P-TWDTW 2.0 - Uma otimização no uso de recursos computacionais para o processamento paralelo de séries temporais de sensoriamento remoto

Roberto U. Paiva[1], **Savio S. Oliveira**[1], **Wellington S. Martins** [1]

[1]Instituto de Informática - Universidade Federal de Goiás (UFG)

urzedabr@ufg.br, wellington@inf.ufg.br,
savioteles@gmail.com

Abstract. *The processing of remote sensing time series is essential for carrying out environmental analyzes related to land use and cover. The TWDTW algorithm stands out for being designed to process this type of data. However, TWDTW depends on computing a dynamic programming matrix $O(n^2)$, consuming a lot of computational resources. Given this scenario, this work analyzes and presents an initial proposal for the optimization of a parallel version of the algorithm, called P-TWDTW. The P-TWDTW 2.0 version performed* 21.1 *times better than the original version.*

Resumo. *O processamento de séries temporais de sensoriamento remoto é essencial para realização de análises ambientas relacionadas com o uso e cobertura do solo. O algoritmo TWDTW tem se destacado por ser projetado para processar este tipo de dado. No entanto, o TWDTW depende da computação de uma matriz de programação dinâmica $O(n^2)$, consumindo muitos recursos computacionais. Diante deste cenário, este trabalho analisa e apresenta uma proposta inicial para otimização de uma versão paralela do algoritmo, denominada P-TWDTW. A versão P-TWDTW 2.0 apresentou desempenho de* $21,1$ *vezes melhor em relação a versão original.*

1. Introdução

As dinâmicas de paisagem da superfície terrestre se alteram constantemente, fazendo com que o processamento e análise de séries temporais sejam indispensáveis para identificar estas mudanças [Foody 2002]. Compreender a dinâmica de alterações no espaço e tempo auxilia na criação de políticas públicas, produtividade e redução na emissão de gases de efeito estufa [Nepstad et al. 2014, Bonan 2008, Bala et al. 2007].

Entre os métodos de análise de séries temporais de sensoriamento remoto, em particular, o algoritmo *Time-Weighted Dynamic Time Warping* (TWDTW) [Maus et al. 2016] tem se destacado, pois é um método sensível a mudanças sazonais climáticas dos diversos tipos de vegetações naturais e cultivadas. No entanto, o TWDTW é um algoritmo de ordem quadrática, demandando muitos recursos computacionais em sua utilização.

Na tentativa de mitigação desse problema, uma versão paralela do TWDTW implementada em CUDA foi proposta em [Oliveira et al. 2018], chamada de P-TWDTW. Esta implementação obteve tempos de respostas menores em relação ao algoritmo sequencial implementado em C++. Esta melhora ocorre devido ao uso da GPU (*Graphics Processing Unit*), uma arquitetura *manycore*. Porém, na implementação do P-TWDTW a

configuração do *kernel* para a computação da matriz de programação dinâmica $O(n^2)$ é definida de forma estática pelo algoritmo através do tamanho dos dados de entrada. Isso faz com que o algoritmo possua limitações de escalabilidade e desempenho.

Nesse sentido, este trabalho propõe uma estratégia para otimização do uso de recursos da GPU na configuração de *kernel* do P-TWDTW. Esta nova proposta foi considerada uma atualização do algoritmo, criando a versão P-TWDTW 2.0. O P-TWDTW é capaz de realizar a computação da medida de similaridade entre um padrão e uma série temporal de quaisquer tamanhos, com maior desempenho e melhor uso dos recursos de GPU.

O artigo está organizado da seguinte forma. A seção 2 apresenta uma breve discussão sobre a arquitetura CUDA e o algoritmo P-TWDTW. Na seção 3 é apresentada a proposta de otimização e resultados iniciais. Por fim, na seção 4 são apresentadas as conclusões e trabalhos futuros.

2. Arquitetura CUDA e Implementação Paralela P-TWDTW

Na arquitetura CUDA existe uma hierarquia de estruturas composta por *grid*, blocos e *threads*. Cada *grid* é composto por um número de blocos de *threads*, de forma que todas as *threads* em um bloco executem em um único multiprocessador de fluxo. Funções que são executadas pelas *threads* na GPU em paralelo são chamadas de *kernels*. Para lançar um *kernel* são configurados parâmetros como a quantidade de blocos e *threads* [NVIDIA 2018].

A versão paralela do algoritmo TWDTW utilizando CUDA, denominada P-TWDTW, realiza o cálculo da matriz de programação dinâmica em diagonais. Dessa forma, cada diagonal pôde ter suas células calculadas em paralelo, mantendo a dependência com as células anteriores, pois na computação desta matriz cada elemento (i, j) da matriz depende dos elementos $(i-1, j)$, $(i, j-1)$ e $(i-1, j-1)$ previamente calculados.

O P-TWDTW faz a concatenação da quantidade de padrões e séries temporais em um único vetor, podendo assim realizar a computação de vários padrões e séries temporais com apenas uma cópia de dados da CPU para a GPU. Para a realização da computação da medida de similaridade entre vários padrões e séries temporais de uma só vez, o P-TWDTW configura o *kernel* de forma a garantir que cada par (Padrão$\times$Série) tenha um bloco específico para realizar a computação de sua similaridade. Para isso são alocados para cada *kernel* um número de *threads* igual ao tamanho da diagonal principal que é o valor $min\{TP, TS\}$, sendo TP o tamanho do padrão e TS o tamanho da série temporal. O número de blocos alocados é definido de acordo com a equação 1, onde QP é a quantidade de padrões e QS a quantidade de séries temporais.

$$Blocos = QP * QS \tag{1}$$

Essa configuração limita o desempenho do algoritmo, tendo em vista que para realizar o cálculo entre um padrão e uma série temporal apenas um bloco de *threads* irá executar a tarefa. Caso os padrões e séries temporais sejam grandes, será necessário aumentar o número de *threads* dentro deste bloco até o máximo possível. A arquitetura de programação em CUDA possui atualmente um limite de 1024 *threads* para cada bloco [NVIDIA 2018]. Com o impedimento na alteração das configurações de *kernel*, O

PTWDTW não é capaz de calcular os valores corretos de medidas de similaridade entre padrões e séries temporais quando ambos forem maiores do que 1024.

3. Otimização na Configuração de *Kernel* e Resultados Iniciais

Para realizar uma otimização na configuração de kernel que permita computar a medida de similaridade entre um padrão e uma série temporal fazendo melhor uso dos recursos da GPU, a ideia consiste em usar vários blocos de *threads* para realizar a computação de um par. Esta nova estratégia elimina a limitação de escalabilidade e aumenta o desempenho durante a computação da medida de similaridade entre um par de padrão e série temporal.

Na nova configuração, foi utilizada uma quantidade fixa de *threads* por bloco igual a 32. Como o *kernel* precisa sincronizar as *threads* em todos os passos do algoritmo, para que nenhuma compute um elemento antes de suas dependências estarem calculadas, existe uma alta quantidade de chamadas de sincronização. Quando isto ocorre, utilizar blocos menores pode gerar um ganho de performance. Blocos de tamanhos múltiplos de 32 também costumam apresentar melhor desempenho, devido as instruções em *warps* definidas pela arquitetura CUDA [NVIDIA 2018]. A equação 2 apresenta a configuração de blocos, onde TP é o tamanho do padrão, TS o tamanho da série temporal e $num\_threads$ o número de *threads*.

$$Blocos = \frac{min\{TP, TS\} + num\_threads - 1}{num\_threads} \tag{2}$$

Para demonstrar o desempenho e funcionalidade da proposta, foram realizados experimentos com alguns tamanhos de padrões e séries temporais. Os experimentos foram executados em um computador com Processador Intel Core i7-9700 (3,2 GHz e 8 MB Cache), memória RAM de 16 GB DDR4 e placa de vídeo (GPU) NVIDIA GeForce GTX 1660 Ti com 6 GB GDDR6 de memória com arquitetura *Turing*, 1536 CUDA *cores* e 1770 MHz de frequência. Cada teste foi executado 10 vezes, e os resultados apresentados são a média aritmética das 10 execuções. Os dados de entrada utilizados nos testes foram gerados de forma sintética a partir do conjunto de dados utilizados em [Maus et al. 2016]. Para os testes do P-TWDTW utilizou-se a versão apresentada em [Oliveira et al. 2018].

A Figura 1 apresenta um gráfico com os resultados dos testes iniciais. O tamanho dos padrões foi mantido em 1000 e os tamanhos das séries temporais variaram de 1000 a 100000. Observa-se que a nova configuração de *kernel* obteve melhor desempenho em todos os cenários testados, obtendo um tempo de resposta $21,1$ vezes menor na computação da matriz de programação dinâmica no teste com o maior tamanho de séries temporais. O teste com tamanho de padrão 2000 e série temporal 100000 apresenta apenas o resultado do WP-TWDTW devido a limitação do P-TWDTW em manter corretude em dados com tamanhos maiores do que 1024 para o padrão e série temporal.

4. Conclusões e trabalhos futuros

O volume de dados de séries temporais de sensoriamento remoto continuará aumentando devido aos constantes avanços tecnológicos. Em consequência, o custo computacional para se trabalhar com esse volume de dados tem também uma tendência de crescimento. Explorar ferramentas, como a computação paralela, para otimizar o processamento destas

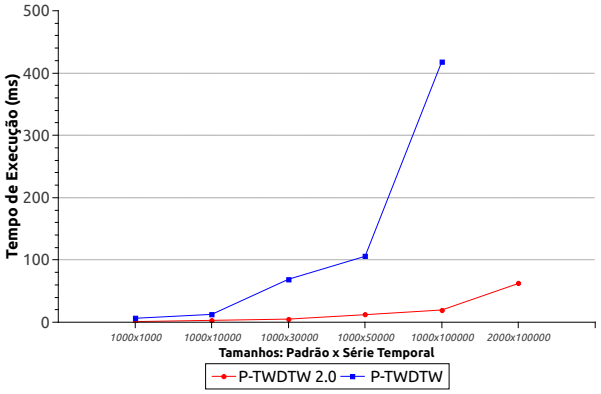

Figura 1. Comparação dos tempos de execução entre as duas configurações de *kernel* para diferentes tamanhos das séries temporais.

séries se torna uma alternativa para viabilizar as análises em grandes áreas e com grandes séries temporais.

O P-TWDTW 2.0 mostrou a possibilidade de criação de estratégias paralelas que podem obter melhor desempenho no cenário de processamento de séries temporais de sensoriamento remoto. Em trabalhos futuros, pretende-se implementar uma estratégia que aumente a carga de trabalho de cada *thread* e permita alterar a configuração de *kernel* mantendo a capacidade de computar diversas séries e padrões simultaneamente.

Referências

Bala, G., Caldeira, K., Wickett, M., Phillips, T., Lobell, D., Delire, C., and Mirin, A. (2007). Combined climate and carbon-cycle effects of large-scale deforestation. *Proceedings of the National Academy of Sciences*, 104(16):6550–6555.

Bonan, G. B. (2008). Forests and climate change: forcings, feedbacks, and the climate benefits of forests. *science*, 320(5882):1444–1449.

Foody, G. M. (2002). Status of land cover classification accuracy assessment. *Remote Sensing of Environment*, 80(1):185–201.

Maus, V., Câmara, G., Cartaxo, R., Sanchez, A., Ramos, F. M., and De Queiroz, G. R. (2016). A time-weighted dynamic time warping method for land-use and land-cover mapping. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 9(8):3729–3739.

Nepstad, D., McGrath, D., Stickler, C., Alencar, A., Azevedo, A., Swette, B., Bezerra, T., DiGiano, M., Shimada, J., da Motta, R. S., et al. (2014). Slowing amazon deforestation through public policy and interventions in beef and soy supply chains. *science*, 344(6188):1118–1123.

NVIDIA, C. (2018). Cuda c programming guide, version 9.1. *NVIDIA Corp*.

Oliveira, S. S. T., do Sacramento Rodrigues, V. J., Ferreira, L. G., and Martins, W. S. (2018). P-twdtw: Parallel processing of time series remote sensing images using manycore architectures. In *2018 Symposium on High Performance Computing Systems (WSCAD)*, pages 252–258. IEEE.