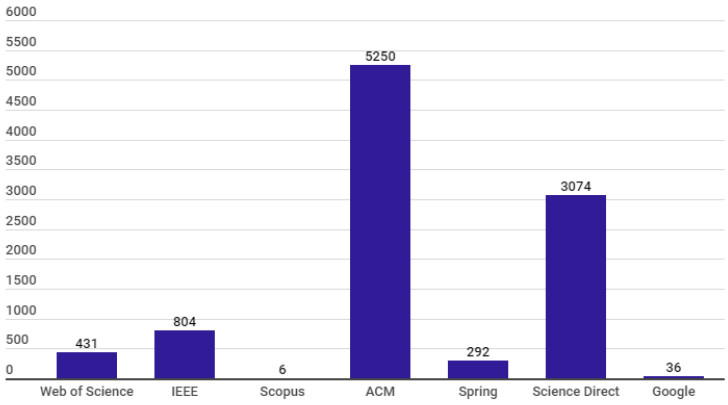
Para cada um dos repositórios selecionados aplicou-se a *string* de busca devidamente escrita em conformidade com suas peculiaridades/regras e fez-se uso da ferramenta de busca disponível em cada um. O resultados foram anexados em um mesma planilha a fim de serem posteriormente tratados.

[1] <www.scopus.com> Acesso em 03 de novembro de 2020
[2] <www.sciencedirect.com> Acesso em 03 de novembro de 2020
[3] <www.springerlink.com> Acesso em 03 de novembro de 2020
[4] <www.isiknowledge.com> Acesso em 03 de novembro de 2020
[5] <www.portal.acm.org> Acesso em 03 de novembro de 2020
[6] <www.ieeexplore.com.br> Acesso em 03 de novembro de 2020
[7] <scholar.google.com.br> Acesso em 03 de novembro de 2020

Inicialmente foram retornados 9857 estudos relacionados a cada base de busca conforme a Figura 3.

Inicialmente foram retornados 9857 estudos: 5250 da ACM Digital Library (ACM), 36 do Google Acadêmico (Google), 804 do IEEE Xplore (IEEE), 3074 do ScienceDirect (SD), 6 do Scopus (SC), 292 da Springer (SP) e, por último, 431 estudos da Web of Science (WoS).

Figura 3 – Distribuição dos documentos retornados inicialmente.

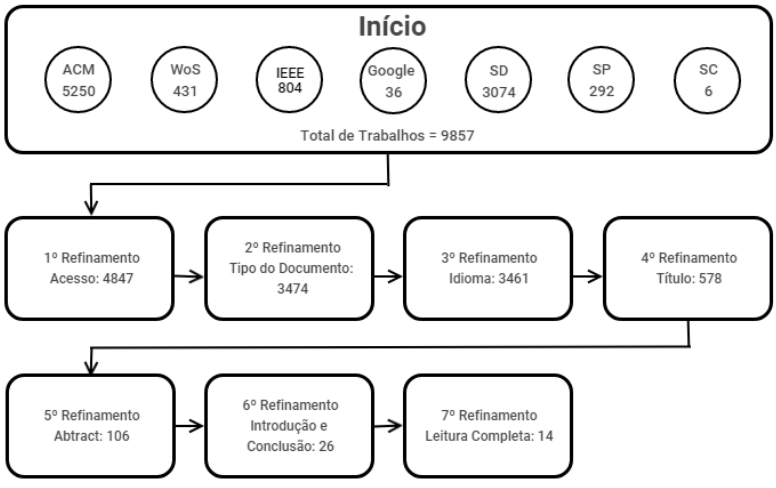


Fonte: Elaborado pela Autora

4.2.2 Processo de Seleção

Ao finalizar o processo de busca executou-se a seleção dos trabalhos a serem analisados aplicando os critérios de inclusão e exclusão pré estabelecidos no protocolo.

Figura 4 – Passos para a seleção dos trabalhos.



Fonte: Elaborado pela Autora

A Figura 4 ilustra as etapas que foram percorridas para a seleção dos trabalhos após a execução da *string* de busca nas sete bibliotecas digitais citadas anteriormente.

As buscas iniciais retornaram 9857 documentos e com o objetivo de selecionar um conjunto final de trabalhos foram executadas 7 etapas de refinamentos durante a fase de execução.

O 1º Refinamento aplicado teve por finalidade destacar apenas estudos acessíveis através das bases de dados conveniadas ao Instituto ou livres, resultando em 4847 documentos.

O 2º Refinamento descartou documentos cujo tipo de trabalho fugiam do escopo da revisão sistemática da literatura. Documentos classificados como capıtulos de livros, registro de patentes, tutoriais, *posters*, painéis, palestras, *workshops* ou outros foram desprezados. Após essa etapa, restaram 3474 estudos.

O 3º Refinamento excluiu documentos cujos tıtulos estavam grafados em idiomas diferentes da língua inglesa ou portuguesa e 21 artigos foram desconsiderados nesse refinamento, deixando 3461 para serem trabalhados. Vale ressaltar que, assim como esperado, não foram encontrados trabalhos dentro do escopo dessa revisão na literatura portuguesa.

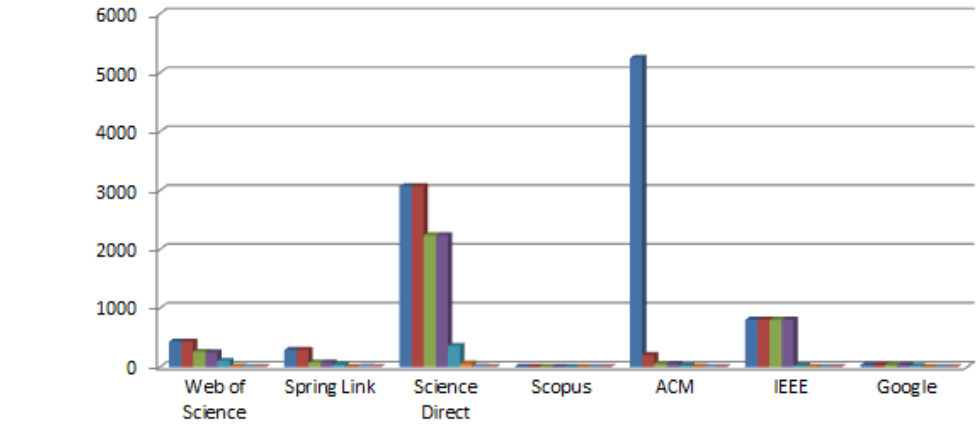
Como após a realização dos citados refinamentos a pesquisa ainda abrangia um grande número de artigos optou-se por acrescentar um 4º refinamento no qual analisou-se apenas o título de cada documento totalizando 578, tornando o número de artigos restantes mais plausível para a execução dos refinamentos mais complexos que vieram a seguir.

O 5º Refinamento baseou-se na análise do título e resumo dos estudos até este ponto selecionados, reduzindo o número de artigos para 106.

No 6º Refinamento, fez-se a análise dos artigos restantes de modo a verificar se atendiam ao propósito da presente revisão sistemática de literatura. Para tanto foram lidos introdução e conclusões dos artigos. Ao fim desse passo, restaram 26 artigos.

Posteriormente, realizou-se o 7º e derradeiro refinamento. Nessa etapa procedeu-se a leitura completa do artigo com o objetivo de confirmar que o artigo atendia aos propósitos desse trabalho. O conjunto resultante do processo do 7º refinamento foi composto de 14 artigos.

Figura 5 – Distribuição dos artigos após cada etapa de refinamento.



	Web of Science	Spring Link	Science Direct	Scopus	ACM	IEEE	Google
Inicial	431	292	3074	6	5250	804	36
1ºRefinamento	431	292	3074	6	204	804	36
2ºRefinamento	260	76	2237	4	57	804	36
3ºRefinamento	247	76	2237	4	57	804	36
4ºRefinamento	105	45	360	2	24	22	20
5ºRefinamento	11	0	60	0	8	6	2
6ºRefinamento	4	8	9	0	4	1	0
7ºRefinamento	3	5	4	0	1	1	0

Fonte: Elaborado pela Autora

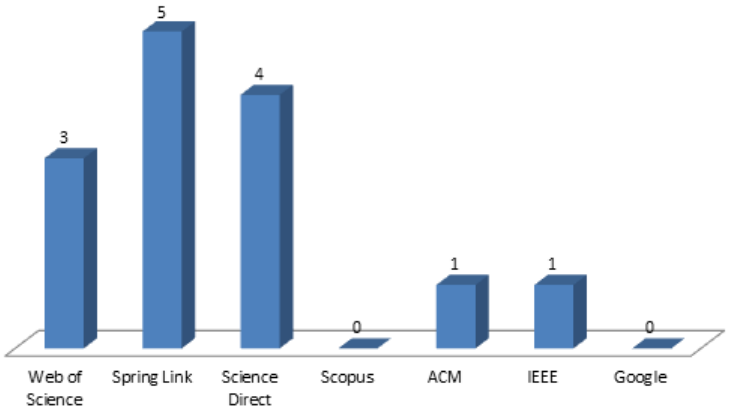
Figura 5 - número total de artigos selecionados após cada etapa de refinamento, agrupando os dados pelas fontes de buscas.

Durante todo o processo, à medida que encontrava-se estudos duplicados removia-os, assim totalizando 11 documentos excluídos. Para tal distinção trabalhos com títulos idênticos foram considerados iguais.

4.2.3 Resultado Final

A Figura 6 nos mostra a distribuição final dos artigos selecionados para análise entre as fontes de busca.

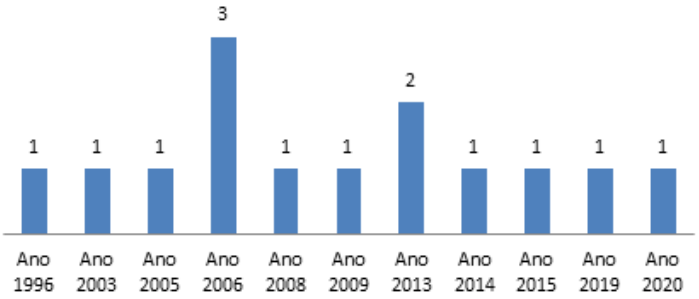
Figura 6 – Distribuição dos artigos selecionados ao final.



Fonte: Elaborado pela Autora

Levando-se em consideração os valores da Figura 6, nota-se que as três fontes de busca com maior contribuição, em número de artigos, após a última etapa de refinamento são: *Web of Science*, *Spring* e *Science Direct*.

Figura 7 – Distribuição dos artigos ao longo dos anos.



Fonte: Elaborado pela Autora

A Figura 7 revela como os artigos analisados foram distribuidos ao longo dos anos. O artigo mais antigo do conjunto selecionado é do ano de 1996 e o mais recente data de 2020. O ano com maior número de publicações selecionadas para análise ocorreu no ano de 2006 com um total de 3 artigos.

A Figura 8 relaciona os 14 artigos analisados neste trabalho.

Tabela 3 – Resultados Finais da Fase de Execução

Document Title	Publisher	Year
What Is Nature-Like Computation? A Behavioural Approach and a Notion of Programmability	Spring	2014
Coupling of quantum angular momenta: an insight into analogic/discrete and local/global models of computation	Spring	2006
Neural and Super-Turing Computing	Spring	2013
Computation and Hypercomputation	Spring	2003
Quantum information processing and communication	Spring	2005
Universality of quantum Turing machines with deterministic control	Web of Science	2015
Universality and programmability of quantum computers	Web of Science	2008
Algorithmic Complexity of Quantum States	Web of Science	2006
BIOCOMPUTATION: Some history and prospects	Science Direct	2013
What is a universal computing machine?	Science Direct	2009
Zeno machines and hypercomputation	Science Direct	2006
The simple dynamics of super Turing theories	Science Direct	1996
The Church-Turing Thesis: Logical Limit or Breachable Barrier?	ACM	2019
Theory of Quantum Computation with Magnetic Clusters	IEEE	2020

Fonte: Elaborado pela Autora

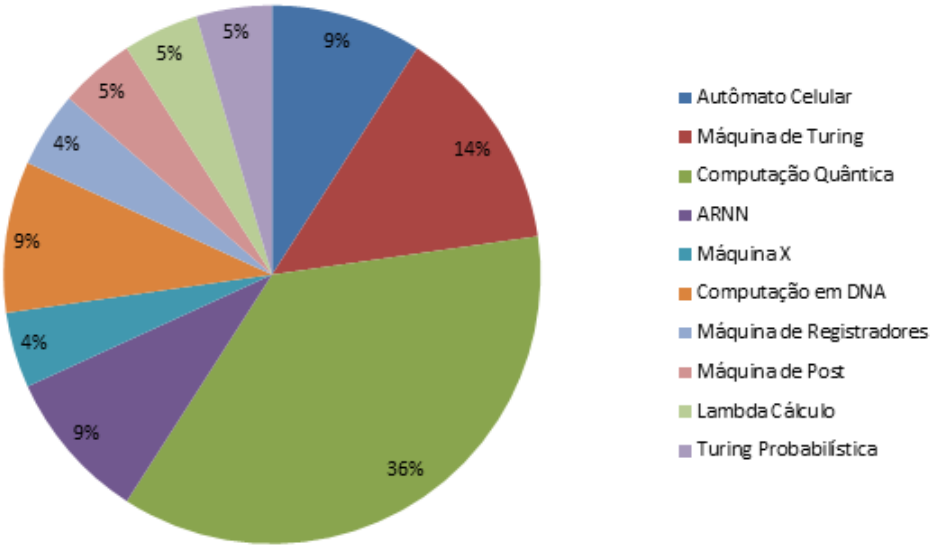
4.3 Fase de Análise

Nesta etapa, com os dados dos 14 artigos extraıdos, busca-se responder as questões desta pesquisa.

Q1: Quais são os modelos computacionais existentes?

No decorrer desta pesquisa foram retornados documentos da literatura que destacaram os modelos computacionais: Máquina de Turing e variações, Máquina de Turing Probabilística, Máquina de Post, Máquina de Registradores, Lambda Cálculo, Máquinas Quânticas, Autômato Celular, ARNN (Modelo de Rede Neural Recorrente Analógica) e Máquina X (Máquinas de Propósito Específico).

Figura 8 – Distribuição do Modelos Computacionais Encontrados nos Artigos.



Fonte: Elaborado pela Autora

Observando a Figura 8 percebe-se que o modelo mais abordado é a Computação Quântica sendo a abordagem presente em 9 artigos e representando 64% dos casos. O segundo modelo mais destacado foi a Máquina de Turing tratado em 3 artigos e representa 21% dos artigos analisados. Os demais modelos, como pode ser Figura 8, foram citados em 2 ou apenas 1 dos artigos trabalhados.

Q2: Qual modelo computacional mais viável (custo computacional x poder computacional e possibilidade de desenvolvimento em larga escala) ?

Como bem colocado no artigo (ZENIL, 2014)

> *"Um sistema pode ser capaz de realizar computações universais porém sua programação pode ser realmente difícil. Isso significa que existe uma diferença entre o que pode ser alcançado em princípio e a capacidade prática de um sistema para a execução de uma tarefa. Essa abordagem é, portanto, sensível aos aspectos práticos da programação de um sistema e não apenas a sua capacidade teórica de ser programado."*

sendo assim o custo computacional pode ser agregado do custo de se programar a máquina e não apenas os recursos necessários.

Os artigos analisados não forneceram métricas para que ocorresse uma comparação, mas destacaram alguns pontos fortes e fracos dos modelos abordados.

Segundo (STANNETT, 2003) Máquina X é um modelo teórico que permite comportamento de modelos complexos serem modelados em termos de máquinas teóricas bastante

simples, originalmente baseada em Máquinas de Estados Finitos (FSM), sendo considerada o modelo de computação mais simples e menos poderoso, mas cujo poder computacional é potencialmente maior que o da Máquina de Turing.

O ARNN trata-se de um modelo biológico analógico, e como descrito nos artigos (SIEGELMANN, 2003) e (CULL, 2013) calcula em tempo polinomial funções da classe Super Turing Polinomial não Uniforme P/Poly e é considerado universal para computação com memória finita contudo, tem como ponto negativo a limitação para funções de limiar linear.

Computação em DNA poderia resolver problemas NP Completos devido ao seu paralelismo químico, que possibilita a geração de todas as possíveis soluções de um problema simultaneamente, o resolvendo mais rapidamente que um computador sequencial. Porém, ambos os artigos em que é menciona apresentam preocupação em relação a sua manipulação e aplicação a grande problemas. A manipulação pode ser mais demorada e difícil visto que os experimentos em laboratórios exigiram mais tempo e pessoal de laboratório especializado em DNA. Outro contraponto seria não poder ser ampliada a problemas grandes, pois a quantidade de material necessário para os experimentos varia, pelo menos, exponencialmente com o tamanho do problema.

Sistemas quântico são vistos como a grande aposta do futuro, em todos os artigos em que é menciona é falado de sua eficiência associado a computadores quânticos.

Segundo os estudos, um computador quântico será capaz de executar tarefas intratáveis para hardware não quânticos, como tarefas que necessitam de ser executadas várias vezes em vários processadores discretos trabalhando em paralelo, o que acarreta impacto decisivo no tempo de execução e requerimento de memória.

Contudo, sistemas quânticos são muito difíceis de simular, devido ao fato de que a dimensão do espaço de Hilbert cresce exponencialmente com o número de partículas, prejudicando nossa capacidade de entender as propriedades físicas em um computador clássico. Sendo assim a grande questão deste modelo é o computador quântico, seu desenvolvimento proporcionará a base necessária para que a computação quântica aconteça de vez.

O Modelo de Turing é o modelo utilizado atualmente, tendo sido amplamente explorado. Sendo assim as novas conquista viriam de provar que NP é igual a P.

Os outros modelos citados pelos artigos não foram amplamente discutidos o que mostra que esses modelos não são atrativos no cenário atual como os outros.

Após todas as informações apresentadas pelos artigos foi possível concluir que os modelos que estão sendo visto como mais atrativos atualmente são a Computação Quântica e a Computação em DNA, mas infelizmente nenhum fornece informações concretas para afirmar que um seja mais viável do que o outro.

Em relação qual o melhor modelo, essa é uma afirmação que provavelmente nenhum cientista irá fazer, pois não existe um modelo melhor que o outro, existem situações em que cada um oferece mais vantagens e/ou desvantagens ao ser empregado.

Q3: Qual o modelo computacional mais didático?

Segundo (STANNETT, 2003) A Máquina X, um modelo teórico que permite comportamento de modelos complexos serem modelados em termos de máquinas teóricas bastante simples. Trata-se um modelo elegante que possui uma estrutura de controle um FSM, o que lhe atribui fácil compreensão e manipulação, tornando o didático.

4.4 Ameaças à Validade

Apesar da realização de um criterioso planejamento, muitos fatores podem afetar os resultados desse estudo e invalidar as respostas encontradas. Verifica-se abaixo as estratégias seguidas para minimizar o impacto desses fatores na validade dos resultados, sendo algumas dessas atitudes aplicadas a fim de aumentar a confiabilidade do trabalho.

Escopo e Estratégia - Para essa Revisão Sistemática da Literatura, foram selecionadas sete bibliotecas digitais diferentes, porém é perfeitamente possível que existam outras que foram desprezadas. Uma das fontes de buscas utilizadas foi o Google Acadêmico que realiza a busca em diversos repositórios de publicações, essa medida foi tomada para promover maior segurança quanto a abrangência da pesquisa.

Validação e Generalização dos Dados e Resultados - A *string* de busca foi cuidadosamente definida para garantir o maior retorno possıvel de resultados relevantes para a presente pesquisa. Sendo assim, foi inicialmente pensada a partir das palavras chaves e seus principais sinônimos. E após algumas pesquisas de teste e alterações criteriosas chegou-se a *string* de busca utilizada no trabalho.

5 MODELOS COMPUTACIONAIS

Considerando os resultados da Revisão Sistemática da Literatura selecionou-se dos modelos descritos nos resultados alguns para mostrar em maiores detalhes afim de deixar esses modelos aqui registrados para auxiliar em futuras pesquisas da área.

5.1 ARNN

Proposto por Eduardo Sontaq, matemático e teórico de controle da Universidade Rutqers, o modelo de Rede Neural Recorrente Analógica (ARNN) é um conjunto finito de ligações simples (ou neurônios), sendo popular, na prática, como uma máquina com aprendizagem automática e capacidade de adaptação (SIEGELMANN, 2003).

Trata-se de um modelo matematicamente simples, porém biologicamente relevante (SIEGELMANN, 2003).

A rede é dita analógica devido ao fato de seus neurônios se atualizam dentro de um espaço de fase contínuo, com continuidade local e incorpora pesos reais. O adjetivo "Recorrente" enfatiza que a interconexão é geral, e não em camadas ou simétrico (SIEGELMANN, 2003).

Cada neurônio faz parte da unidade de processamento e a memória é implicitamente codificada na influência mútua entre qualquer par de neurônios. Essa influência é representada por um peso numérico real (SIEGELMANN, 2003).

Quando os pesos são parâmetros desconhecidos, a rede é um sistema adaptativo, tecnologia que aproxima os mapeamentos de entrada e saída através de parâmetros (aprendizado). Quando os pesos são considerados constantes as redes podem executar cálculos exatos em vez de meras aproximações (SIEGELMANN, 2003).

Formalmente, o ARNN consiste em um número finito de neurônios simples em tempo discreto. Existem dois canais de entrada nos quais as entradas são locais. Cada neurônio atualiza seu valor de ativação (estado) de acordo com uma função das ativações(x_j) , entradas (u_j) e um conjunto de coeficientes / pesos reais (a_ij ,b_ij,c_i) (SIEGELMANN, 2003)(SIEGELMANN, 1996):

$$x_i\ (t+1) = \sigma\ (\ \Sigma_{j=1}^{N}\ a_ij\ x_j\ (t) + \Sigma_{j=1}^{M}\ b_ij\ u_j\ (t) + c_j\ )\ , i = 1, ..., N,$$

onde N é o número fixo de neurônios, M é o número de sinais de entrada externa, x_j são as ativações dos neurônios, u_j são as entradas externas e a_ij,b_ij c_i são as reais

coeficientes, também chamados de constantes ou pesos e uma função "sigmoide"muito simples, uma função linear saturada:

- $\sigma := 0 if x < 0$,
- $\sigma := x if 0 \leq x \leq 1$,
- $\sigma := 1 if x > 1$.

Um subconjunto de N neurônios é escolhido para comunicar a saída da rede para o ambiente.

Entradas e processos são fluxos de letras, e a computabilidade é definida sob a convenção que às vezes é usada em redes de comunicação práticas: há são dois canais de entrada binários, onde um é usado para transportar o sinal de entrada binário, e o outro índica quando uma entrada está ativa. Uma convenção semelhante é aplicada ao resultado (SIEGELMANN, 1996).

Uma rede é especificada por seus pesos e conjunto de definição de saída. A estrutura da rede, incluindo os valores dos pesos de interconexão permanecem constantes. As alterações são os valores de ativação usado na próxima iteração. Nesse sentido, o modelo é "uniforme"(SIEGELMANN, 2003).

5.2 Autômato Celular

O estudo dos autômatos celulares despertou um grande interesse em razão da sua habilidade de gerar um amplo espectro de padrões comportamentais complexos a partir de conjuntos de regras relativamente simples e sua capacidade de gerar simulações, previsões e resultados não conseguidos utilizando outros métodos como as equações diferenciais. Outra característica interessante é que eles demonstram capturar a essência de comportamentos auto-organizados complexos observados em sistemas naturais (GREMONINI, 2020).

Os modelos naturais podem ser representados por sistemas evolutivos que partindo de uma configuração inicial aleatória, cada componente desse sistema tem sua evolução baseada na situação atual dos seus vizinhos e em um mesmo conjunto de regras para todos os componentes que compõem esse sistema. Apesar de todos os componentes sofrerem ação de um mesmo conjunto de regras, suas situações podem variar indefinidamente e complexamente durante o tempo, podendo assim, originar novos sistemas e chegando até mesmo a sua autorreprodução (CASTRO; CASTRO, 2008).

Os autômatos celulares são ferramentas simples e poderosas para representar sistemas físicos compostos por elementos discretos com interações locais. Em outras palavras, são modelos matemáticos discretos no tempo, no espaço e nas variáveis dinâmicas, e têm

sua evolução regida por regras simples. São compostos por unidades simples (chamadas de células) que interagem umas com as outras, ou seja, influenciando mutuamente seus comportamentos, e à medida que o sistema dinamicamente evolui, surgem comportamentos complexos decorrentes dessas mútuas influências, sendo assim sendo essa uma outra característica importante desses tipos de sistemas complexos (GREMONINI, 2020).

Dessa forma, ao se assumir que cada célula é um autômato, pode-se classificar um autômato celular como um conjunto de autômatos igualmente programados que interagem entre si (GREMONINI, 2020).

O conceito de Autômato Celular foi proposto por John Von Neumann e Stanislaw Ulam no início dos anos 50. Von Neumann mostrou que um autômato celular pode ser universal, ou seja, computacionalmente completo, no entanto, as regras utilizadas para essa demonstração nunca foram implementadas em um computador devido a sua grande complexidade (GREMONINI, 2020).

Um autômato celular pode ser definido, segundo (CASTRO; CASTRO, 2008) formalmente como uma quádrupla:

$$(\mathrm{L},\ \mathrm{S},\ \mathrm{N},\ \mathrm{f}\ )$$

- L uma grade regular (os elementos de L são chamados de células),
- S um conjunto finito de estados,
- N um conjunto finito, (de tamanho n = |N|) de vizinhança, tal que $\forall c \in \mathbf{N}, \forall r \in \mathbf{L}$: $r + c \in \mathbf{L}$,
- $f : \mathbf{S}^X \longrightarrow S$ uma função de transição.

Simplificando, uma grade regular pode ser definida como sendo "ladrilho" de dimensão d e tamanho finito, isto é, as células unidas preenchem totalmente um espaço de dimensão d e a grade pode ser transladada em d direções independentes para obtê-la (CASTRO; CASTRO, 2008).

Uma configuração $C_t : L \longrightarrow$ S é uma função que atribui um estado a uma célula na grade. O papel desempenhado pela função de transição f é troca de uma configuração C_t por uma nova configuração C_{t+1} dada pela função (CASTRO; CASTRO, 2008):

$$C_{t+1}(r) = f(\{C_t(i) | i \in N(r)\}),$$

onde N(r) é o conjunto das células vizinhas de r, de forma que:

$$\mathrm{N(r)} = \{i \in L \mid \mathrm{r} - \mathrm{i} \in N\}$$

A fim de deixar o conceito apresentado mais claro segue-se uma definição informal.

Um autômato celular pode ser caracterizado pelas seguintes propriedades fundamentais segundo (CASTRO; CASTRO, 2008):

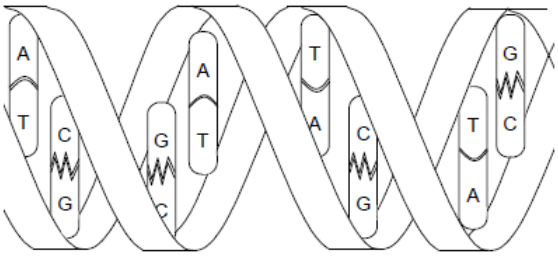
- Consistem em uma grade ou matriz de células ;
- A evolução ocorre em passos discretos de tempo;
- Cada célula é associada a um estado pertencente a um conjunto finito de estados;
- Cada célula evolui de acordo com as mesmas regras que dependem somente do estado em que a célula se encontra e de um número finito de vizinhos;
- A relação com a vizinhança é local e uniforme.

5.3 Computação em DNA

A computação com DNA é um dos novos desafios para a Ciência da Computação pertencendo à uma linha de pesquisa mais ampla denominada de computação molecular. Baseada na manipulação genética de fitas de DNA, esse modelo computacional, através de um conjunto de ações microbiológicas, codifica os dados estruturados de um problema em sequências de nucleotídeos (fitas de DNA) e computa uma solução manipulando essas estruturas (FILHO, 2004).

Esse modelo surgiu como uma proposta para utilizar o grande poder do paralelismo das reações moleculares na solução de problemas ainda não resolvidos satisfatoriamente pela computação convencional (FILHO, 2004). Essa forma massivamente paralela deve-se ao fato de utilizar DNA como estrutura de dados, assim tendo um alfabeto quaternário A - Adenosina,C - Citosina,T - Timina,G - Guanina em vez de binário 0,1 (ROCHA et al., 2012).

Figura 9 – Modelo de Molécula de DNA



Fonte: (ZUBEN; ATTUX, 2003)

Os computadores de DNA funcionam através da biologia molecular, diferentemente da tecnologia convencional, que opera através de processadores de silício. O DNA propriamente dito atua como um software e as enzimas como o hardware. Contrapondo os computadores tradicionais, não utilizam impulsos elétricos e sim reações químicas, por essa razão demandam muito pouca energia e são altamente econômicos na armazenagem de informação (ANTONIO; JUNIOR, 2011).

Os computadores de DNA surgiram da necessidade de solucionar problemas combinatórios com mais eficiência que um computador convencional, se mostrando bem sucedido no experimento realizado por Leonard Adleman em 1994, sendo esta a primeira utilização da computação com DNA de sucesso na história. Adleman empregou técnicas de manipulação de moléculas de DNA para solucionar uma pequena instância do caminho Hamiltoniano, um conhecido problema da classe NP-Completo. A vantagem dos computadores de DNA em relação aos computadores convencionais (de processadores de silício) na resolução dessa classe de problemas advém justamente do fato dele poder testar as possibilidades simultaneamente enquanto o tradicional tem que checar uma a uma (ANTONIO; JUNIOR, 2011).

A vantagem desse modelo advém de sua capacidade de gerar todas as possíveis soluções de um problema de forma simultânea devido ao paralelismo químico. Todavia, a lenta velocidade de processamento por se tratar de um experimento laboratorial e a preocupação com a aplicação a grandes problemas (a quantidade de partículas necessária aumenta, pelo menos, à uma taxa exponencial ao tamanho do problema) são pontos que desfavorecem a utilização desse modelo(ZUBEN; ATTUX, 2003).

5.4 Computação Quântica

Pode-se considerar como o marco inicial da computação quântica o ano de 1981, no qual o físico Richard Philips Feynman (1918 – 1988) elaborou a primeira proposta de um dispositivo que executasse rotinas computacionais através da utilização de fenômenos quânticos (SILVA, 2018).

Foi argumentado por Feynman que computadores convencionais podem simular a física clássica, diferente do que ocorre com a física quântica, uma vez que a dimensão do espaço nela tratado cresce exponencialmente em função do número de partículas acrescentadas ao sistema. Feynman então, trouxe a tona a dúvida se não seria possível simular eficientemente a mecânica quântica através de um dispositivo que usasse suas leis para realizar cálculos (SILVA, 2018).

Alguns anos depois este pensamento de Feynman foi formalizado por David Deutsch, que definiu modelos de computação baseados na mecânica quântica. Primeiramente, Deutsch generalizou o modelo de MTs através de um modelo de Máquina de Turing

Quânticas (MTQs), para em seguida formular o modelo de circuítos lógicos através de um modelo de Circuitos Quânticos (CQs). Contudo, foi Andrew Yao que demonstrou que qualquer função computável em uma MTQ em tempo polinomial pode ser computada por um CQ de tamanho polinomial, legitimando a equivalência entre os dois modelos (CARLOS et al., 2009).

5.4.1 QUBIT

Um bit quântico, ou qubit, consiste em um vetor unitário pertencente a um espaço vetorial complexo de duas dimensões com uma base particular fixa, denotada por |0>,|1>. Esses estados correspondem aos bits clássicos 0 e 1. Contudo, ao contrário dos bits clássicos, um qubit pode estar em uma superposição dos dois estados denotada por $|\psi = a\ |0> + b\ |1>$ (CARDONHA; SILVA; FERNANDES, 2004).

Apesar de um q-bit poder estar em infinitos estados - isso ocorre variando-se os valores de a e b - só é possível obter um bit de informação dele, pois ao se realizar uma medição, o seu estado se colapsará em somente |0> ou |1> (CARDONHA; SILVA; FERNANDES, 2004).

O grande poder dos computadores quânticos advém de um fenômeno denominado entrelaçamento. Este ocorre quando se opera com múltiplos q-bits simultaneamente (CARDONHA; SILVA; FERNANDES, 2004).

De forma geral, n q-bits podem estar em superposições de 2^n estados base, o que possibilita o computador quântico realizar múltiplas computações simultaneamente e com fator exponencial. Entretanto, é necessário pontuar que é possível ler apenas um desses valores, sendo sempre fornecidos n bits de informação. Este fato aparenta inutilizar a capacidade de um computador quântico, contudo é possível usar alguma propriedade coletiva de todos os estados para se realizar uma computação útil em um tempo inferior ao de um computador tradicional (CARDONHA; SILVA; FERNANDES, 2004).

5.4.2 Máquina de Turing Quântica

Em uma Máquina de *Turing* Quântica (MTQ) as funções são realizadas através de interações quânticas. A fita e a cabeça existem em um estado quântico contudo, no lugar da célula a MTQ abriga os q-bits, que apresentam estados de superposição de 0 ou 1. Outra característica importante da MTQ é o poder de codificar múltiplas entradas para um problema e calculá-las simultaneamente (SILVA, 2018).

5.4.3 Definição Formal

A formalização de um modelo é uma ferramenta imprescindível na especificação do que se é possível ou não de ser computado por ele. Todavia, a definição formal de uma

MTQ não é um conceito trivial de ser compreendido (GUEDES; JR, 2004).

Uma Máquina de Turing Quântica pode ser definida de acordo com (GUEDES; JR, 2004) por uma 6-tupla ordenada $(\Sigma, \mathbf{\Lambda}, Q, q_i, q_f, \delta)$, onde:

1. Σ é o alfabeto: um conjunto finito de todos os possíveis símbolos da fita;
2. $\mathbf{\Lambda} \in \Sigma$ é o símbolo que representa branco;
3. Q é um conjunto finito denominado conjunto de estados;
4. q_i é o estado inicial;
5. q_f é o estado final;
6. $\delta : \Sigma x Q \longrightarrow$ H é a função de transição onde H é um espaço de Hilbert estendido por vetores da base correspondentes à tripla de Σ x Q x {L,R}, e as correspondentes transições finitas da matriz unitária para todos os comprimentos de entrada.

O estado de uma MTQ pode ser visto como uma superposição de configurações de uma MT clássica. Em um instante de tempo t, o estado de uma MTQ M é representado por um vetor unitário $|\psi(\mathrm{t})\rangle$, no espaço de Hilbert H(Q,Σ) gerado pelo espaço de configurações $C(Q, \Sigma) = \mathrm{Qx}\Sigma^{\#}\mathrm{xZ}$, onde $\Sigma^{\#}$ é o conjunto de todas as possíveis configurações de fita T. Assim sendo, cada elemento $|\ q_i$, T, n$>$ de C(Q,Σ) representa uma possível configuração de uma MT tradicional, onde q_i corresponde ao estado atual, T ao conteúdo da fita e n à posição atual da máquina (CARLOS et al., 2009).

Com essa correlação explicitada é possível então interpretar os estados superpostos das MTQs como a coexistência de configurações de MTs clássicas, e o paralelismo quântico como a execução simultânea das diferentes possíveis computações (CARLOS et al., 2009).

É importante indicar que, segundo Deutsch, não é viável explicar a computação quântica através de qualquer outra interpretação da mecânica quântica sem assim perder completamente o poder elucidativo (CARLOS et al., 2009).

Deutsch também define, assim como Turing fez para sua máquina clássica, condições de finitude em sua formalização do modelo de MTQ. Assim, as operações de uma MTQ são restritas pelas seguintes condições:

1. Durante cada passo de computação somente uma parte finita do sistema pode mudar;
2. A mudança deve depender somente do estado de um subsistema finito;
3. As regras que determinam a mudança devem ser dadas finitamente.

5.5 *Lambda* Cálculo

O lógico americano Alonzo Church criou o *Lambda* Cálculo na década 1930, antes mesmo dos computadores estarem disponíveis. Esse cálculo explora teoricamente a base do que significa computação (MUELLER, 2019).

A importância do *Lambda* Cálculo ultrapassa o estudo da computabilidade, tendo importantes aplicações no estudo das linguagens de programação e da lógica (DIVERIO; MENEZES, 2009).

Segundo a visão de Church, os objetivos do *Lambda* Cálculo englobam estudar a interação da abstração funcional e a aplicação da função a partir de uma perspectiva abstrata e puramente matemática (MUELLER, 2019).

O *Lambda* Cálculo usa apenas funções, sendo qualquer outro tipo encontrado nas linguagens de programação atuais, como *string*, inteiro e booleano, é codificado como parte de uma função, portanto a função é a base de tudo (MUELLER, 2019).

(DIVERIO; MENEZES, 2009) define formalmente o cálculo lambda como mostrado a seguir:

O *Lambda* Cálculo busca uma noção de igualdade, que tem como finalidade determinar quando dois lambda-termos denotam a mesma função.

Consiste na linguagem *Lambda* e esta é munida de axiomas e as regras de inferência de acordo com o descrito abaixo:

Sejam M, N e $K\lambda$-termos:

1. Principal axioma:

$$(\lambda x.M)N = M[x \longleftarrow N]$$

2. Axiomas e regras lógicas:

   a) Igualdade

   - Reflexiva: M=M;
   - Simétrica: se M = N, então N = M;
   - Transitiva: se M = N e N = K, então M = K

   b) Regras de compatibilidade

   - Distributiva à Esquerda: se M = N, então K N = K M
   - Distributiva à Direita: se M = N, então N K = M K
   - Se $M = N$, então $\lambda x.N = \lambda x.M$

3. Os termos M e N são iguais no *Lambda* Cálculo, ou seja, M = N se, e somente se, existe uma dedução de M = N no λ-cálculo.

Assim sendo, o principal axioma formaliza que o resultado de uma aplicação é calculado por substituição.

É importante perceber que, a partir da notação *Lambda*, origina-se tanto uma linguagem quanto um cálculo. O cálculo tem por objetivo verificar a equidade de termos da linguagem. A noção de cálculo é explanada pelo conceito de redução, o qual pode ser entendido como um "passo computacional"na busca de um "valor"representativo para um λ-termo qualquer. Portanto, a redução pode ser interpretada como uma operacionalização do *Lambda* Cálculo. São elas:

- Redução *alfa*: renomeação de variáveis ligadas;
- Redução *beta*: aplicação de uma função a um argumento, via substituição;
- Redução *iterada*: sucessiva aplicação de qualquer das reduções acima.

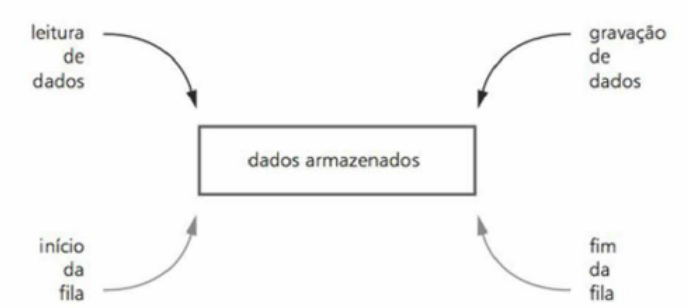
5.6 Máquina de Post

Em meados dos anos 1930, Emil Leon Post propôs modelo de máquina universal denominado Máquina de Post (DIVERIO; MENEZES, 2009).

Como sua principal característica, a Máquina de Post apresenta o uso de uma estrutura de dados do tipo fila para a entrada, saída e memória de trabalho como representado na Figura 10 (DIVERIO; MENEZES, 2009).

Estruturalmente, a característica essencial de uma fila é que o primeiro valor gravado é também o primeiro a ser lido e com isso excluído, visto que uma leitura exclui o dado lido (DIVERIO; MENEZES, 2009).

Figura 10 – Exemplo da Estrutura de Fila



Fonte: (DIVERIO; MENEZES, 2009)

Pode-se classificar uma Máquina de Post em, basicamente, duas partes: variável x e programa.

Variável X

- Se trata de uma variável a qual é atribuída o tipo de estrutura da máquina, a fila, e assim sendo é utilizada como entrada, saída e memória de trabalho;
- não é restringida nem em tamanho e nem em limites fixos. Seu comprimento assume o tamanho da palavra corrente armazenada;
- os símbolos podem pertencer a # - único símbolo auxiliar - ou ao alfabeto de entrada;
- inicialmente, seu valor é a palavra de entrada;
- no caso de entrada vazia (representada por ϵ) X não possui nenhum símbolo. (caso X não possua nenhum símbolos a entrada é vazia (ϵ).)

Programa

- Pode ser entendido como uma sequência de instruções, as quais são representadas como um fluxograma, mais especificamente um diagrama de fluxos, onde cada vértice é uma instrução, e esta pode ser classificada em quatro tipos:
  - Partida
  - Parada
  - Desvio
  - Atribuição

Definição formal de Máquina de Post encontrada em (DIVERIO; MENEZES, 2009)

Uma máquina de Post é uma tripla M = (Σ, D, #) onde:

- Σ: alfabeto de símbolos de entrada;
- D: programa representado em diagrama de fluxos construído a partir dos quatro tipos de componentes elementares: partida, parada, desvio e atribuição;
- #: símbolo auxiliar.

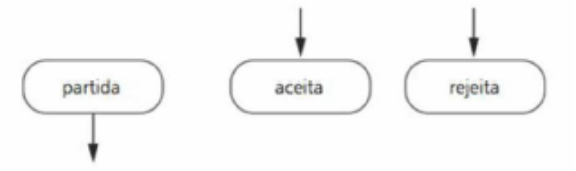
Os quatro tipos de componentes elementares de um diagrama de fluxo podem ser definidos como:

a) Partida: Existe somente uma instrução desse tipo - instrução de início - em um programa;

b) Parada: Existem apenas duas alternativas de instruções de parada em um programa: uma de aceitação (aceita) e outra de rejeição (rejeita);

A Figura 11 trás um diagrama exemplificando esse mecanismo de saída e parada.

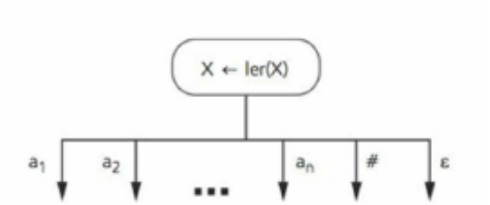
Figura 11 – Diagrama exemplificando a partida e a parada



Fonte: (DIVERIO; MENEZES, 2009)

c) Desvio ou teste: Estabelece o fluxo do programa segundo o símbolo mais à esquerda da palavra armazenada na variável X (início da fila). Deve ser antevisto a possibilidade de X conter a palavra vazia. Assim sendo, é um desvio condicional, e se trata de uma função total, isto é, definida para todos os valores do domínio. Assim, caso o cardinal de Σ for n, então deverão existir $n + 2$ arestas de desvios condicionais, pois se deve incluir as possibilidades # e ε, onde $X \longleftarrow ler(X)$ denota uma leitura destrutiva, explanando melhor, que lê o símbolo mais à esquerda da palavra o retirando.

Figura 12 – Exemplo de desvio em um diagrama de fluxos.



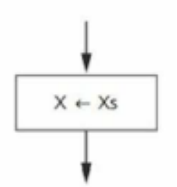
Fonte: (DIVERIO; MENEZES, 2009)

d)Atribuição: Concatena o símbolo indicado (pertencente a $\Sigma \cup \{\#\}$) ao final da fila, ou seja, à direita da palavra armazenada na variável X . A operação de atribuição é representada como ilustrado na figura 0, supondo que $s \in \Sigma \cup \{\#\}$.

Observações e Funcionamento do Diagrama de Fluxo

Em um diagrama de fluxos, existe apenas uma única instrução de partida, porém, podem existir várias (zero ou mais) instruções de parada, tanto de aceitação como de rejeição, tornando assim possível uma máquina de Post entrar em *loop* infinito. Uma palavra de entrada é dita como aceita ou não (ou seja, rejeitada) se a computação, inicializada com a variável X, contendo a entrada, alcançar uma instrução aceita ou rejeita, respectivamente.

Figura 13 – Exemplo de atribuição em um diagrama de fluxos.



Fonte: (DIVERIO; MENEZES, 2009)

Quando ocorre um desvio X é verificado, caso ele contenha a palavra vazia E segue-se o fluxo correspondente. Caso contrário, lê-se o símbolo mais à esquerda da palavra em X e o remove após decidir qual aresta do fluxo seguir para a próxima instrução.

5.7 Máquina de Registradores

As Máquinas de Registradores são modelos mais recentes quando comparados a máquina de Turing e outros formalismos conhecidos. Richard Bird propôs, em 1976, a Máquina Norma - *Number Theoretic Register* - uma máquina de registradores especialmente interessante, pois distingue as noções de programa e máquina (DIVERIO; MENEZES, 2009).

A máquina Norma possui um conjunto infinito - acordo com a necessidade - de registradores naturais (que podem assumir qualquer valor natural sem limite de tamanho) como memória e três instruções sobre os registradores:

- adição do valor um;
- subtração do valor um;
- teste do valor armazenado igual a zero.

Definição Formal da Máquina Norma de acordo com (DIVERIO; MENEZES, 2009)

A máquina Norma é uma sétupla (suponha que $K \in X, Y, A, B, \ldots$):

$Norma = (N^{\infty}, N, N, ent, sai, ad_K, sub_K, zero_K)$

onde:

a) Cada elemento do conjunto de valores de memória N^{∞} indica uma configuração de seus infinitos registradores, os quais são representadas por:

X, Y, A, B, ...

N^{∞} denota o conjunto de todas as tuplas com infinitos componentes sobre o conjunto dos números naturais. Por exemplo:

(0, 1, 2, 3, ...) e (5, 5, 5, 5, ...)

A definição formal de N^∞ contida em (DIVERIO; MENEZES, 2009) é:

- para cada $n \in N$, a_n é a n-ésima componente da upla (a0, a1, a2, ...);
- $f : N \longrightarrow N$ tal que, para qualquer $n \in N, f(n) = a_n$;
- um elemento de N^∞ pode ser visto como uma função nos naturais, ou seja: $N^\infty = \{f : N \longrightarrow N | f \text{ é função } \}$

b) A função de entrada carrega no registrador X o valor de entrada, inicializando todos os demais registradores com zero:

$ent : N \longrightarrow N^\infty$

c) A função de saída retorna o valor atual do registrador Y:

$sai : N^\infty \longrightarrow N$

d) O conjunto de operações é uma família indexada pelos registradores onde, para cada registrador K, tem-se que:

Adição: acrescenta um ao componente de memória correspondente ao registrador K, e os demais mantêm seus valores.

$ad_K : N^\infty \longrightarrow N^\infty$

Subtração: retira um do componente de memória correspondente ao registrador K, caso seu valor for maior que zero (senão, mantém o valor como zero), deixando os outros com seus valores inalterados.

$sub_K : N^\infty \longrightarrow N^\infty$

e) O conjunto de interpretações de testes é indexado pelos registradores onde, tal que para cada registrador K tem-se:

$zero_K : N^\infty \longrightarrow$ verdadeiro, falso

retornando verdadeiro se o componente de memória correspondente ao registrador K for igual a zero , caso contrário, retorna em falso.

Por simplicidade, para um registrador K as operações $ad_K, sub_K, zero_K$ são denotadas, respectivamente, por:

$$K := K + 1$$
$$K := K - 1$$
$$K = 0$$

A máquina Norma é uma máquina extremamente simples, de tal forma que parece difícil acreditar que o seu poder computacional é, no mínimo, o de qualquer computador moderno, porém, diversas características de máquinas reais podem ser simuladas usando a máquina Norma, corroborando com o fato de se tratar de uma máquina universal (DIVERIO; MENEZES, 2009).

5.8 Máquina de Turing

Em 1936, com a apenas 24 anos, Alan M. Turing consagrou-se como um dos maiores matemáticos do seu tempo quando fez antever que era possível executar operações computacionais sobre a teoria dos números por meio de uma máquina que tivesse embutidas as regras de um sistema formal (SOLTEIRA, 2017).

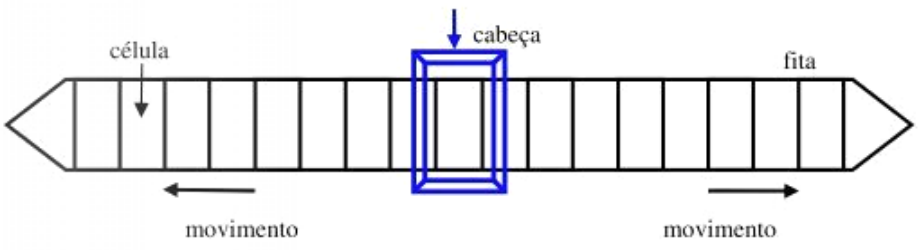
Trata-se de um modelo simples e poderoso e que formaliza a ideia de uma pessoa que realiza cálculos. É muito similar a um computador moderno, embora tenha sido proposto alguns anos antes do primeiro computador digital. Na realidade, não se trata de uma máquina no sentido usual dado a essa palavra, mas sim de um programa para uma máquina universal (DIVERIO; MENEZES, 2009).

Trata-se de um modelo computacional semelhante á um autômato finito, no entanto, a Máquina de Turing (MT) tem propósito de uso geral e apresenta um maior poder computacional (AUGUSTO; FILHO,).

Sendo um modelo abstrato de computador formaliza apenas os aspectos lógicos do seu funcionamento, como a capacidade de armazenamento, estados e transições, não se atendo a nenhum aspecto de sua implementação física (AUGUSTO; FILHO,).

As MTs são constituídas por um conjunto finito de estados, uma fita dividida em células e uma cabeça de leitura/escrita (RW) que atua sobre apenas uma célula de cada vez (MOREIRA, Nelma; MATOS, 2001).

Figura 14 – Diagrama de uma Máquina de Turing.



Fonte: (MOREIRA, Nelma; MATOS, 2001)

a) Fita: Usada simultaneamente como dispositivo de entrada, de saída e de memória de trabalho;

b) Unidade de controle: Reflete o estado corrente da máquina. Possui uma unidade de leitura e gravação (cabeça da fita), a qual acessa uma célula da fita de cada vez e movimenta-se para a esquerda ou para a direita;

c)Programa ou função de transição. Função que define o estado da máquina e comanda as leituras, as gravações e o sentido de movimento da cabeça.

Adota-se que a fita é finita à esquerda e infinita (tão grande quanto necessário) à direita e é dividida em células, cada uma armazenando um símbolo. Os símbolos podem pertencer ao alfabeto de entrada, ao alfabeto auxiliar ou ainda pode ser branco ou marcador de início de fita.

Inicialmente, a palavra a ser processada (ou seja, a informação de entrada para a máquina) ocupa as células mais à esquerda, após o marcador de início de fita, ficando as demais com branco.

A unidade de controle possui um número finito e predefinido de estados.

A cabeça da fita lê o símbolo de uma célula de cada vez e grava um novo símbolo. Após a leitura/gravação (a gravação é realizada na mesma célula de leitura), a cabeça move uma célula para a direita ou para a esquerda. O símbolo gravado e o sentido do movimento são definidos pelo programa.

O programa é uma função que, dependendo do estado corrente da máquina e do símbolo lido, determina o símbolo a ser gravado, o sentido do movimento da cabeça e o novo estado.

Formalmente, segundo (MOREIRA, Nelma; MATOS, 2001),uma MT pode ser descrita como tupla contendo as seguintes informações:

$$M = (Q, \Sigma, \Gamma, \delta, q_0, \beta, F),$$

onde

- Q é um conjunto finito de estados;
- Σ conjunto finito de símbolos de entrada.
- Γ conjunto completo de símbolos, Σ é um subconjunto.
- δ é uma função parcial $\delta : Q \times \Gamma \longrightarrow Q \times \Gamma \times \{L, R\}$, chamada de função de transição.
- $q_0 \in Q$ é o estado inicial;
- β é um símbolo que representa um branco (está em Γ mas não em Σ).
- $F \in Q$ é o subconjunto de estados finais.

Primeiramente a *string* de entrada é escrita na fita e o restante das posições são preenchidas com o símbolo β. O cabeçote é inicializado na posição mais à esquerda que contém um símbolo da *string* de entrada.

Um passo de computação (movimento da Máquina de Turing) depende da sua configuração atual e da programação a ser executada (transição).

A função de transição $\delta(q_i, a) = (q_j, b, d)$ indica que quando a máquina está no estado q_i lendo um símbolo a, ela deve mudar para o estado q_j , escrever o símbolo b no lugar do símbolo a e, em seguida, mover o cabeçote na direção d.

Os computadores modernos são MT (exceto pelo fato de terem memória finita)(MOREIRA, Nelma; MATOS, 2001):

- O processador corresponde à unidade de controle, cujos estados podem ser definidos pelos padrões de bits que podem ser associados aos registradores;
- A memória da máquina corresponde ao sistema de armazenamento em fita;
- Os padrões de bits (0 e 1) correspondem ao alfabeto da fita.

Existem três maneiras de abordar o estudo das máquinas de Turing e e de seus modelos equivalentes:

- Processamento de funções - Funções computáveis e suas propriedades;
- Reconhecimento de linguagens - Linguagens que podem ser reconhecidas e suas propriedades;
- Solucionabilidade de problemas - Problemas solucionáveis e não solucionáveis, problemas parcialmente solucionáveis (computáveis) e completamente insolúveis (não computáveis), bem como suas propriedades.

A terceira (solucionabilidade de problemas) constitui um dos problemas fundamentais da ciência da computação.

5.8.1 Variações da Máquina de Turing

1. Máquina de Turing Não Determinística

   A máquina de Turing não determinística é tal que, para o mesmo valor de entrada, pode existir mais de um valor de saída. Assim, para o mesmo estado corrente e símbolo lido, diversas alternativas são possíveis. Cada alternativa é percorrida de forma totalmente independente. Isto significa que as alterações de conteúdo na fita

realizadas em um caminho não modificam o conteúdo da mesma nos demais caminhos alternativos.

A máquina, ao processar uma entrada, tem como resultado um conjunto de novos estados. Ou seja, assume um conjunto de estados alternativos, como se houvesse uma multiplicação da unidade de controle, uma para cada alternativa, processando independentemente, sem compartilha, recursos com as demais. Assim, o processamento de um caminho não influi no estado geral, nem no símbolo lido dos demais caminhos alternativos.

Para uma Máquina de Turing M não determinística e para uma palavra w:

- w pertence a ACEITA(M), se existe pelo menos um caminho alternativo que aceita a palavra;
- w pertence a REJEITA(M), se todas as alternativas rejeitam a entrada;
- w pertence a LOOP(M), se nenhum caminho aceita a palavra e pelo menos um fica em *loop*.

2. Máquina de Turing com fita infinita à esquerda e à direita

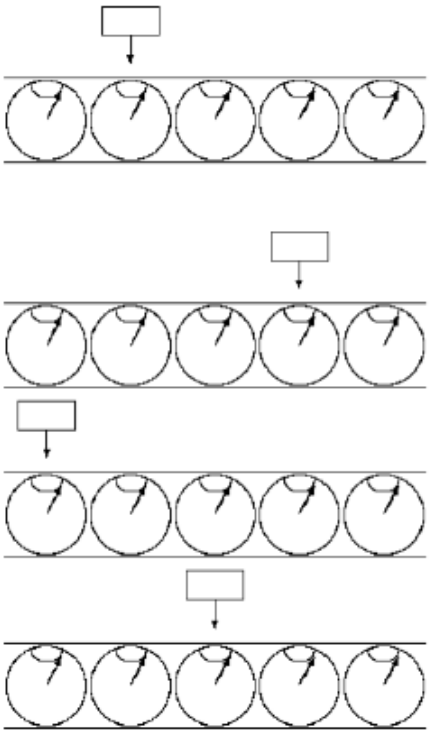
   Ama modificação da definição básica da máquina de Turing, permitindo que a fita seja infinita dos dois lados, não aumenta o seu poder computacional. Na realidade, a fita infinita à esquerda e à direita é facilmente simulável por uma fita tradicional.

3. Máquina de Turing com múltiplas fitas

   A máquina de Turing com múltiplas fitas possui k fitas infinitas à esquerda e à direita e k correspondentes cabeças de fita. O processamento é realizado como segue:

   - Inicialmente, a palavra de entrada é armazenada na primeira fita, ficando as demais com valor branco;
   - depende do estado corrente da máquina e do símbolo lido em cada uma das fitas;
   - grava um novo símbolo em cada uma das fitas, move cada uma das cabeças independentemente para a esquerda ou para a direita, e a máquina assume um (único) novo estado.

Figura 15 – Exemplo Máquina de Turing com Múltiplas Fitas.



Fonte: (DIVERIO; MENEZES, 2009)

5.9 Máquina de Turing Probabilística

Uma Máquina de Turing Probabilística (MTP) corresponde à formalização do conceito de um computador acoplado a um gerador de símbolos aleatórios, sendo possível também descrever seu funcionamento postergando as escolhas aleatórias para o final. Ou seja, pode-se calcular a probabilidade de se estar em uma configuração específica a cada passo e apenas ao final realizar um único sorteio a fim de determinar a configuração em que a máquina se encerra (CARDONHA; SILVA; FERNANDES, 2004).

Uma MTP pode ser interpretada, também, como uma Máquina de Turing Não Determinística (MTND) porém, distinguem-se apenas por seu modo de funcionamento. Assim sendo, todas as definições dadas para as MTND podem ser aplicadas naturalmente as MTP. Do mesmo modo, a seguinte comparação pode ser implicada: uma MTD é uma MTP em que, a cada estado, existe somente uma transição aplicável (CARDONHA; SILVA; FERNANDES, 2004).

Em uma MTP, a cada passo, tem sua transição escolhida dependentemente de uma probabilidade uniforme, diferentemente do que ocorrem com as MTND, na qual a escolha da transação é feita de maneira arbitrária dentre todas as aplicáveis. Visto isto, pode-se apontar a probabilidade de uma MTP que recebe uma determinada entrada em uma computação produzir uma saída específica (CARDONHA; SILVA; FERNANDES,

2004).

5.10 Máquina X

A máquina X é um modelo teórico de produção que permite modelar categorias complexas em máquinas teóricas bastante simples (STANNETT, 2003).

O 'X' em 'Máquina-X' refere-se ao tipo de sistema que está sendo implementado por essas operações, como, por exemplo, uma calculadora pode ser modelada como uma máquina numérica ou um aparelho de ar condicionado como uma máquina de temperatura (STANNETT, 2003).

O modelo original é baseado na máquina de estados finitos (FSM), o que oferece um modelo muito elegante, pois esse atua como estrutura de controle de qualquer programa (STANNETT, 2003).

Em sua forma padrão, uma máquina X se comporta sequencialmente, realizando uma construção de cada vez em ordem estrita, assim tornando o modelo apropriado para hábitos simultâneos ou de tempo contínuo (STANNETT, 2003).

Esses modelos estão se tornando cada vez mais importantes no campo de teste de software, por permitirem um nível de certeza anteriormente inatingível. Desde que determinadas condições mínimas sejam atendidas, é possível usar uma especificação do sistema da máquina X para gerar um conjunto de testes finito e completo (HOLCOMBE; IPATE, 2012).

6 CONSIDERAÇÕES FINAIS

Este trabalho apresentou uma Revisão Sistemática da Literatura como objetivo principal mostrar um panorama geral e analisar os modelos computacionais existentes, gerando um documento com uma compilação de informação a fim de auxiliar aqueles que se interessarem por essa área.

Após a efetivação das etapas de planejamento e execução, foram selecionados para análise 14 artigos publicados no período de 1996 e o atual ano, 2020, o que indica que ainda há um crescente interesse sobre o assunto.

Como resultado dessa RSL pode-se inferir, visando assim responder uma pergunta implícita de qual modelo pode ser dito o melhor, que não apenas entre os modelos computacionais retornados (ARNN, Automato Celular, Computação em DNA, Computação Quântica, Lambda Cálculo, Máquina de Post, Máquina de Registradores, Máquina de Turing, Máquina de Turing Probabilística e Máquina X), que não pode-se categoricamente afirmar que existe um melhor modelo, visto que cada um representará um custo e poder computacional que será vantajoso ou não dependendo da situação.

Apesar dos principais modelos abordados pelos artigos estudados terem sido a Máquina de Turing, a Computação Quântica e a Computação em DNA, apenas as duas últimas são mencionadas projetando-se um futuro, se mostrando mais atrativas no cenário atual, podendo ser apontadas como as mais viáveis, contudo essa revisão não foi capaz de eleger um e responder com convicção a pergunta de qual seria o modelo mais viável computacionalmente.

Contudo, a última pergunta levantada pela revisão foi satisfatoriamente respondida, concluindo como o modelo mais didático o modelo computacional Máquina X, que por ser o mais simples apresentou maior poder de compreensão e manipulação.

Os demais modelos apresentados e citados têm sua relevância porém, quando voltado para o futuro, a área científica quer inovar e não apenas expandir o conhecimento existente.

Como pôde-se perceber anteriormente, o objetivo desta RSL foi atingido de forma parcial, uma vez que as pesquisas não identificaram artigos que fornecessem ferramentas concretas para um comparação entre os modelos. No entanto, foram obtidos resultados interessantes, sobretudo sobre o os novos rumos dessa área da computação.

Tal fato mostra aos pesquisadores que ainda pode-se desenvolver novas contribuições nessa área.

Como trabalhos futuros, sugere-se expandir esta revisão englobando novas bases de

dados e também alterando a *string* de busca. Outro caminho para um trabalho futuro seria expandir as explanações sobre os modelos computacionais, principalmente o ARNN e a Máquina X que tiveram poucos dados encontrados na RSL do presente trabalho.

REFERÊNCIAS

ANTONIO, W.; JUNIOR, S. COMPUTADORES DE DNA : UMA NOVA TECNOLOGIA. 2011. Citado na página 51.

AUGUSTO, G.; FILHO, C. N. L. Decidibilidade. Citado na página 60.

BA, K.; CHARTERS, S. Guidelines for performing systematic literature reviews in software engineering. v. 2, 01 2007. Citado na página 38.

CARDONHA, C. H.; SILVA, M. K. D. C.; FERNANDES, C. G. Computação Quântica: Complexidade e Algoritmos. 2004. Disponível em: <https://linux.ime.usp.br/{~}cef/mac499-04/monografias/cardonha/quantum.p>. Citado 5 vezes nas páginas 27, 29, 52, 64 e 65.

CARLOS, J. et al. Computação Paraconsistente : Uma Abordagem Lógica à Computação Quântica. 2009. Citado 2 vezes nas páginas 52 e 53.

CARVALHO, R. de. *Modelos de computação e sistemas formais*. DCC/IM, COPPE/UFRJ, NCE-UFRJ, 1998. Disponível em: <https://books.google.com.br/books?id=EoL3GwAACAAJ>. Citado na página 23.

CASTRO, M. L. A.; CASTRO, R. D. O. Autômatos celulares: implementações de von Neumann, Conway e Wolfram. *Revista de Ciências Exatas e Tecnologia*, v. 3, n. 3, p. 89–106, 2008. ISSN 1980-1793. Citado 3 vezes nas páginas 48, 49 e 50.

CORDEIRO[1], A. M. et al. Revisão sistemática: uma revisão narrativa. SciELO Brasil, 2007. Citado na página 25.

CORMEN, T. et al. *Introduction to Algorithms*. MIT Press, 2009. (Computer science). ISBN 9780262533058. Disponível em: <https://books.google.com.br/books?id=aefUBQAAQBAJ>. Citado na página 28.

CULL, P. Biocomputation: Some history and prospects. *BioSystems*, Elsevier Ireland Ltd, v. 112, n. 3, p. 196–203, 2013. ISSN 03032647. Disponível em: <http://dx.doi.org/10.1016/j.biosystems.2012.12.005>. Citado na página 45.

DIVERIO, T.; MENEZES, P. *Teoria da Computação - 3.Ed. - UFRGS: Máquinas Universais e Computabilidade*. Bookman, 2009. ISBN 9788577808311. Disponível em: <https://books.google.com.br/books?id=459EInmoh2cC>. Citado 10 vezes nas páginas 24, 27, 54, 55, 56, 57, 58, 59, 60 e 64.

FILHO, C. *História da computação: O Caminho do Pensamento e da Tecnologia*. Edipucrs, 2007. ISBN 9788574306919. Disponível em: <https://books.google.com.br/books?id=_YRy1lKnniEC>. Citado na página 23.

FILHO, E. I. Uma Metodologia para Computação com DNA Dissertação apresentada como requisito parcial para a obtenção do grau de. p. 83, 2004. Disponível em: <https://www.lume.ufrgs.br/bitstream/handle/10183/16662/000704501.pdf?sequence=1>. Citado na página 50.

GRANDE, C. I. C. Uma Revisão Sistemática da Literatura sobre Desenvolvimento de Aplicativos para Dispositivos Móveis : Tendências e Desafios Uma Revisão Sistemática da Literatura sobre Desenvolvimento de Aplicativos para Dispositivos Móveis : Tendências e Desafios. 2014. Citado na página 35.

GREMONINI, L. AUTÔMATOS CELULARES : REVISÃO BIBLIOGRÁFICA E. n. March, 2020. Citado 2 vezes nas páginas 48 e 49.

GRILO, A. B. Projeto de Pesquisa Computação Quântica e Teoria da Computação. 2010. Citado 2 vezes nas páginas 29 e 30.

GUEDES, E. B.; JR, B. L. Máquina de Turing Quântica Q a Aceitação Rejeição. p. 3–6, 2004. Citado na página 53.

HOLCOMBE, M.; IPATE, F. *Correct Systems: Building a Business Process Solution*. Springer London, 2012. (Applied Computing). ISBN 9781447134350. Disponível em: <https://books.google.com.br/books?id=Go_aBwAAQBAJ>. Citado na página 65.

JESUS, H. Revisão sistemática de engenharia de software experimental in vitro: uma análise preliminar. 2015. Disponível em: <http://repositorio.ufla.br/handle/1/5066>. Citado 3 vezes nas páginas 25, 31 e 32.

MAFRA, S. N.; TRAVASSOS, G. H. Estudos Primários e Secundários apoiando a busca por Evidência em Engenharia de Software. *Relat{ó}rio T{é}cnico, RT-ES*, v. 687, n. 06, p. 32, 2006. Disponível em: <http://www.cin.ufpe.br/{~}in1037/leitura/EBSE-MafraTravassos-COPPE-2006.p>. Citado na página 32.

MOREIRA, Nelma; MATOS, A. Máquinas de Turing – Introdução. 2001. Citado 3 vezes nas páginas 60, 61 e 62.

MUELLER, J. *Programação Funcional Para Leigos*. Alta Books, 2019. (Para Leigos). ISBN 9788550813509. Disponível em: <https://books.google.com.br/books?id=bJ_BDwAAQBAJ>. Citado na página 54.

NEIVA, F. W. *Revisão Sistemática da Literatura em Ciência da Computação Um Guia Prático*. 7 p. Tese (Doutorado) — Universidade Federal de Juiz de Fora, 2016. Citado na página 32.

OLIVEIRA, I. C. Complexidade computacional eo problema P vs NP. p. 125, 2010. Disponível em: <http://www.bibliotecadigital.unicamp.br/document/?code=000772316>. Citado 2 vezes nas páginas 28 e 29.

PAPADIMITRIOU, L. *Elements Of The Theory Of Computation 2Nd Ed.* Prentice-Hall Of India Pvt. Limited, 1998. (Prentice-Hall software series). ISBN 9788120322332. Disponível em: <https://books.google.com.br/books?id=jJJUnQAACAAJ>. Citado na página 24.

PY, M. X. Análise da máquina de turing persistente com múltiplas fitas de trabalho. 2003. Citado na página 24.

ROCHA, A. d. R. et al. Computação Baseada em DNA. p. 12, 2012. Disponível em: <http://www.ic.unicamp.br/{~}rocha/college/src/dnaComputing.p>. Citado na página 50.

SAMPAIO, R.; MANCINI, M. Estudos de revisão sistemática : um guia para síntese. *Revista Brasileira de Fisioterapia*, v. 11, p. 83–89, 2007. ISSN 1413-3555. Citado na página 25.

SENOCAK, G. *Revisao Sistematica da Literatura em Engenharia de Software_ Teoria e Pratica*. [S.l.: s.n.], 2019. ISSN 1098-6596. ISBN 9788578110796. Citado 4 vezes nas páginas 31, 32, 33 e 36.

SIEGELMANN, H. T. The simple dynamics of super Turing theories. *Theoretical Computer Science*, v. 168, n. 2, p. 461–472, 1996. ISSN 03043975. Citado 2 vezes nas páginas 47 e 48.

SIEGELMANN, H. T. Neural and Super-Turing Computing. p. 103–114, 2003. Citado 3 vezes nas páginas 45, 47 e 48.

SILVA, W. J. N. da. *Uma introdução à Computação Quântica*. Tese (Doutorado) — Universidade de São Paulo, 2018. Citado 2 vezes nas páginas 51 e 52.

SOLTEIRA, I. JOÃO PAULO DA CRUZ ALMEIDA INDUÇÃO FINITA , DEDUÇÕES E MÁQUINA DE TURING. 2017. Citado na página 60.

STANNETT, M. Computation and Hypercomputation. p. 115–153, 2003. Citado 3 vezes nas páginas 44, 46 e 65.

TANENBAUM, A. *Organização estruturada de computadores*. PRENTICE HALL BRASIL, 2013. ISBN 9788581435398. Disponível em: <https://books.google.com.br/books?id=q9vBjwEACAAJ>. Citado na página 23.

ZENIL, H. What Is Nature-Like Computation? A Behavioural Approach and a Notion of Programmability. *Philosophy and Technology*, v. 27, n. 3, p. 399–421, 2014. ISSN 22105441. Citado na página 44.

ZUBEN, F. J. V.; ATTUX, R. R. Introdução a Computação Natural Computação de DNA. 2003. Citado 2 vezes nas páginas 50 e 51.