Implementação de Máquina de Turing Não-Determinística de Fita Limitada onde as Arestas são Ponderadas e Utiliza o Algoritmo de Dijkstra

Victor Ferreira Braga[11]
William Valther Silva Martins[2]

Universidade Federal do Maranhão – UFMA – São Luís – Maranhão
victor.braga@discente.ufma.br
william.valther@discente.ufma.br

Resumo

Este trabalho explora a implementação de uma Máquina de Turing não-determinística com fita limitada, onde as arestas são ponderadas, utilizando o algoritmo de Dijkstra. As Máquinas de Turing não-determinísticas, introduzidas por Alan Turing em 1936, são fundamentais na teoria da computação, permitindo a exploração simultânea de múltiplos caminhos de computação. Esta implementação visa analisar problemas de decisão e complexidade computacional, incorporando uma abordagem formal com uma 8-tupla que define a máquina, incluindo componentes como o alfabeto da fita, conjunto de estados, função de transição, e estado inicial. Com as arestas ponderadas tem-se uma grande quantidade de aplicações que junto com o algoritmo de Dijkstra pode ser utilizado em problemas complexos de cálculo de custos computacionais.

Palavras-Chave: Máquina de Turing. Não-determinística. Aresta Ponderada. Dijkstra.

[11] Aluno graduando no curso de Engenharia da Computação. Graduado como Bacharel Interdisciplinar em Ciência e Tecnologia.

1 INTRODUÇÃO

As Máquinas de Turing não-determinísticas (MTNDs) são um conceito fundamental na teoria da computação, introduzidas por Alan Turing em 1936. Elas possuem a capacidade de explorar múltiplos caminhos de computação simultaneamente, o que as torna uma ferramenta poderosa para a análise de problemas de decisão e complexidade computacional (Sipser, 2012).

Formalmente, uma máquina de Turing é definida como uma 8-tupla:

$$M = (\Sigma, Q, \delta, q_0, F, V, \circledast, \dagger)$$

Onde:

- **Σ:** Alfabeto da fita.
- **Q:** Conjunto de estados
- **δ:** Função de transição (regras de mudança de estado)
- **q_0:** Estado inicial
- **F:** Conjunto de estados de aceitação
- **V:** Símbolos da fita
- **$\circledast$ e $\dagger$:** Delimitadores de início e fim da fita

No contexto das MTNDs de fita limitada, a restrição imposta sobre o espaço de fita disponível para a computação adiciona uma camada extra de complexidade ao modelo clássico. Este tipo de MTND é particularmente relevante na análise de algoritmos e na compreensão de como as restrições de espaço influenciam a capacidade computacional (Papadimitriou, 1994).

A ponderação das arestas em uma MTND reflete a introdução de custos associados às transições entre estados. Esse conceito é essencial quando se deseja modelar problemas de otimização, onde a minimização do custo total de uma computação é de interesse (Cormen et al., 2009).

O algoritmo de Dijkstra, desenvolvido por Edsger Dijkstra em 1956, é um dos algoritmos mais consagrados para encontrar o caminho mais curto em um grafo com arestas não-negativas. Sua aplicação em uma MTND ponderada permite explorar de maneira eficiente as transições de menor custo, proporcionando uma nova

perspectiva sobre a otimização de caminhos em grafos (Dijkstra, 1959).

A implementação clássica do algoritmo de Dijkstra utiliza uma estrutura de dados eficiente, como uma fila de prioridades, para selecionar rapidamente o próximo nó a ser processado. Isso resulta em uma complexidade temporal de O((V + E) log V), onde V é o número de vértices e E é o número de arestas do grafo. Esta eficiência torna o algoritmo de Dijkstra uma escolha preferida em muitos aplicativos práticos, incluindo sistemas de roteamento de redes, planejamento de rotas em sistemas de navegação e diversos problemas de otimização.

Uma característica notável do algoritmo de Dijkstra é sua incapacidade de lidar corretamente com grafos que possuem arestas de peso negativo, pois tais arestas podem levar a ciclos negativos, resultando em distâncias mínimas indefinidamente diminuídas. Nesses casos, algoritmos alternativos, como o algoritmo de Bellman-Ford, são mais apropriados.

A robustez e eficácia do algoritmo de Dijkstra na resolução de problemas de caminho mais curto em grafos com pesos não-negativos fazem dele um componente fundamental no estudo de grafos e em aplicações de otimização.

2 METODOLOGIA

A implementação descrita neste trabalho baseia-se em uma Máquina de Turing Não-Determinística (MTND) de fita limitada, na qual as transições entre estados são ponderadas. Para otimizar a busca pelo caminho de menor custo, foi empregado o algoritmo de Dijkstra. A metodologia adotada é estruturada em diferentes etapas: inicia-se com a modelagem teórica da máquina, seguida pela conversão desse modelo em um grafo ponderado, culminando na aplicação do algoritmo de Dijkstra e na visualização dinâmica da execução. Cada um desses aspectos será detalhado a seguir.

2.1 Modelagem Teórica da Máquina de Turing Não-Determinística

A Máquina de Turing é um modelo matemático fundamental para a teoria da computação, sendo composta por uma fita na qual símbolos podem ser lidos e escritos, além de um cabeçote que se move para a esquerda ou para a direita. O sistema opera com um conjunto de estados, incluindo um estado inicial e um ou

mais estados de aceitação, que determinam quando a computação é concluída com sucesso.

Na versão não-determinística, diferentemente da versão determinística, onde há uma única transição possível para cada combinação de estado e símbolo lido, uma MTND pode optar entre múltiplas transições possíveis. Essa característica permite explorar simultaneamente diferentes caminhos computacionais. Contudo, na implementação proposta, a máquina foi adaptada para operar com uma fita de tamanho limitado, garantindo que o processamento se restrinja ao escopo da entrada fornecida, o que aumenta a eficiência da simulação.

Transições Ponderadas

Um dos diferenciais desta implementação é a atribuição de pesos às transições entre estados. Em uma Máquina de Turing clássica, as transições ocorrem sem custos associados. No entanto, em diversas aplicações computacionais, como otimização de rotas e problemas de tomada de decisão, é essencial considerar esses custos, que podem representar fatores como tempo de execução, quantidade de operações ou complexidade computacional.

No modelo proposto, cada transição possui um peso específico, permitindo comparações entre diferentes caminhos. Essa estrutura possibilita a aplicação do algoritmo de Dijkstra, que será abordado posteriormente, permitindo que a máquina identifique e siga o caminho de menor custo para alcançar um estado de aceitação.

2.2 Transformação da Máquina de Turing em um Grafo Ponderado

Para viabilizar a utilização do algoritmo de Dijkstra, a Máquina de Turing foi modelada como um grafo ponderado. Cada estado corresponde a um vértice do grafo, e cada transição foi representada por uma aresta direcionada, cujo peso reflete o custo da transição.

Diferentemente de um grafo tradicional, onde as conexões entre vértices representam relações fixas, neste modelo as transições dependem do estado e do símbolo lido na fita. No código, essa estrutura é representada por meio de uma tabela de transições, armazenando para cada par (estado, símbolo) as possíveis transições, incluindo o estado de destino, o símbolo a ser escrito, a direção do

movimento do cabeçote e o custo da transição.

Essa abordagem permite reformular a execução da máquina como um problema de otimização de caminhos, onde se busca minimizar o custo necessário para atingir um estado de aceitação.

2.3 Aplicação do Algoritmo de Dijkstra

Uma das inovações centrais deste trabalho foi substituir a busca em largura (BFS) pelo algoritmo de Dijkstra, otimizando a seleção de transições com base no custo acumulado. Enquanto a BFS encontra o caminho mais curto em termos de número de passos, ela desconsidera os pesos das transições, o que pode ser um fator limitante em cenários onde a eficiência é crucial.

O algoritmo de Dijkstra, por outro lado, é projetado para encontrar o caminho de menor custo em grafos ponderados. Para isso, utiliza uma fila de prioridades (implementada por meio da biblioteca `heapq`), que permite a seleção dinâmica das transições com menor custo acumulado.

Na implementação, a cada passo da execução da máquina, a transição mais eficiente é priorizada. Quando um estado de aceitação é alcançado, o algoritmo retorna o caminho percorrido e o custo total. A função `run()` no código implementa essa lógica, atualizando continuamente os custos acumulados e registrando a sequência de estados e conteúdos da fita para posterior análise.

2.4 Visualização Dinâmica e Animação

Para facilitar a compreensão do processo computacional, foi desenvolvida uma visualização dinâmica da execução da Máquina de Turing. A animação permite acompanhar a evolução do estado da máquina, o conteúdo da fita e o caminho percorrido ao longo do tempo, destacando as transições realizadas.

Essa visualização foi implementada utilizando a biblioteca `matplotlib` para a geração dos gráficos interativos e a biblioteca `networkx` para a construção e exibição do grafo de transições. A cada novo passo da execução, a animação é atualizada para refletir o estado atual da máquina e o custo acumulado da trajetória

percorrida.

A função `draw_graph()` é responsável por criar essa representação visual, utilizando a estrutura `history`, que armazena o histórico de estados e transições ao longo da execução. Além de auxiliar no entendimento do processo, essa abordagem facilita a depuração do código e pode ser utilizada para gerar animações exportáveis, como arquivos GIF, permitindo sua reutilização em análises posteriores.

2.5 Resumo da Execução

Em síntese, a implementação desenvolvida integra conceitos da teoria da computação com técnicas de otimização e visualização dinâmica. A Máquina de Turing foi modelada como um grafo ponderado, permitindo a aplicação do algoritmo de Dijkstra para priorizar as transições de menor custo. Além disso, a visualização interativa possibilita um acompanhamento detalhado da execução, tornando a análise mais acessível e didática.

Dessa forma, este trabalho não apenas explora fundamentos da teoria da computação, mas também introduz aprimoramentos na forma como a Máquina de Turing é simulada e interpretada, combinando eficiência computacional com ferramentas interativas para melhor compreensão dos processos envolvidos.

3 RESULTADOS

A Máquina de Turing foi executada na linguagem python, com os estados iniciais abaixo:

Tabela 1: Lista de estados

Estados					
q0	**q1**	**q2**	**q3**	**q4**	**q_accept**

Tabela 2: Estados da fita

Fita				
0	**0**	**1**	**1**	**0**

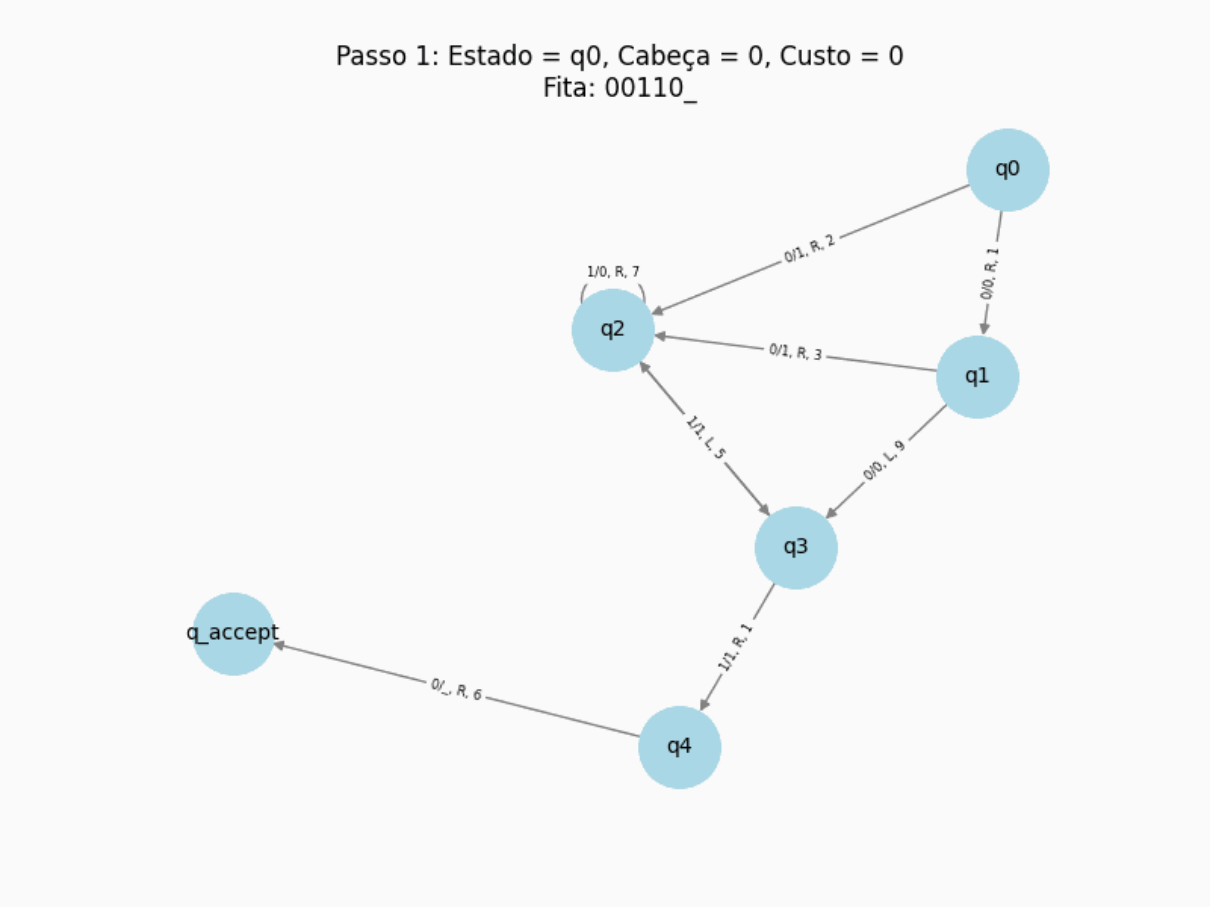
Tabela 3: Conjunto de transições

Estado_Atual	Símbolo_Lido	Próximo_Estado	Símbolo_Escrito	Direção	Peso
q0	0	q1	0	R	1
q0	0	q2	1	R	2
q1	0	q2	1	R	3
q1	0	q3	0	L	9
q2	1	q3	0	R	6
q2	1	q2	0	R	7

q3	1	q4	1	R	1
q3	1	q2	1	L	5
q4	0	q_accept	_	R	6

Com esses estados iniciais foram obtidos a seguinte Máquina de Turing:

Figura 1: Máquina de Turing Não Determinística



Devido a implementação do algoritmo de Dijkstra ele escolheu o caminho com menor custo percorrendo os caminhos da tabela 3:

Tabela 3: Passos seguidos pelo algoritmo de Dijkstra

Nº	Estado	Cabeça	Custo	Fita
1	q0	0	0	00110_
2	q1	1	1	00110_
3	q2	2	4	01110_
4	q3	3	10	01010_
5	q4	4	11	01010_
6	q_accept	5	17	0101_

4 CONCLUSÃO

O projeto utilizou uma abordagem computacional teórica combinada com otimização de caminhos em grafos, garantindo um modelo eficiente e visualmente interativo da Máquina de Turing.

O uso de Dijkstra garantiu que a execução fosse otimizada, priorizando caminhos de menor custo. A integração de visualização dinâmica tornou o processo mais intuitivo para análise e depuração.

REFERÊNCIAS

Cormen, T. H., Leiserson, C. E., Rivest, R. L., Stein, C. (2009). Introduction to Algorithms. MIT Press.

Dijkstra, E. W. (1959). A Note on Two Problems in Connexion with Graphs. Numerische Mathematik, 1, 269-271.

Papadimitriou, C. H. (1994). Computational Complexity. Addison-Wesley.

Sipser, M. (2012). Introduction to the Theory of Computation. Cengage Learning.