

# Algoritmos de Contagem Exata e Aproximada para Frequências em Fluxos de Dados: Caso de Estudo do Dataset Amazon Prime Cast

Eduardo Carvalho N°113816

**Resumo** –O processamento de *data streams* com memória limitada torna as contagens exatas inviáveis. Este estudo compara o desempenho de algoritmos determinísticos e probabilísticos na estimativa de frequência num conjuntos de dados, utilizando um *dataset* da Amazon Prime. Os resultados experimentais confirmam que o *Morris Counter* oferece compressão logarítmica com variância controlável, enquanto o *Frequent Count* identifica eficazmente elementos frequentes como um estimador de limite inferior. O trabalho conclui quantificando o compromisso crítico entre precisão e eficiência espacial em algoritmos de *streaming*.

**Abstract** – The processing of data streams under limited memory constraints renders exact counting infeasible. This study compares the performance of deterministic and probabilistic algorithms for frequency estimation over large amounts of data, using an Amazon Prime dataset as a case study. Experimental results confirm that the Morris Counter achieves logarithmic compression with controllable variance, while the Frequent Count algorithm effectively identifies frequent elements as a lower-bound estimator. The work concludes by quantifying the critical trade-off between accuracy and space efficiency in streaming algorithms.

**Palavras chave** –Data Streaming, Contagem Exata, Morris Counter, Frequent Count.

## I. INTRODUÇÃO

### A. Contextualização do Problema

Na era do *Big Data*, o processamento de fluxos contínuos de informação sob restrições severas de memória constitui um desafio fulcral na ciência de dados. O crescimento exponencial do volume e da velocidade com que os dados são gerados e recolhidos por aplicações modernas impõe constrangimentos substanciais às arquiteturas tradicionais de processamento.

Historicamente, os algoritmos clássicos foram desenvolvidos com base na premissa de que o conjunto de dados completo poderia ser armazenado e acedido aleatoriamente na memória principal (RAM) [1]. Contudo, em cenários contemporâneos de grande escala, esta suposição já não se verifica, não apenas devido aos elevados custos de hardware, mas também à latência acrescida inerente ao acesso a memória secundária ou a sistemas distribuídos. Como consequência, afigura-se imperativa a adoção de novos paradigmas algorítmicos

aptos a operar eficientemente sob restrições de espaço de memória [2], [3].

### B. O Paradigma de Data Streaming

Em resposta a estas limitações, emergiu e consolidou-se o paradigma de Data Streaming [2], [3], [4]. Neste paradigma, os dados são caracterizados como uma sequência potencialmente infinita e de alta velocidade, exigindo que o algoritmo processe cada elemento à medida que este chega. Ao contrário dos modelos estáticos, os algoritmos de streaming operam sob restrições estritas. Especificamente, impõe-se que o processamento seja efetuado numa única ‘leitura’ e que mantenham um estado interno de memória substancialmente inferior ao tamanho total do fluxo de dados ( $\mathcal{O}(\log N)$  ou  $\mathcal{O}(1)$ , idealmente) [2], [3].

Este contexto impõe a renúncia à precisão absoluta em prol da eficiência. Algoritmos aproximados e estruturas de dados probabilísticas (como *sketches*) [4] viabilizam a estimativa de propriedades estatísticas dos dados, como a cardinalidade ou a frequência de elementos, com garantias teóricas de erro delimitadas e um consumo de memória drasticamente minimizado [5].

### C. O Desafio da Contagem de Frequência

No âmbito da análise de streams, um dos problemas fundamentais é a contagem de frequência. Formalmente, dado um fluxo de dados  $S = \{x_1, x_2, \dots, x_m\}$ , onde cada elemento  $x_i$  pertence a um domínio  $U$ , o objetivo consiste em determinar o número exato de ocorrências  $f_x$  de um determinado item  $x$ .

Num cenário computacional tradicional, a solução trivial para este problema recorre a tabelas de dispersão (*hash tables*) ou contadores exatos para armazenar cada único item e a sua respetiva contagem. No entanto, esta abordagem apresenta uma complexidade espacial linear de  $\mathcal{O}(N)$ , onde  $N$  representa a cardinalidade do conjunto de itens distintos no conjunto. Quando o domínio  $U$  é vasto, como é o caso de endereços IP numa rede global, endereços de carteiras em sistemas de *blockchain* (por exemplo, carteiras *Bitcoin*, *Ethereum*, etc.) ou, no contexto deste trabalho, o conjunto de todos os atores numa base de dados considerável [6], a memória necessária para manter contadores exatos excede rapidamente a capacidade da RAM disponível, tornando a contagem exata computacionalmente inviável para fluxos massivos, como a Figura 1 pretende ilustrar.

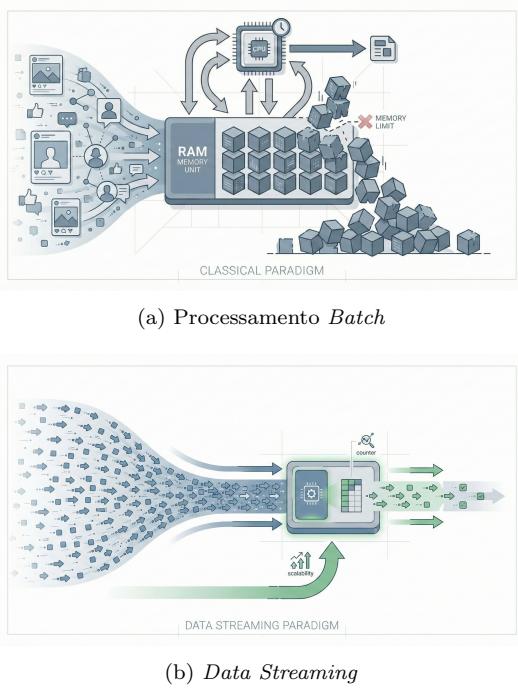


Fig. 1: Comparação entre o paradigma clássico de processamento em lote, *batch processing* (a), e o paradigma de *data streaming* (b)), evidenciando as restrições de memória e o compromisso entre precisão e eficiência.

### C.1 Abordagens Aproximadas e Estruturas Probabilísticas

Para contornar a barreira da memória linear, o novo paradigma científico propõe o uso de algoritmos aproximados e estruturas de dados probabilísticas. Estas estruturas operam através de *trade-off's* estratégicos, aceitando uma pequena margem de erro na estimativa da frequência em troca de uma redução drástica no consumo de memória [4]. O objetivo é sempre alcançar complexidades espaciais logarítmicas ou constantes.

A eficácia destas abordagens é medida não pela exatidão absoluta, mas pela garantia de que o erro de estimativa se mantém dentro de limites aceitáveis ( $\varepsilon$ ). É neste contexto que se inserem os algoritmos analisados neste estudo. O *Morris Counter*, que utiliza contagem probabilística para minimizar o espaço de armazenamento de inteiros, e o algoritmo *Frequent-Count*, desenhado para identificar os itens mais frequentes num fluxo sob restrições severas de memória.

## II. DESCRIÇÃO DOS ALGORITMOS

O objetivo principal deste trabalho, como brevemente referido acima, é análise comparativa de estratégias de contagem de frequência em fluxos de dados, avaliando o compromisso entre precisão e eficiência.

O estudo confronta três abordagens distintas:

1. **Exact Counter.** Uma implementação determinística que serve como ground truth (verdade

absoluta);

2. **Morris Counter.** Um algoritmo probabilístico que utiliza incrementos estocásticos  $1/2^k$  para minimizar o uso de memória;
3. **Frequent-Count.** Um algoritmo determinístico desenhado para identificar elementos frequentes mantendo um número fixo de contadores.

### A. Exact Counter

O algoritmo Exact Counter foi implementado para estabelecer a linha de base de comparação, permitindo o cálculo rigoroso dos erros de estimativa das abordagens aproximadas. A estrutura de dados subjacente é uma *hash map*, dicionário, onde cada entrada mapeia exatamente uma chave (o nome do ator, do tipo `string`) para um valor inteiro correspondente à sua frequência acumulada.

O método de processamento é puramente determinístico. Ou seja, ao receber um item do conjunto, o algoritmo verifica a sua existência no dicionário. Se o item já existir, o contador associado é incrementado, caso contrário, uma nova entrada é criada com o atributo inicial de 1. A consulta, `query`, devolve o valor armazenado ou zero se o item nunca tiver sido observado.

Apesar da precisão absoluta, o *Exact Counter* apresenta limitações de escalabilidade inerentes à sua natureza linear.

- Complexidade Espacial:  $\mathcal{O}(N \cdot L)$ , onde  $N$  é o número de itens distintos no fluxo e  $L$  o comprimento médio das cadeias de caracteres. Esta dependência linear em  $N$  torna a abordagem impraticável em cenários de elevada cardinalidade e memória limitada.
- Complexidade Temporal (Processamento):  $\mathcal{O}(1)$ . As operações de inserção e atualização em tabelas de dispersão apresentam, em média, tempo constante.
- Complexidade Temporal (Consulta):  $\mathcal{O}(N \log N)$ . A determinação dos  $n$  elementos mais frequentes, `get_top_n`, requer a ordenação das entradas, sendo por isso uma operação realizada apenas após o processamento do fluxo.

### B. Morris Counter

O *Morris Counter* é um algoritmo probabilístico clássico, concebido para contar eventos até um número  $n$  muito elevado utilizando registo de memória pequenos. Ao contrário do contador exato, que armazena, exatamente, o valor linear  $n$ , o *Morris Counter* armazena apenas um expoente  $k$ , introduzindo um erro controlado em troca de uma eficiência espacial logarítmica.

A implementação mantém um dicionário, só que desta vez o valor associado a cada ator não é a contagem de ocorrências, mas sim o expoente  $k$ . O processo de atualização é estocástico. De forma concreta, ao observar um item, calcula-se uma probabilidade de atualização  $P = 1/2^k$ . Gera-se, posteriormente, um número aleatório uniforme entre 0 e 1. Se o valor for

menor que  $P$ , o contador  $k$  é incrementado ( $k = k + 1$ ). Desta forma, à medida que a contagem cresce, torna-se exponencialmente mais difícil incrementar  $k$ . A função de consulta, `query`, devolve a estimativa não-viesada da contagem real através da fórmula  $\hat{n} = 2^k - 1$ .

A principal vantagem desta abordagem resulta da utilização de uma representação da contagem muito mais eficiente do ponto de vista computacional.

- Complexidade Espacial:  $\mathcal{O}(\log \log N)$  bits por contador. Ao contrário de um contador exato, que requer  $\log_2 N$  bits para representar o valor  $N$ , o Morris Counter armazena apenas o expoente  $k \approx \log_2 N$ , necessitando assim de  $\log_2(\log_2 N)$  bits.
- Complexidade Temporal (Processamento):  $\mathcal{O}(1)$ . As operações envolvidas na atualização do contador são executadas em tempo constante.
- *Trade-off* Precisão-Memória: A estimativa introduz variância. Embora a esperança matemática seja não-enviesada, i.e.,  $\mathbb{E}[\hat{n}] = n$ , uma única execução pode apresentar erro significativo, justificando a utilização de múltiplas execuções, bem como da média, para reduzir a variância.

### C. Frequent-Count

O algoritmo *Frequent-Count* constitui uma abordagem determinística concebida para identificar elementos cuja frequência de ocorrência excede uma fração predefinida do conjunto total. Em contraste com as abordagens previamente descritas, este algoritmo opera sob uma restrição de memória rígida, mantendo um número fixo de contadores  $k$  (determinado pela capacidade).

O algoritmo mantém um dicionário de contadores monitorizados, cuja cardinalidade nunca excede o parâmetro  $k$ . O processamento de cada item do fluxo segue uma lógica de três cenários.

1. Monitorização: Se o item já se encontra no dicionário, o seu contador é incrementado.
2. Inserção: Se o item não está a ser monitorizado e o dicionário ainda possui espaço livre ( $< k$ ), o item é inserido com contagem inicial de 1.
3. Decremento Global: Se o item é novo e a memória está cheia, capacidade  $k$  atingida, ocorre um passo de redução. O algoritmo decremente o valor de todos os contadores armazenados até ao momento. Os itens cuja contagem atinge zero após este decremento são removidos da memória, libertando espaço para futuros elementos.

O *Frequent Count* destaca-se pela sua eficiência em cenários de recursos limitados.

- Complexidade Espacial:  $\mathcal{O}(k \cdot L)$ . O consumo de memória depende apenas do parâmetro de capacidade  $k$ , sendo independente do tamanho do fluxo  $N$ , o que assegura um uso de memória limitado e previsível.
- Complexidade Temporal: O processamento é eficiente em média, contudo, o passo de decremento global requer a atualização das  $k$  entradas armazenadas, resultando num custo de pior caso de  $\mathcal{O}(k)$  por elemento.

• Garantia Teórica: Qualquer item com frequência superior a  $\frac{N}{k+1}$  é garantidamente identificado. As frequências estimadas constituem sempre limites inferiores da frequência real, podendo ocorrer subestimação, mas nunca sobreestimação.

## III. IMPLEMENTAÇÃO E ANÁLISE EXPERIMENTAL

Esta secção descreve detalhadamente o processo de aquisição e limpeza dos dados, bem como a avaliação quantitativa dos algoritmos desenvolvidos. A experimentação foi conduzida, empregando o *dataset* da Amazon Prime [6], com uma abordagem centrada exclusivamente no atributo `cast` (elenco).

### A. Pré-processamento dos Dados

Antes da implementação foi necessário ultrapassar alguns desafios relacionados com a qualidade e a semântica dos dados textuais. O atributo `cast` apresenta uma elevada cardinalidade e uma formatação inconsistente, exigindo um pré-processamento robusto.

Durante a análise preliminar, foram identificadas três classes de ruído que distorceriam as contagens de frequência:

- Artefactos de Dados: Registos contendo apenas dígitos (ex: "1") ou *placeholders* nulos, resultantes de possíveis erros de importação na base de dados original.
- Ruído Semântico: Atores creditados que não são humanos, nomeadamente cavalos famosos como 'Champion' (cavalo de Gene Autry) e 'Trigger' (cavalo de Roy Rogers). A sua inclusão inflacionaria artificialmente o ranking de atores.
- Títulos Académicos: A separação de strings por vírgulas resultou na interpretação incorreta de suffixos como 'Ph.D.' ou 'M.D.' como sendo nomes de atores distintos (ex: 'Michael D. Swords, Ph.D.' seria dividido em duas entidades).

Para mitigar estes problemas, implementou-se um filtro (*Blacklist*) e verificações de integridade (remoção de dígitos isolados e normalização de texto) antes dos dados serem submetidos aos algoritmos de contagem. O stream final processado contabilizou um total de 44.326 entradas válidas.

### B. Análise do Contador Probabilístico (Morris)

A Tabela 1 apresenta os resultados da aplicação do *Morris Counter*. Dado a natureza estocástica do algoritmo (probabilidade de atualização  $1/2^k$ ), os testes foram repetidos 20 vezes para calcular uma média estatisticamente relevante.

A análise dos resultados revela o compromisso, que foi sendo apresentado ao longo deste *report*, entre memória e precisão:

TABELA I: Exatidão do Contador Probabilístico (Morris 1/2<sup>k</sup>)

Rank	Actor	Real	Avg (Est)	Rel. Err (%)	Min	Max
1	Maggie Binkley	56	50.2	10.4	15	255
2	Gene Autry	32	32.6	1.9	15	63
3	Nassar	30	32.6	8.7	7	63
4	Anne-Marie Newland	25	19.4	22.4	7	63
5	Prakash Raj	24	25.0	4.2	7	63
6	John Wayne	23	23.8	3.5	7	63
7	Roy Rogers	23	26.2	13.9	7	63
8	Cassandra Peterson	22	28.2	28.2	7	127
9	Danny Trejo	22	18.6	15.5	7	31
10	Michael Madson	20	16.8	16.0	3	31
11	Amitabh Bachchan	19	19.2	1.1	3	31
12	Eric Roberts	18	20.2	12.2	7	63
13	Erin Webb	18	17.6	2.2	3	31
14	Anjali	17	16.4	3.5	3	31
15	Grace Tamayo	17	12.6	25.9	3	63

TABELA II: Impacto do Tamanho do *Buffer* na Precisão (Subestimação)

Rank	Actor	Real	k = 100	k = 500	k = 1000	k = 2500	k = 3500	k = 44000
1	Maggie Binkley	56	-	-	24	43	46	56
2	Gene Autry	32	-	-	5	20	23	32
3	Nassar	30	1	3	4	16	19	30
4	Anne-Marie Newland	25	-	-	-	8	13	25
5	Prakash Raj	24	-	-	-	11	14	24
6	John Wayne	23	-	-	-	6	11	23
7	Roy Rogers	23	-	-	-	6	11	23
8	Cassandra Peterson	22	-	13	17	19	20	22
9	Danny Trejo	22	-	-	-	5	10	22
10	Michael Madson	20	-	-	-	3	8	20
11	Amitabh Bachchan	19	-	-	-	2	7	19
12	Eric Roberts	18	-	-	-	4	7	18
13	Erin Webb	18	-	-	-	1	6	18
14	Anjali	17	-	-	-	3	6	17
15	Grace Tamayo	17	-	-	-	5	17	-

1. Convergência da Média: Apesar de ser um algoritmo aproximado, a média das estimativas aproximou-se significativamente do valor real. Para o ator mais frequente, *Maggie Binkley* (Real = 56), a estimativa média foi de 50.2, correspondendo a um erro relativo de apenas 10.4%.
2. Alta Variância: A coluna ‘Min/Max’ demonstra a volatilidade de execuções individuais. No caso de *Maggie Binkley*, as estimativas oscilaram entre 15 ( $2^4 - 1$ ) e 255 ( $2^8 - 1$ ). O caso de *Cassandra Peterson* (Erro 28.2%) ilustra situações onde a ‘moeda’ probabilística favoreceu sobreestimações pontuais ( $2^7 - 1 = 127$ ).

Estes dados validam que o *Morris Counter* é um estimador não-viesado, mas a sua utilização prática requer a utilização da média de múltiplos contadores para mitigar o desvio padrão.

#### C. Análise do Frequent-Count

A Tabela 2 ilustra o comportamento do algoritmo determinístico *Frequent Count* sob diferentes restrições de memória (capacidade  $k$ ). Este teste evidencia a propriedade de *Lower Bound* do algoritmo (nunca sobreestima) e a mecânica ‘agressiva’ de ‘decremento global’ quando o *buffer* está cheio.

Como representado na Tabela 2, para capacidades baixas ( $k = 100$  e  $k = 500$ ), o algoritmo não conseguiu identificar nenhum dos atores do *Top 15* (representados por traços na tabela). Isto ocorre porque o limiar teórico de deteção é  $N/k + 1$ . Com  $N \approx 44.000$  e  $k = 100$ , um ator precisaria de aparecer mais de 435 vezes para garantir a sua sobrevivência no *buffer*. Como a frequência máxima do dataset é 56, todos os contadores foram sucessivamente decrementados até zero e repetitivamente removidos.

À medida que  $k$  aumenta, o limiar de exclusão diminui, permitindo a deteção dos *Heavy Hitters*

- $k = 1000$ : O algoritmo começa a detetar a estrutura do topo (*Maggie Binkley* = 24), embora ainda subestime o valor real em 32 unidades devido aos decrementos iniciais.
- $k = 2500$  a 3500: A precisão aumenta substancialmente. Com  $k = 3500$ , o erro absoluto para o topo da lista reduz-se para apenas 10 unidades (Est = 46 vs Real = 56).
- $k = 44000$  (Memória Linear): Quando a capacidade iguala ou supera a cardinalidade do stream, o algoritmo comporta-se como um contador exato, apresentando erro zero, conforme demonstrado na última coluna da Tabela 2.

Estes resultados confirmam que o algoritmo *Frequent Count* é altamente eficaz para identificar elementos muito frequentes em streams, mas exige um dimensionamento cuidadoso do parâmetro  $k$  quando a distribuição dos dados é mais uniforme, como observado neste conjunto de atores.

## IV. CONCLUSÕES

Este estudo avaliou a eficácia de algoritmos probabilísticos e de streaming na estimativa de frequência em grandes volumes de dados, utilizando um dataset real da *Amazon Prime*. Os resultados obtidos validam a premissa fundamental da computação em *Data Streams*, é possível trocar a precisão absoluta por ganhos massivos em eficiência espacial, desde que as limitações de cada algoritmo sejam compreendidas e mitigadas.

A análise comparativa permite retirar três conclusões principais:

1. Compressão Probabilística: O algoritmo *Morris Counter* provou ser uma solução viável para contar eventos até à ordem dos milhares ou milhões utilizando apenas registos de tamanho reduzido (teoricamente 8 bits). No entanto, a elevada variância observada nos testes (com erros pontuais superiores a 100%) dita que este algoritmo não deve ser utilizado isoladamente. Em cenários de produção, a agregação de múltiplos contadores (média de várias instâncias independentes) é mandatória para estabilizar a estimativa.
2. Deteção de Elementos Frequentes: O algoritmo *Frequent Count* demonstrou comportar-se estritamente como um limite inferior. A experiência revelou que este algoritmo é altamente sensível à distribuição dos dados. Num *dataset*, onde a frequência dos atores é relativamente baixa comparada com o total de registos, o algoritmo exige uma capacidade de memória ( $k$ ) substancial para começar a reportar resultados úteis. Isto contrasta com cenários de tráfego de rede, onde ‘elefantes’ (IPs muito frequentes) dominam o fluxo e são detetados mesmo com  $k$  reduzido.
3. Qualidade dos Dados: O pré-processamento revelou-se tão crítico quanto a escolha do algo-

ritmo. A presença de ruído semântico (como animes creditados ou títulos académicos) teria distorcido significativamente os resultados finais, sublinhando que algoritmos de são vulneráveis a ‘lixo de entrada’, uma vez que não possuem memória para reavaliar dados passados.

Em suma, não existe uma solução universal. Para aplicações críticas onde a exatidão é inegociável, o custo linear das abordagens exatas é inevitável. Para monitorização de tendências ou estatísticas de grande escala, as abordagens aproximadas oferecem um desempenho superior, permitindo o processamento de *streams* massivas com recursos finitos.

## V. ASSISTÊNCIA POR IA

Este *report* foi desenvolvido no âmbito da unidade curricular de Algoritmos avançados da Universidade de Aveiro. Reconhece-se igualmente a utilização de inteligência artificial (*ChatGPT-5*) como apoio à organização do texto, à revisão gramatical e criação da Figura 1.

## BIBLIOGRAFIA

- [1] Jeffrey Scott Vitter, “External memory algorithms and data structures: dealing with massive data”, vol. 33, no. 2, pp. 209–271.
- [2] J. Madeira, “Data stream algorithms i”, Lecture slides, Unidade Curricular Algoritmos Avançados, 2024.
- [3] J. Madeira, “Data stream algorithms ii”, Lecture slides, Unidade Curricular Algoritmos Avançados, 2024.
- [4] J. Madeira, “Data stream algorithms iii”, Lecture slides, Unidade Curricular Algoritmos Avançados, 2024.
- [5] Daniel Anderson, Pryce Bevan, Kevin J. Lang, Edo Liberty, Lee Rhodes, e Justin Thaler, “A high-performance algorithm for identifying frequent items in data streams”, *Proceedings of the 2017 Internet Measurement Conference*, 2017.
- [6] Shivam Bansal, “Amazon prime movies and tv shows”, <https://www.kaggle.com/datasets/shivamb/amazon-prime-movies-and-tv-shows>, 2021, Acedido em: 19 de dezembro de 2025.