CL-raster: Uma Nova Abordagem para Compactação e Processamento de Séries de Dados Raster*

Luana Pereira dos Reis[1], **Daniel S. Kaster**[1]

[1] Departamento de Computação – Universidade Estadual de Londrina (UEL)
Caixa Postal 10.011 – 86.057-970– Londrina – PR – Brasil

{luana.pereira.reis, dskaster}@uel.br

Abstract. *Map algebra operations have contributed to making the efficient management and processing of spatial data essential. These operations on raster data series are costly but crucial for obtaining valuable insights. The compact data structures proposed to support these operations face issues of data representation or performance. This work presents the CL-raster, a new structure that applies line-by-line compression to the data. The CL-raster stores raster data in a compressed format ready for operations processing. Experiments conducted show that our approach is efficient, significantly outperforming the competitor in terms of processing time and memory consumption.*

Resumo. *As operações da álgebra de mapas contribuíram para que o gerenciamento e processamento eficiente de dados espaciais se tornassem essenciais. Essas operações em séries de dados raster são custosas, contudo cruciais para obter insights valiosos. As estruturas de dados compactas que foram propostas para suportar essas operações enfrentam problemas de representação de dados ou desempenho. Este trabalho apresenta a CL-raster, uma nova estrutura que aplica compressão nos dados linha a linha. A CL-raster armazena dados raster de forma comprimida pronta para o processamento de operações. Os experimentos realizados mostram que a abordagem é eficiente, superando significativamente o concorrente em tempo de processamento e consumo de memória.*

1. Introdução

A crescente demanda por eficiência na gestão e processamento de dados espaciais tem impulsionado a busca por abordagens para lidar de forma eficaz com conjuntos de dados volumosos, como os gerados por sensoriamento remoto [Tomlin et al. 1990, de Oliveira et al. 2020]. A gestão eficiente e o processamento de dados espaciais tornaram-se uma preocupação crescente para os pesquisadores, especialmente à medida que o volume desses dados continua a crescer para atender a diversas necessidades [Paiva et al. 2021].

Os Sistemas de Informação Geográfica (SIGs) utilizam diferentes modelos de dados para representar informações espaciais, sendo os principais o modelo vetorial (baseado em objetos) e o modelo *raster* (baseado em campos) [Ladra et al. 2016]. Um *raster* é uma matriz multidimensional onde todas as células são preenchidas com valores, tipicamente variáveis numéricas associadas ao ponto correspondente no espaço geográfico,

*Este trabalho foi apoiado pelas agências de fomento Fundação Araucária, CNPq e CAPES.

como temperatura, reflectância, saturação de algum nutriente no solo, entre outras. Séries de *rasters* são conjuntos de *rasters* que representam dados geoespaciais coletados em intervalos de tempo regulares ou irregulares, onde cada *raster* na série corresponde a uma instância dos dados em um momento específico.

As operações de álgebra de mapas são essenciais para a análise e manipulação de dados *raster*, englobando operações locais que processam cada célula individualmente com base em seu valor, focais que analisam uma célula e suas vizinhas dentro de uma janela e zonais que agregam valores dentro de zonas definidas por atributos comuns. A eficiência dessas operações depende do armazenamento e representação dos dados, tornando a compressão crucial para grandes conjuntos de dados *raster*. Embora existam várias estruturas compactas, como as descritas na literatura, tanto para *rasters* individuais [Brisaboa et al. 2014, Pinto et al. 2017, Ladra et al. 2016] quanto para séries de *rasters* [Brisaboa et al. 2020, Cruces et al. 2019, Silva-Coira et al. 2021], muitas são limitadas a valores binários ou exigem decodificação durante o processamento A estrutura T-k^2-raster é, atualmente, o estado da arte para esse fim. Contudo, seu desempenho degrada ao processar longas séries de *rasters*, pois seu esquema de compressão utiliza *logs* armazenando uma codificação diferencial dos dados do *raster* original para uma instância específica da série, chamada de *snapshot*, demandando realizar a decodificação durante o processamento de operações.

Este trabalho apresenta uma estrutura de dados compacta para o processamento de séries de *rasters*, denominada CL-raster (do inglês *Compressed Line raster*). Inspirada em matrizes esparsas, a CL-raster armazena uma linha de dados *raster* sem a repetição de valores, preservando a resolução espacial e temporal dos dados. A proposta concentra-se no armazenamento de valores distintos do *raster*, eliminando redundâncias e aproveitando a tendência de regiões geograficamente próximas compartilharem valores semelhantes. Projetada para permitir o processamento linha por linha de uma série de *rasters*, a abordagem proposta é especialmente relevante para lidar com grandes volumes de dados em sistemas como calculadoras de mapas *raster*, amplamente utilizados em análise geoespacial, preparação de dados para aprendizado de máquina e outras operações de dados.

O artigo apresenta resultados de experimentos que demonstram as vantagens significativas da CL-raster em relação à T-k^2-raster, avançando o estado da arte das estruturas de dados compactas para séries de *rasters*. A proposta é mais eficiente em termos de tempo de execução e requisitos de memória para operações de álgebra de mapas típicas em uma série de *rasters*. Especificamente, a CL-raster é (i) mais rápida do que a variação mais rápida implementada usando a T-k^2-raster e ainda consumindo várias vezes menos memória, e (ii) mais eficiente em termos de memória do que a variação mais econômica da T-k^2-raster, além de várias vezes mais rápida.

2. Conceitos e Trabalhos Relacionados

Operações comuns em séries de *raster* envolvem álgebra de mapas, que ainda são discutidas e definidas [Amâncio and de Senna Carneiro 2018], permitindo operações matemáticas e lógicas em dados geoespaciais. Essas operações são cruciais para processar, analisar e visualizar informações em SIGs, sendo amplamente aplicadas em áreas como sensoriamento remoto, geologia, climatologia e agricultura. A álgebra de mapas é essencial para extrair *insights* de séries temporais de *raster*, facilitando a compreensão de

padrões e mudanças ao longo do tempo em dados geoespaciais [Tomlin 1994].

Para lidar com a análise de grandes volumes de dados *raster* de forma eficiente, estruturas de dados compactas desempenham um papel fundamental. Essas estruturas permitem o processamento e a consulta de dados sem necessidade de descompressão prévia, contribuindo para a otimização do armazenamento e eficiência no acesso às informações. As principais estruturas compactas dados *raster*, k^2-tree [Brisaboa et al. 2014], 3D2D-mapping [Pinto et al. 2017] e k^2-raster [Ladra et al. 2016], são baseadas em *quadtrees*, devido à eficácia da aplicação de *quadtrees* na compressão de dados de imagem [Gonzalez and Woods 2008]. As três propostas têm variações voltadas à compactação de séries de dados raster, respectivamente, k^3-tree [Brisaboa et al. 2020], 4D3D-mapping [Cruces et al. 2019] e T-k^2-raster.

A k^2-tree é uma estrutura compacta projetada para representar grafos expressos como matrizes de adjacência [Brisaboa et al. 2014]. Baseado em *quadtrees*, ela realiza uma divisão recursiva do espaço em quadrantes, terminando quando um quadrante está totalmente preenchido com 0 ou quando todos os valores das células são alcançados. Cada nó filho na estrutura é representado por um bit positivo se houver pelo menos um bit 1 no quadrante correspondente, ou por um bit negativo se o quadrante estiver totalmente preenchido com bits 0. Na prática, é implementada por meio de dois vetores: T, que armazena todos os bits da árvore, e L, responsável pelos nós folha. No entanto, o k^2-tree enfrenta desafios distintos. Primeiramente, exige que o *raster* seja binário, o que não se aplica à maioria dos dados *raster*. Em segundo lugar, requer que o *raster* seja quadrado, caso contrário, é necessário estender e preencher com valores 0. Por fim, não considera a sequência temporal dos dados *raster*, o que é uma limitação significativa durante o processamento sequencial.

A estrutura compacta k^3-tree [Brisaboa et al. 2020] representa uma evolução significativa em relação ao k^2-tree, incorporando uma terceira dimensão para otimização. Essa dimensão adicional pode ser interpretada como *rasters* temporais, formando uma sequência ao longo do tempo, ou como adaptações de *rasters* não binários, onde a terceira dimensão representa os valores dos *rasters*. Apesar dos avanços oferecidos pela k^3-tree, ele ainda enfrenta desafios. Um deles é a perda de precisão dos dados devido à necessidade de converter valores em inteiros para indexação da terceira dimensão. Além disso, a estrutura não se adapta bem a valores extremos, resultando em desempenho comprometido quando o intervalo entre os maiores e menores valores é muito amplo,

As estruturas 3D2D-mapping [Pinto et al. 2017] e 4D3D-mapping [Cruces et al. 2019] foram desenvolvidas para lidar com dados *raster* não binários, representando uma evolução significativa na capacidade de armazenamento eficiente desses dados. O 4D3D-mapping, uma evolução do 3D2D-mapping para séries de dados, tem como principal objetivo converter dados *raster* não binários em um formato binário, utilizando a estrutura k^2-tree e k^3-tree para armazenamento eficiente. No entanto, o 3D2D-mapping, apesar de preservar a localidade espacial ao usar a *Z-order*, não demonstrou eficiência de espaço satisfatória devido ao fato de que cada coluna nunca possui mais de um bit positivo, resultando em um mapeamento consideravelmente grande. Ainda assim, em operações de busca restritas a quadrantes específicos, mostrou-se competitiva. A 4D3D-mapping apresenta vantagens significativas em eficiência de espaço de memória para séries de dados. No entanto, em termos de tempo de acesso aos

dados, a estrutura é aproximadamente quatro vezes mais lenta do que a linha de base, conforme demonstrado em diferentes conjuntos de dados.

A k^2-raster [Ladra et al. 2016, Silva-Coira et al. 2023] é uma estrutura eficiente para a compressão e indexação de dados *rasters* não binários. Utiliza uma árvore hierárquica para armazenar células raster de forma compacta e inclui índices espaciais. A estrutura divide o espaço recursivamente e aplica codificação diferencial para compressão adicional. O k^2-raster usa três vetores como estrutura, T, para armazenar valores da árvore; Lmax para valores máximos e Lmin para o valores mínimos dos quadrantes. Embora ofereça vantagens, como a eficiência na compressão, pode haver questões com precisão dos dados e a necessidade de adaptação para *rasters* não quadrados. Além disso, é a estrutura fundamental para operações da álgebra de mapas.

Por fim, a estrutura T-k^2-raster representa uma evolução do k^2-raster, especialmente projetada para lidar com dados *raster* temporais e considerada a estrutura de dados compactas mais avançada para séries de dados *raster* atualmente. A T-k^2-raster utiliza os conceitos de *snapshots* e *logs* para representar sequências temporais de dados, visando melhorar a eficiência no armazenamento e processamento dessas informações. O *snapshot* é armazenado em uma k^2-raster convencional, Figura 1(a), usado como uma amostra de referência da sequência temporal. Os *logs* armazenam um codificação diferencial em relação ao *snapshot*. Para possibilitar essa diferenciação, é necessário um vetor adicional que indica se um determinado quadrante tem uma referência ao *snapshot* correspondente. Esse vetor contém valores binários, sendo verdadeiro quando todas as células em um quadrante têm a mesma diferença em relação ao *snapshot*, ou falso quando não há igualdade, mas ainda ocorre compressão devido à uniformidade dos valores do quadrante do *log*. O processo de construção da T-k^2-raster para os *logs*, envolve dois casos distintos de compressão ilustrados na Figura 1. O primeiro caso ocorre quando o *raster* não é idêntico ao quadrante do *snapshot*, mas tem o mesmo valor em todas as células (Figura 1(b)). Já o segundo caso é quando todas as subtrações são as mesmas para todas as células, determinando que o quadrante é o mesmo que o do *snapshot*, sendo necessário apenas manter a referência no *log* (Figura 1(c)). Embora a T-k^2-raster seja eficiente em termos de espaço, ela enfrenta desafios em séries temporais extensas e cenários de memória limitada. Para executar as operações, é necessário carregar tanto o *snapshot* quanto o *log* na memória, e, em casos onde a operação envolve múltiplos *logs*, ter o *snapshot* carregado é indispensável devido à relação de codificação dos dados dos *logs* para o *snapshot*.

Em muitos casos, o processamento de séries de *rasters* é realizado linha por linha em calculadoras de *raster* para reduzir a quantidade de memória necessária. Essa abordagem é vantajosa, pois permite carregar apenas a linha atual de cada *raster* na série, o que é crucial considerando o tamanho massivo desses conjuntos de dados. No entanto, todas as estruturas mencionadas nesta seção apresentam limitações ao lidar com operações de álgebra de mapas, especialmente em cenários de processamento linha por linha. Isso ocorre devido à necessidade de percorrer extensivamente a árvore e realizar múltiplas operações para encontrar o valor real de uma célula. Outra consideração importante é que essas estruturas requerem que o *raster* seja quadrado. Caso contrário, é necessário realizar um pré-processamento dos dados para adaptá-los às estruturas, o que pode ser uma tarefa adicional e computacionalmente custosa. Diferentemente das abor-

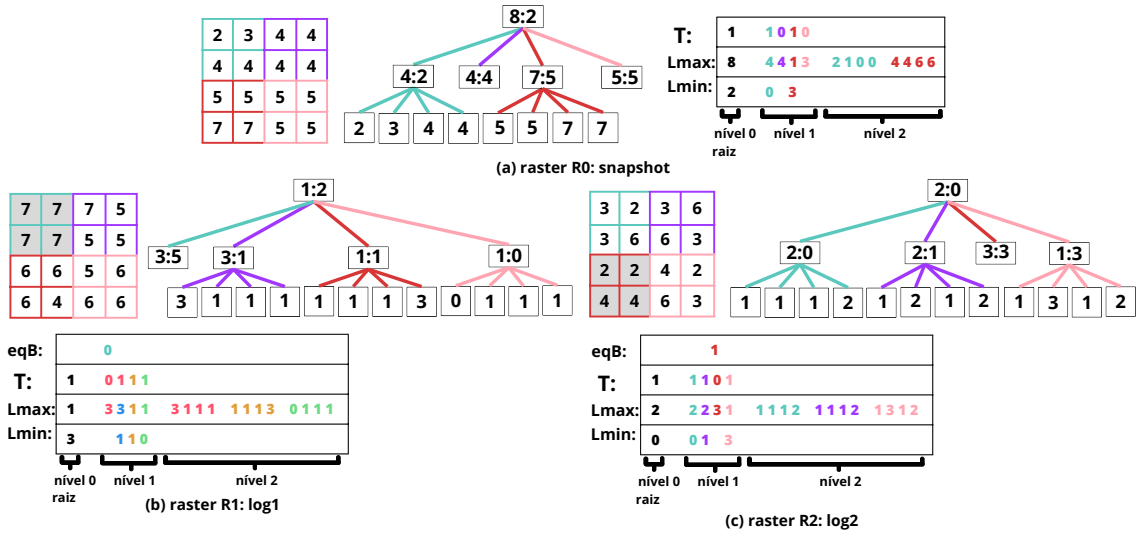

Figura 1. Representação da T-k^2-raster para uma série de três *rasters*.

dagens existentes, a proposta apresentada neste trabalho compacta os dados, é eficiente no processamento linha a linha e adapta-se a *rasters* de dimensões diferentes.

3. A Abordagem CL-raster

A nova abordagem proposta, chamada CL-raster (do inglês *Compressed Line raster*), realiza a compressão de dados *raster* baseada em linhas. O principal objetivo é permitir o processamento rápido de uma série de dados *raster* com consumo reduzido de memória. A CL-raster armazena os dados *raster* de forma comprimida, permitindo o processamento linha por linha de maneira eficiente. Além disso, essa abordagem depende de *rasters* individuais, simplificando o processamento de séries temporais de *raster* com diferentes tamanhos, ao contrário das abordagens existentes que exigem a definição prévia do tamanho da série para compressão.

A CL-raster comprime os dados usando uma abordagem baseada em linhas, inspirada no conceito de vetores esparsos e em técnicas de codificação run-length (RLE), que permite a compactação sem perdas. Mas, diferentemente de vetores esparsos, que não armazenam nulos ou zeros, a abordagem CL-raster não armazena valores repetidos sequenciais na linha do *raster*. Essa escolha é motivada pela natureza dos dados *raster*, que geralmente apresentam uma variação contínua da variável de estudo no espaço geográfico, resultando em valores semelhantes em pontos adjacentes. A eliminação de valores repetidos de células adjacentes reduz o consumo de memória, mas ainda permite o processamento direto de operações de álgebra de mapas. Uma vantagem adicional da CL-raster é que ela suporta valores de ponto flutuante, garantindo maior precisão nas operações, em contraste com os trabalhos relacionados que se restringem a valores binários, inteiros positivos ou exigem a aplicação de fatores de escala.

3.1. Organização da Estrutura

Cada linha do *raster* é representada por um vetor esparso. Este vetor esparso é estruturado como uma lista de blocos, onde cada bloco é definido por uma tupla $\tau\langle$*vetor_de_valores, quantidade, próximo*$\rangle$. Nesta tupla, *vetor_de_valores* é um vetor que armazena uma

sequência de valores distintos na linha do *raster*, *quantidade* indica o número de vezes que o último valor em *vetor_de_valores* se repete consecutivamente e *próximo* é um ponteiro para o próximo CL-raster, que será nulo se este for o último CL-raster na sequência.

Por exemplo, considere uma linha de um *raster* com valores das células $[1, 1.2, 1.3, 1.3, 1.3, 1.3, 1.4, \ldots]$. Esta linha é armazenada na estrutura como $\langle [1, 1.2, 1.3], 4, *ptr \rangle$ para o primeiro bloco, seguido por outro bloco iniciando com o valor 1.4 e assim por diante. A Figura 2 apresenta como o *raster* de entrada mostrado na Figura 2(a) é representado na estrutura proposta. A Figura 2(b) ilustra os vetores esparsos, onde cada linha l_1, l_2, l_3 contém apenas um bloco, enquanto a linha l_4 requer dois blocos.

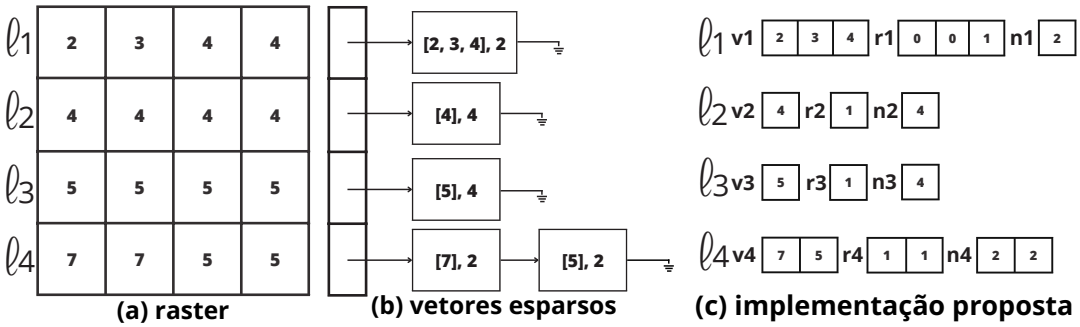

Figura 2. Raster compactado usando uma estrutura CL-raster.

Os ponteiros de memória usados na implementação convencional de vetores esparsos podem representar uma sobrecarga significativa de espaço se os dados não forem suficientemente esparsos, especialmente ao armazenar elementos de tipos de dados simples, como números de ponto flutuante. Para superar esse problema, a implementação da CL-raster elimina a cadeia de ponteiros da lista de blocos, reduzindo ainda mais o consumo de memória. Para alcançar essa otimização, cada linha comprimida é composta por três vetores:

- um vetor *v*, que armazena os valores na linha do *raster*, mantendo apenas uma ocorrência de cada sequência de valores repetidos;
- um vetor de bits *r*, indicando se o valor correspondente em v se repete ou não; e
- um vetor *n*, indicando o número de repetições sempre que for o caso.

O uso desses vetores não apenas minimiza o uso de memória ao eliminar ponteiros desnecessários, mas também simplifica o acesso aos dados comprimidos, melhorando a eficiência tanto em termos de espaço quanto de tempo de processamento. A Figura 2(c) ilustra a implementação da proposta. No exemplo, a primeira linha apresenta o vetor $v_1 = [2, 3, 4]$ contendo os valores de células não repetidos na linha, o vetor $r_1 = [0, 0, 1]$ indicando as repetições, onde apenas o terceiro valor é verdadeiro, pois 4 é o único elemento que se repete, e o vetor $n_1 = [2]$ para a contagem das repetições, armazenando 2 como o número de repetições para o valor 4. Observe-se que o número de vetores por linha é sempre três, independentemente do número de blocos usados para comprimir conceitualmente a linha. Por exemplo, a linha l_4 conceitualmente requer dois blocos, que são representados nos vetores $v_4 = [7, 5]$, $r_4 = [1, 1]$ e $n_4 = [2, 2]$. Isso indica que os valores 7 e 5 se repetem duas vezes respectivamente. Os tamanhos dos vetores v e r são os mesmos

e menores ou iguais ao número de colunas na linha do *raster*, enquanto o tamanho de n é igual ao número de valores verdadeiros em r.

As linhas l_2 e l_3 na Figura 2 exemplificam o cenário ideal para compressão na proposta, pois possuem apenas valores repetidos. O pior cenário ocorre quando não há sequência de repetições na linha do *raster*. Nesse caso, não há compressão, pois o vetor v armazena todos os elementos na linha do *raster*, exigindo a mesma quantidade de memória que a linha original. Na verdade, a proposta impõe uma pequena sobrecarga na memória necessária para armazenar o vetor de bits r cheio de valores falsos (o vetor n fica vazio). No entanto, essa sobrecarga é diminuta, tornando a proposta utilizável mesmo em casos de baixas taxas de compressão. Por exemplo, para um vetor de 512 colunas de 4 bytes, ocupando $4 \times 512 = 2.048$ bytes, o espaço adicional seria de 512 bits, ou 64 bytes, representando um aumento de cerca de 3%. No caso de sequências com apenas duas repetições, o consumo de memória permanece equivalente ao consumo original do *raster*. A única mudança é o valor repetido, que é substituído pelo número de repetições mais o consumo do vetor binário, resultando em um aumento de 3% no uso de memória. A compressão vem de três ou mais repetições contíguas, o que pode reduzir significativamente a memória em comparação ao tamanho original do *raster* para conjuntos de dados *raster* típicos (veja a Seção 4).

3.2. Algoritmos

Os algoritmos foram desenvolvidos em C++, para o processo de codificação da CL-raster a partir de um *raster* de entrada é apresentado no Algoritmo 1. Durante a execução do algoritmo, para cada linha do *raster*, a iteração inicia adicionando o primeiro valor ao vetor v, que armazena os valores únicos, e registrando no vetor r que esse valor não possui repetições inicialmente, marcando como falso. A partir do segundo valor na linha, o algoritmo verifica se ele é uma repetição do valor anterior. Para o caso em que se repete, o número de repetições no vetor n correspondente é incrementado. Se este valor for o último na linha, a posição correspondente no vetor r é atualizada para verdadeiro, e o número de repetições é registrado no vetor n. Se o valor não for uma repetição, o algoritmo verifica se o valor anterior tinha repetições. Se tinha, essa informação é registrada no vetor r e o número de repetições é armazenado em n. Em seguida, o novo valor é adicionado ao vetor v, o valor correspondente em r é definido como falso, e o algoritmo continua para a próxima iteração.

A estratégia central da CL-raster é otimizar o processamento de operações em *rasters*, especialmente a varredura sequencial dos elementos em uma linha, utilizando o padrão de projeto de *software iterator*. O Algoritmo 2 ilustra essa ideia, iterando sobre os vetores da representação física da proposta e retornando valores de acordo com o *raster* original. O algoritmo começa indexando o vetor n da linha y com a variável k e usa outra variável ($iteratorPosition$) para indicar a posição atual do iterador. Este iterador é crucial para a função $GetCell$, que será discutida posteriormente. O algoritmo itera sobre o vetor v, retornando valores e atualizando a posição do iterador. Se um valor tiver repetições, estas são retornadas uma a uma até finalizar a iteração. O algoritmo proposto impõe um custo adicional mínimo na manipulação dos vetores v, r e n ao iterar sobre os valores de uma linha de *raster*, em comparação com a iteração sobre um vetor não comprimido que armazena a linha.

Além de varrer linhas, o algoritmo permite acessar células específicas e realizar

Algorithm 1 Codificação CL-raster

Require: M ▷ Raster de entrada

```
1: for i ← 1 to num_lines(M) do
2:     v[i].push_back(M[i, 1])
3:     r[i].push_back(false)
4:     count_repetitions ← 1
5:     for j ← 2 to num_columns(M) do
6:         if M[i, j] = v[i].back() then                ▷ O valor atual se repete?
7:             count_repetitions ← count_repetitions + 1
8:             if j = num_columns(M) then               ▷ O valor atual é o último?
9:                 r[i].back() ← true
10:                n[i].push_back(count_repetitions)
11:            end if
12:        else
13:            if count_repetitions > 1 then            ▷ O valor anterior se repetiu?
14:                r[i].back() ← true
15:                n[i].push_back(count_repetitions)
16:                count_repetitions ← 1
17:            end if
18:            v[i].push_back(M[i, j])                  ▷ Adiciona o valor não repetido
19:            r[i].push_back(false)
20:        end if
21:    end for
22: end for
```

varreduras parciais em uma linha. A função $GetCell(x, y)$ realiza essas operações. A primeira chamada da função $GetCell(x, y)$ precisa iterar os vetores v, r, e n até chegar à coluna x. Essa operação resulta em complexidade linear em relação ao tamanho da linha. Porém, chamadas subsequentes de $GetCell$ têm complexidade constante devido à preservação do estado do iterador. Isso amortiza o custo das varreduras parciais de linha, que são mais frequentes que o acesso a células aleatórias no *raster*.

4. Experimentos

A estrutura CL-raster apresenta-se como uma proposta de inovação significativa no processamento e gerenciamento de dados *raster*, oferecendo uma solução eficiente para conjuntos de dados em larga escala. Esta seção descreve uma avaliação desta estrutura frente ao estado da arte, explorando suas propriedades e analisando a sua eficiência em operações de álgebra de mapas.

A bateria de testes realizada incluiu cinco tamanhos distintos para as séries de dados, considerando dois conjuntos de dados *raster* reais: CPTEC, que combina dados de precipitação observados e estimativas de satélite na América Latina, com uma grade de 924×1001 células [Rozante et al. 2020]s, e NLDAS-2, que contém dados de temperatura e indicadores ambientais na América do Norte, com uma grade de 224×464 células [Xia et al. 2012]. Foram consideradas séries de variados tamanhos: 5, 25, 50, 75 e 100 *rasters* consecutivos.

Algorithm 2 CL-raster Line Iterator

Require: v, r, n, y ▷ Os vetores CL-raster e a linha a ser varrida
1: $output \leftarrow \emptyset$; $k \leftarrow 0$; $iteratorPosition \leftarrow 0$
2: **for** $i \leftarrow 0$ **to** $|v[y]|$ **do** ▷ Percorre os valores das células
3: $iteratorPosition \leftarrow iteratorPosition + 1$
4: $output \leftarrow v[y].at(i)$
5: **if** $r[y].at(i) = 1$ **then** ▷ O valor se repete?
6: **for** $n \leftarrow 0$ **to** $|n[y].at(k)|$ **do** ▷ Adicione as repetições à saída
7: $iteratorPosition \leftarrow iteratorPosition + 1$
8: $output \leftarrow v[y].at(i)$
9: **end for**
10: $k \leftarrow k + 1$
11: **end if**
12: **end for**
13: **return** $output$

Comparou-se a CL-raster com a T-k^2-raster, avaliando-se o tempo de codificação, taxa de compressão, tempo de execução e consumo de memória. A avaliação foi realizada em um computador com processador Intel® Core™ i5-9300H CPU@2.40GHz × 8, 40GB de RAM e HDD de 2.1 TB. Os resultados são apresentados nas seções seguintes.

4.1. Tempo de Codificação e Taxa de Compressão

A Figura 3 apresenta o tempo de codificação da CL-raster e da T-k^2-raster para séries temporais de tamanhos crescentes dos dois conjuntos de dados. Nota-se que o tempo de construção aumenta com o número de *rasters* para ambas as estruturas e conjuntos de dados. No entanto, a CL-raster exibe um tempo de construção consistentemente menor que a T-k^2-raster. Especificamente, a CL-raster codificou os dados CPTEC em apenas 40% do tempo necessário para a T-k^2-raster. A maior eficiência no conjunto CPTEC se deve ao fato de que seus *rasters* são maiores que os do NLDAS-2, com quase nove vezes mais células.

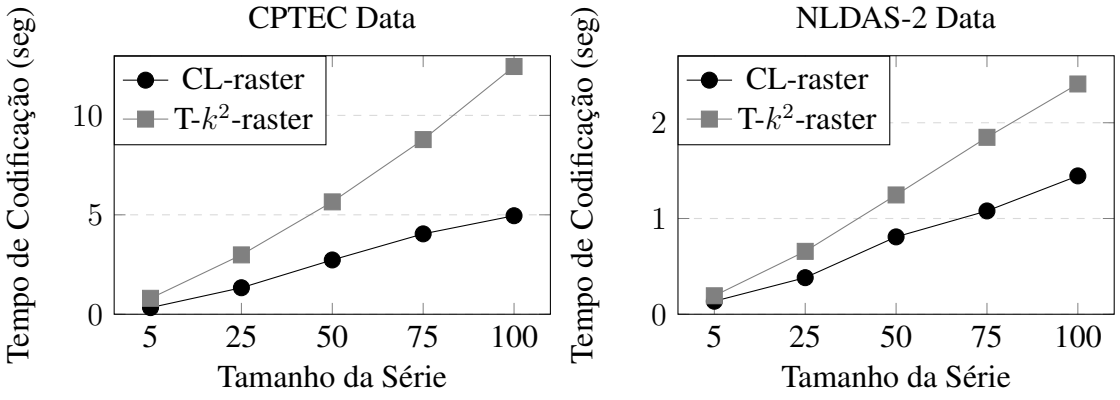

Figura 3. Tempo de codificação de séries de *rasters*.

Em relação ao desempenho de compressão, a Tabela 1 mostra que T-k^2-raster alcança uma compressão maior do que CL-raster. No entanto, a implementação padrão

da T-k^2-raster considera apenas a parte inteira dos valores do *raster*, enquanto CL-raster preserva o valor real original. Essa diferença favorece a compressão de T-k^2-raster, mas ao custo de reduzir a precisão das operações futuras nos dados *raster* codificados. Para os dados do CPTEC, CL-raster conseguiu uma redução significativa no tamanho original do *raster*, embora tenha alcançado apenas metade do poder de compressão de T-k^2-raster. No entanto, para os dados do NLDAS-2, a compressão foi baixa, devido à ausência de valores repetidos significativos, já que a temperatura é registrada com precisão de dois pontos decimais e as células cobrem áreas de até 100 km², reduzindo a probabilidade de valores iguais em células adjacentes.

É fundamental ressaltar-se que, neste teste, a vantagem significativa de T-k^2-raster sobre CL-raster com relação à taxa de compactação deve-se muito ao fato de que a T-k^2-raster codifica números inteiros, enquanto a CL-raster codifica valores de ponto flutuante. Para demonstrar isso, pré-processamos os *rasters* de entrada para truncar os valores reais para inteiros antes de codificá-los na CL-raster. Nesse caso, CL-raster (int) alcançou uma taxa média de compressão de 58,7%, mais próxima da taxa de T-k^2-raster. A T-k^2-raster pode adotar uma abordagem que utiliza o fator de escala para considerar casas decimais dos dados originais, o que oferece maior precisão nos dados, mas resulta em uma taxa de compactação menor. Ao usar um fator escalar de 100 na T-k^2-raster, a taxa de compressão cai quase pela metade, portanto existe um compromisso entre a taxa de compressão e a precisão dos dados.

Tamanho da Série	Dados CPTEC		Dados NLDAS-2	
	CL-raster	T-k^2-raster	CL-raster	T-k^2-raster
5	46.3%	91.7%	5.1%	90.0%
25	48.0%	90.8%	5.1%	87.7%
50	46.5%	90.4%	5.2%	87.3%
75	44.6%	90.1%	5.2%	87.3%
100	44.3%	89.9%	5.2%	87.2%

Tabela 1. Taxas de compressão das estruturas para séries de *rasters*.

4.2. Desempenho de Operações de Álgebra de Mapas

As operações de álgebra de mapas, como o cálculo de somas e médias para uma janela de tempo específica, são comuns ao processar séries de *rasters*. Essas operações são utilizadas para análises e extração de características em tarefas de aprendizado de máquina, como a precipitação acumulada ou a temperatura média em um período determinado. Para avaliar o desempenho de CL-raster nessas operações, implementamos a soma dos valores do *raster* na série, uma operação representativa. Para a T-k^2-raster, foram implementadas duas variações: uma usando acesso baseado em célula, recuperando apenas uma célula por vez, e outra usando acesso baseado em janela, recuperando todas as células de uma vez para realizar os cálculos.

A Figura 4 mostra os resultados de desempenho em termos de tempo de execução e consumo de memória. Note-se que os gráficos estão em escala logarítmica. A CL-raster consistentemente obteve o melhor desempenho em todos os casos. Em termos de tempo de execução, a CL-raster foi até 60% mais rápida do que a versão baseada em janela da T-k^2-raster para dados CPTEC e até 68% mais rápida para dados NLDAS-2. Comparando

a CL-raster com a versão baseada em célula de T-k^2-raster, a CL-raster foi até 35 vezes mais rápida para CPTEC e 20 vezes mais rápida para NLDAS-2.

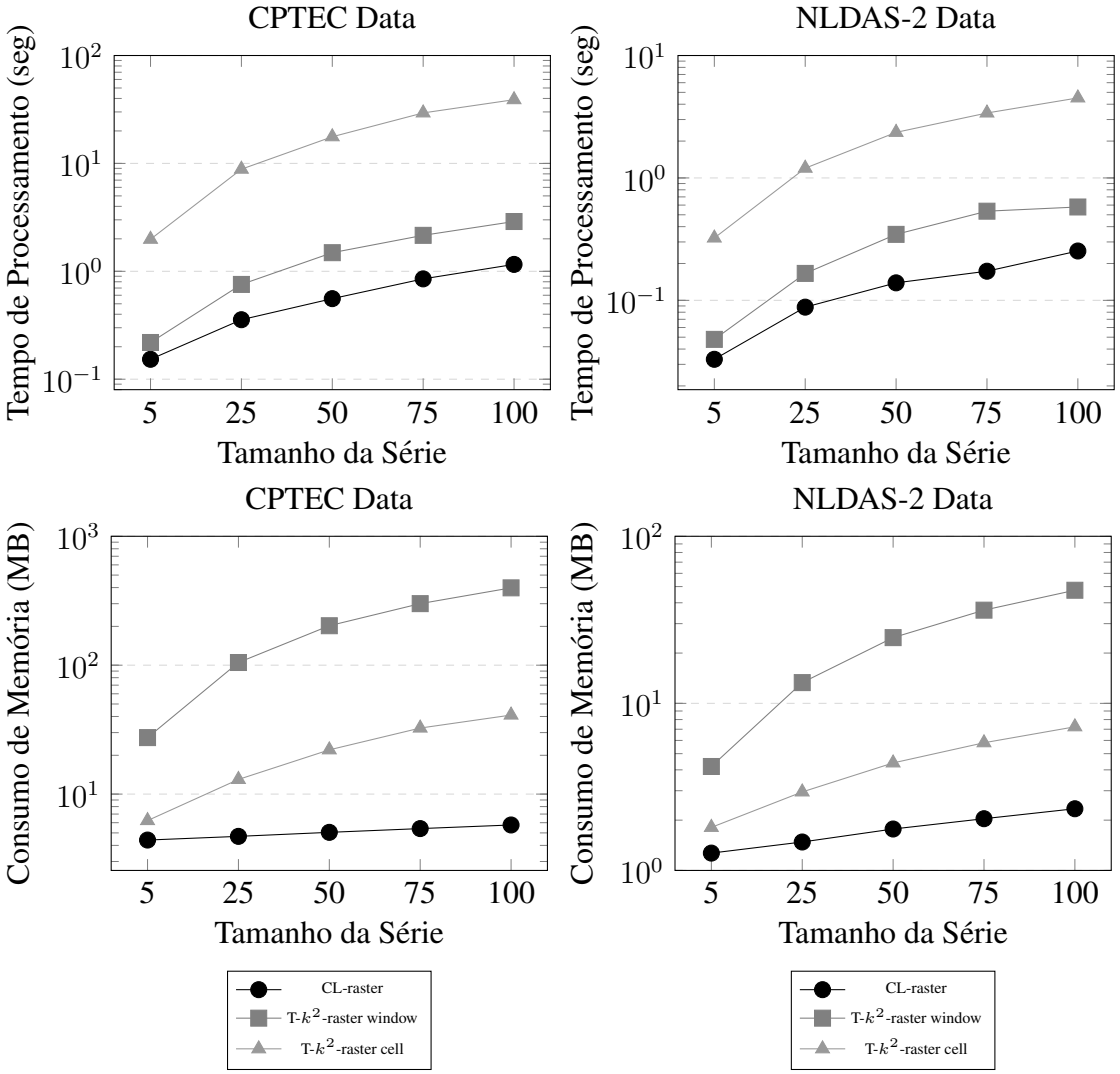

Figura 4. Desempenho de execução da soma de uma série de *rasters*.

A Figura 4 também mostra que a estrutura CL-raster alcançou maior eficiência no consumo de memória em comparação com a estrutura T-k^2-raster, tanto para a variação baseada em células quanto para a variação baseada em janela. A variação baseada em janela do T-k^2-raster é mais rápida, mas consome até 10 vezes mais memória do que a variação baseada em células, que, além de ser lenta, consumiu até 7 e 3 vezes mais memória do que a CL-raster para os dados CPTEC e NLDAS-2, respectivamente. A CL-raster mostrou um aumento menor aumento no consumo de memória conforme a série de dados crescia, enquanto o T-k^2-raster apresentou incrementos significativos.

Em resumo, a avaliação apresentada confirma que a CL-raster possibilita um processamento eficiente de séries de dados *raster* compactadas, superando ambas as variações do T-k^2-raster em termos de tempo de processamento e consumo de memória. Suas vantagens fazem dela uma ferramenta essencial para aqueles que lidam com grandes volumes de informações geoespaciais, proporcionando soluções mais eficientes para

o processamento de dados.

5. Conclusão

Este trabalho apresentou a CL-raster, uma estrutura de dados compacta para o processamento eficiente baseado em linha de séries de dados *raster*. O método proposto permite comprimir dados *raster* e realizar operações de álgebra de mapas utilizando um método eficiente baseado em linhas. A análise experimental demonstrou sua vantagem em relação aos trabalhos existentes, tanto em termos de tempo de execução quanto de requisitos de memória. Outro ponto a ressaltar é que a estrutura CL-raster não precisa ser parametrizada, ao contrário de outras estruturas encontradas na literatura, o que facilita sua ampla utilização.

A CL-raster se mostrou uma alternativa viável e promissora para o armazenamento e processamento de dados *raster*, avançando o estado da arte. Trabalhos futuros incluem a aplicação da CL-raster em outras operações de álgebra de mapas e de análise espacial, além da integração da estrutura a calculadoras de mapas para melhorar o processamento de séries de *rasters*.

Referências

Amâncio, A. F. and de Senna Carneiro, T. G. (2018). An algebra for modeling and simulation of continuous spatial changes. *Journal of Information and Data Management*, 9(3):275–275.

Brisaboa, N. R., Cerdeira-Pena, A., de Bernardo, G., Navarro, G., and Óscar Pedreira (2020). Extending general compact querieable representations to GIS applications. *Information Sciences*, 506:196–216.

Brisaboa, N. R., Ladra, S., and Navarro, G. (2014). Compact representation of web graphs with extended functionality. *Information Systems*, 39:152–174.

Cruces, N., Seco, D., and Guitérrez, G. (2019). A compact representation of raster time series. In *2019 Data Compression Conference (DCC)*, pages 103–111.

de Oliveira, S. S. T., do Sacramento Rodrigues, V. J., and Martins, W. S. (2020). SmarT: Uso de aprendizado de máquina para filtragem e recuperação eficiente de dados espaciais e temporais em big data. In *Proceedings of the 35th Brazilian Symposium on Databases (SBBD)*, pages 85–96.

Gonzalez, R. and Woods, R. (2008). *Digital Image Processing*. Prentice Hall.

Ladra, S., Paramá, J. R., and Silva-Coira, F. (2016). Compact and queryable representation of raster datasets. In *Proceedings of the 28th International Conference on Scientific and Statistical Database Management (SSDBM)*, pages 1–12.

Paiva, R. U., Oliveira, S. S., Pascoal, L. M., Parente, L. L., and Martins, W. S. (2021). Parallel processing of remote sensing time series applied to land-use and land-cover classification. *Journal of Information and Data Management*, 12(4).

Pinto, A., Seco, D., and Gutiérrez, G. (2017). Improved queryable representations of rasters. In *2017 Data Compression Conference (DCC)*, pages 320–329.

Rozante, J. R., Gutierrez, E. R., Fernandes, A. d. A., and Vila, D. A. (2020). Performance of precipitation products obtained from combinations of satellite and surface observations. *International Journal of Remote Sensing*, 41(19):7585–7604.

Silva-Coira, F., Paramá, J. R., and Ladra, S. (2023). Map algebra on raster datasets represented by compact data structures. *Softw. Pract. Exp.*, 53(6):1362–1390.

Silva-Coira, F., Paramá, J. R., de Bernardo, G., and Seco, D. (2021). Space-efficient representations of raster time series. *Information Sciences*, 566:300–325.

Tomlin, C. (1994). Map algebra: one perspective. *Landscape and Urban Planning*, 30(1):3–12. Special Issue Landscape Planning: Expanding the Tool Kit.

Tomlin, C. D. et al. (1990). *Geographic information systems and cartographic modeling*, volume 249. Prentice Hall Englewood Cliffs, NJ.

Xia, Y., Mitchell, K., Ek, M., Sheffield, J., Cosgrove, B., Wood, E., Luo, L., Alonge, C., Wei, H., Meng, J., et al. (2012). Continental-scale water and energy flux analysis and validation for the north american land data assimilation system project phase 2 (NLDAS-2): 1. intercomparison and application of model products. *Journal of Geophysical Research: Atmospheres*, 117(D3).