Pipeline Distribuído para Análise Espacial em Larga Escala: Avaliação da Regra 3 do Índice 3-30-300 em Fortaleza com Apache Spark e Sedona

Lucas L. Silva[1,3], **Marta C. Gonzalez** [2], **Lucas F. A. Babadopulos** [1,4], **Jorge B. Soares** [1,5] **Lara S. Furtado**[1,6]

[1]Grupo de Inteligência de Dados, INCT Infra
Universidade Federal do Ceará (UFC) – Fortaleza, CE – Brazil

[2]Department of City and Regional Planning – University of California, Berkeley

[3]Ciência de Dados – Departamento de Computação – UFC

[4]Programa de Pós-Graduação em Engenharia Civil – UFC

[5]Programa de Pós-Graduação em Engenharia de Transportes – UFC

[6]Programa de Pós-Graduação em Ciência da Computação – UFC

lucaslsilva@alu.ufc.br, martag@berkeley.edu, babadopulos@ufc.br,

jsoares@det.ufc.br, lara.furtado@insightlab.ufc.br

Abstract. *The application of spatial operations to large datasets faces limitations in traditional geoprocessing tools and libraries such as Geopandas. This work presents a distributed pipeline based on Apache Spark and Sedona to analyze geolocated data of buildings and tree coverage in Fortaleza, Brazil, identifying residences with visibility of at least 3 trees within 30 meters. Batch processing, spatial indexing, and structured persistence enabled overcoming Python's performance bottlenecks. Detailed documentation of modular code is provided, allowing scripts to be tracked and replicated in other cities to perform large-scale geometric and spatial operations.*

Resumo. *A aplicação de operações espaciais em grandes conjuntos de dados enfrenta limitações nas ferramentas tradicionais de geoprocessamento e em bibliotecas como Geopandas. Este trabalho apresenta um pipeline distribuído baseado em Apache Spark e Sedona para analisar dados geolocalizados de edificações e arborização, em Fortaleza, CE, identificando residências com visibilidade mínima de 3 árvores em 30 metros. O processamento em batches, a indexação espacial e a persistência estruturada permitiram superar gargalos do Python. Apresenta-se uma documentação detalhada de código modular que permite rastrear scripts e replicá-los em outras cidades para efetuar operações geométricas e espaciais em larga escala.*

1. Introdução

O índice 3-30-300 é um indicador internacional de três regras, criado para mensurar o acesso da população urbana às áreas verdes, apresentando uma abordagem que une a qualidade ambiental com o bem-estar físico e psicológico [Zheng et al. 2024]. A Regra 3 estabelece que toda residência urbana deve ter visibilidade a pelo menos 3 árvores dentro de uma distância de 30 metros; a 30 estabelece que cada bairro tenha pelo menos 30% de cobertura de copas de árvores; e a 300 determina que cada morador esteja a no máximo 300 metros de um espaço verde público de qualidade [Nieuwenhuijsen et al. 2022].

O cálculo dessa regra depende de um grande número de operações para medição de distância, segmentação de geometrias e contagem e filtragem de elementos espaciais. Logo, a avaliação sistemática desse critério esbarra em limitações técnicas das ferramentas de geoprocessamento tradicionais como o QGIS, e de bibliotecas em Python como o GeoPandas, especialmente diante de bases de dados com centenas de milhares de edificações e espaços verdes [Croeser et al. 2024].

Neste contexto, o objetivo central do artigo é apresentar uma metodologia para avaliar a regra 3 em Fortaleza. Para viabilizar a replicabilidade dessas análises em larga escala para outras cidades, apresenta-se também um pipeline geoespacial distribuído, baseado em Apache Spark e Apache Sedona. O Spark é uma plataforma de processamento de dados distribuído e em memória, acessível pela biblioteca PySpark, e sua extensão Sedona realiza operações espaciais [Lyon et al. 2025]. As heurísticas apresentadas na metodologia foram validadas com PySpark e a biblioteca Geopandas, mais difundida, permitindo comparações nas diversas etapas de padronização e limpeza dos dados, análises e persistência de resultados. Destacam-se as vantagens do PySpark para processamento em *batches*, logs detalhados e modularização por etapas e regras.

2. Fundamentação Teórica

Nos últimos anos, o interesse acadêmico pelo índice 3-30-300 cresceu de forma significativa; porém, sua aplicação em escala municipal enfrenta desafios computacionais. A Regra 3, em particular, demanda avaliações no nível da edificação individual, exigindo o cálculo da visibilidade de milhões de residências em relação a milhões de árvores, algo inviável com abordagens convencionais [Konijnendijk 2023]. A maioria dos estudos limita-se à análise do índice em recortes geográficos agregados, como bairros ou setores censitários, o que reduz a granularidade e pode mascarar desigualdades intraurbanas [Croeser et al. 2024, Wyrzykowski and Mościcka 2024].

Ferramentas amplamente utilizadas na análise espacial em Python, como GeoPandas, Shapely e QGIS, apresentam restrições diante de grandes volumes de dados urbanos, pois realizam operações sequenciais e mantêm todos os dados em memória. Isso gera gargalos de processamento e limitações de escalabilidade em cidades de grande porte [García-García et al. 2023]. Dessa forma, avanços metodológicos em acessibilidade a áreas verdes dependem da adoção de estratégias baseadas em processamento geoespacial distribuído e técnicas de Big Data, capazes de oferecer análises detalhadas e abrangentes do espaço urbano [Croeser et al. 2024].

Apache Sedona (antigo GeoSpark) é um sistema de computação distribuída que se integra ao Apache Spark, habilitando o processamento de dados espaciais em larga escala.

Sedona oferece suporte a feições geográficas (Point, Polygon), índices espaciais (R-tree, Quad-tree), consultas espaciais, operações de distância, *join* espacial, e processamento distribuído e em série [Yu et al. 2015]. Frameworks como Sedona superam ferramentas tradicionais para efetuar análises urbanas dependentes de Big Data espacial, como estudos de alagamento [Bai et al. 2024] e qualidade do ar [Moussa 2021]. No entanto, ainda não há registros de sua utilização para análise computacional do índice 3-30-300, e nem de sua aplicação para estudos em cidades brasileiras. Da mesma forma, não são comuns na literatura pipelines totalmente documentados da interface Spark+Sedona.

O presente trabalho preenche essa lacuna ao implementar um pipeline distribuído detalhado para avaliar a Regra 3 em Fortaleza. São exploradas técnicas de carregamento, limpeza e indexação de dados, de cálculos de distâncias ao vizinho mais próximo e de persistência. Conclui-se com uma comparação sobre as principais funções dos dois sistemas e suas vantagens no processamento de Big Data espacial.

3. Dados e ferramentas

Foram utilizados dados oficiais de edificações e arborização do município de Fortaleza, no formato GeoJSON, além de informações de bairros e demais camadas auxiliares espaciais. Os dados compreendem 1.006.080 polígonos de todas as residências mapeadas na cidade e 660.636 pontos representando a localização das árvores, reprojetados para o sistema de referência métrico EPSG:31984.

O pipeline foi desenvolvido em Python 3.10, utilizando o Apache Spark 3.3.2 e a biblioteca Apache Sedona 1.3.1-incubating, com recursos mínimos de 16 GB de RAM para processamento eficiente em ambientes distribuídos. O ambiente foi isolado por ambientes virtuais (Conda/venv) e configurado para desabilitar PyArrow, garantindo compatibilidade com Pandas a partir da versão 2.0.

4. Metodologia de cálculo da Regra 3 com pipeline Spark + Sedona

O pipeline foi estruturado em 4 fases e respectivas etapas, detalhadas a seguir.

4.1. Saneamento dos dados

Consistiu nas seguintes etapas: (i) Leitura distribuída dos arquivos GeoJSON; (ii) Conversão para o formato Well-Known Text (WKT), compatível com as funções do Sedona; (iii) Validação topológica (ST_IsValid e ST_MakeValid), garantindo a integridade das feições; (iv) Padronização do sistema de referência espacial; (v) Dissolução de polígonos fragmentados, garantindo que cada edificação seja representada por seu contorno externo; (vi) Sequência de operações Join para enriquecimento dos dados com atributos socioeconômicos e espaciais; e (vii) Criação de uma versão duplicada do dataset, removendo entidades redundantes.

4.2. Particionamento e Processamento em Batches

Consistiu nas seguintes etapas: (i) Divisão das edificações em lotes controlados por parâmetros customizáveis (batches), otimizando uso de CPU e memória; (ii) Persistência de cada batch com compressão (formato snappy) que permite que os dados sejam gerados em etapas intermediárias do processamento, em vez de mantê-los na memória RAM local (exigindo alto poder computacional) ou de descartá-los após cada operação (exigindo reprocessamento para obter resultados novamente); (iii) Log rigoroso para garantir rastreabilidade das execuções e resultados intermediários.

4.3. Extração de Segmentos e Geração de Janelas Virtuais

Consistiu nas seguintes etapas: (i) Conversão das geometrias poligonais das edificações em segmentos lineares, representando fachadas externas; (ii) Remoção de segmentos muito curtos ou inválidos; (iii) Cálculo do vetor ortogonal da janela para orientar a geração dos ângulos de visada e encontrar janelas em pavimentos altos que não possuem visibilidade para árvores no nível da rua; (iv) Interpolação de pontos (janelas virtuais) ao longo dos segmentos em intervalos de 5 metros (seguindo metodologia por [Croeser et al. 2024]; (v) Remoção de janelas cegas geradas em paredes conjugadas com outras edificações.

4.4. Linhas de Visada e aplicação da Regra 3

Consistiu nas seguintes etapas: (i) Cálculo de linhas de visada Linestring entre cada janela virtual e árvores (ST_DWithin); (ii) Classificação das residências que atendem à regra 3; (iii) Contagem final agregada ao nível de edificação; e (iv) Persistência dos resultados em formato Apache Parquet otimizado para desempenho e projetado para grandes volumes de dados analíticos.

5. Resultados: cálculo da Regra 3

Para melhor representação dos resultados em cada etapa, realizou-se análises para uma amostra das edificações no Bairro Meireles. A (Figura 1) ilustra o passo para gerar janelas e filtrar pontos inválidos, fundamental para representar com veracidade as edificações em Fortaleza. Ao segmentar as paredes em intervalos de 5 metros, totalizou-se 3.936 janelas, das quais 2.970 (75%) são inválidas.

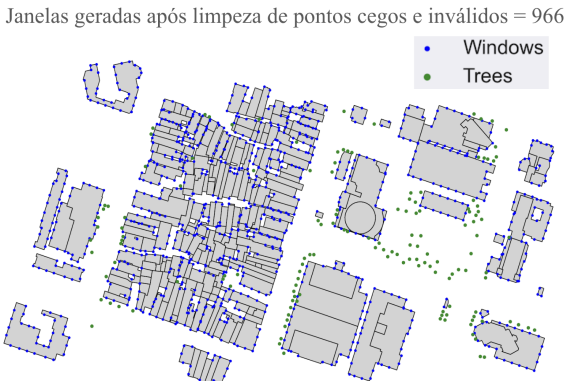

Figura 1. Janelas geradas em cada face das edificações depois da remoção de janelas cegas simuladas em paredes conjugadas, totalizando 966 válidas.

A seguir foram calculadas linhas de visada entre cada janela válida e todas as árvores em um raio de 30 metros simulando a visibilidade da edificação às árvores (Figura 2). Ao final, as residências foram avaliadas quanto à presença de pelo menos 3 linhas de visada conectando janelas a árvores únicas. É atribuído a cada polígono um valor correspondente ao número de linhas de visada com menos de 30 metros de comprimento, indicando as edificações que atendem à regra 3 ou não (Figura 3).

A operação de cálculo de distâncias entre centroides de edificações e árvores próximas dentro de 30 metros apresentou tempo estimado de execução de 217,7min com

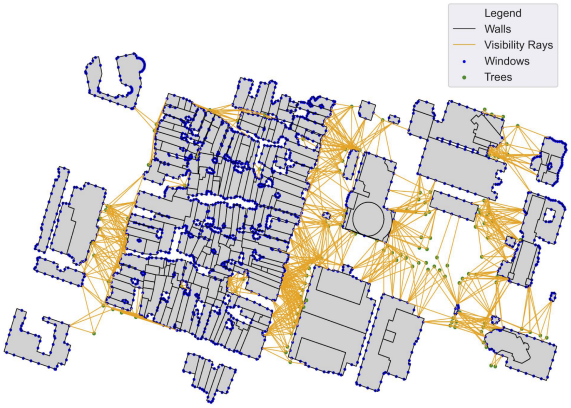

Figura 2. Linhas de visada em amarelo conectando todas as janelas a todas as árvores mais próximas em um raio de 30 metros.

Figura 3. Resultado final mostrando edifícios que atendem (verde) ou não (vermelho) à regra 3 de visibilidade mínima à árvores.

GeoPandas e 80,6min com Spark. A memória total projetada para processar 2 milhões de residências foi de cerca de 7,96GB com GeoPandas e 6,85GB com Spark. Esses resultados destacam a superioridade do Spark mesmo em tarefas leves, com menor risco de falha em etapas posteriores.

6. Comparando funcionalidades Geopandas e Apache Spark + Sedona

As principais distinções entre o Geopandas e Apache Spark evidentes a partir da pesquisa estão descritas a seguir.

Modularização Escalável e Distribuída: No GeoPandas, operações espaciais são executadas localmente via métodos da biblioteca `shapely`. As operações são feitas de forma sequencial e centralizada sendo ineficaz para datasets com mais de poucos milhões de registros. Em contraste, a modularização no ambiente Spark + Sedona permite que cada etapa do processamento (leitura, limpeza, enriquecimento, geração de segmentos, janelas e análises) seja executada de forma paralela e distribuída. No Apache Sedona, essas operações são efetuadas por funções espaciais padrão SQL — como `ST_Intersects`, `ST_Distance` e `ST_Contains` — que operam de forma distribuída sobre bilhões de geometrias. Integradas nativamente ao mecanismo paralelo do Spark, essas funções permitem que buffers, dissoluções, interseções e até mesmo os custosos *joins* espaciais sejam realizados com alta eficiência.

Persistência Nativa, Automática e Otimizada de Intermediários: Spark + Sedona oferecem persistência de dados intermediários onde cada etapa tem seus resultados salvos em formatos eficientes (como `Parquet` com compressão `Snappy`). Isso permite o reprocessamento parcial, sem necessidade de repetir etapas anteriores, além de facilitar a depuração com inspeção dos outputs intermediários. No GeoPandas, a persistência é manual, propensa a erros e não escalável.

Documentação Rigorosa por Manifestos: A documentação estruturada por arquivos de manifesto (`manifest.json`), registra os parâmetros da execução (CRS, espaçamentos, ângulos, buffers) para garantir o determinismo das operações, análises estatísticas detalhadas (número de batches, tempo de execução, total de geometrias) para a depuração dos resultados, caminhos, nomes dos arquivos e hashes de integridade para a rastreabilidade das operações, além de logs resumidos de sucesso, alertas e erros críticos para a depuração de erros e auditoria de resultados.

A migração de uma implementação em Geopandas para Spark+Sedona apresenta desafios. O Spark necessita de um setup inicial mais trabalhoso e sua documentação ainda é superficial. Dessa forma, o Geopandas se mostra uma alternativa mais rápida para prototipagem inicial com um conjunto menor de dados. No entanto, percebe-se a necessidade de migrar para Spark para ganhar escala.

7. Conclusões

A migração do processamento espacial da biblioteca GeoPandas para a arquitetura distribuída baseada em Apache Spark e Sedona permitiu superar gargalos críticos no tratamento de grandes volumes de dados urbanos. O novo pipeline viabilizou a ingestão, limpeza, transformação, extração de feições, cálculos espaciais complexos e persistência dos resultados para toda a base de edificações do município de Fortaleza. Destacam-se como avanços o processamento em batches que distribui a carga computacional, tornando possível analisar centenas de milhares de edificações até 50x mais rápido [Forrest 2025] e a persistência eficiente, onde os resultados intermediários e finais são comprimidos e armazenados prontos para análise em outras ferramentas (QGIS, PowerBI, etc).

O processo de migração evidenciou algumas limitações importantes. O protótipo atual ainda não utiliza sistemas de arquivos distribuídos (HDFS/S3), limitando o processamento de cidades muito maiores ou de múltiplas cidades em paralelo. Algumas funções de usuário (UDFs) e etapas geométricas carecem de testes sistemáticos automatizados, o que pode dificultar a detecção de falhas em grandes execuções.

A pesquisa apresentada se encontra em andamento, mas o pipeline modular já produz resultados visualizáveis em múltiplos formatos e permite ajustes de parâmetros (ex: tamanho mínimo das áreas verdes, raio de análise, filtros de janelas), adaptação para diferentes cidades ou cenários e adição de novas etapas sem comprometer a robustez do processo. Maiores detalhes sobre o pipeline para sua replicação podem ser acessados no documento *3-30-300: Pipeline Spark + Sedona para Análise de dados Geoespaciais* disponível em: `https://www.overleaf.com/read/yqdynxrxbtcr#90571b`.

Referências

Bai, Y., Yang, Z., Yu, J., Ju, R.-Y., Yang, B., Mas, E., and Koshimura, S. (2024). Flood data analysis on spacenet 8 using apache sedona. *arXiv preprint arXiv:2404.18235*.

Croeser, T., Sharma, R., Weisser, W. W., and Bekessy, S. A. (2024). Acute canopy deficits in global cities exposed by the 3-30-300 benchmark for urban nature. *Nature Communications*, 15(1):9333.

Forrest, M. (2025). Geospatial tools compared: When to use geopandas, postgis, duckdb, apache sedona, and wherobots. Acesso em: 19 jun. 2025.

García-García, F., Corral, A., Iribarne, L., and Vassilakopoulos, M. (2023). Efficient distributed algorithms for distance join queries in spark-based spatial analytics systems. *International Journal of General Systems*, 52(3):206–250.

Konijnendijk, C. C. (2023). Evidence-based guidelines for greener, healthier, more resilient neighbourhoods: Introducing the 3–30–300 rule. *Journal of forestry research*, 34(3):821–830.

Lyon, W., Yu, J., and Sarwat, M. (2025). *Cloud Native Geospatial Analytics with Apache Sedona*. O'Reilly Media, Inc., Sebastopol, CA, first edition edition. ISBN not specified.

Moussa, R. (2021). Scalable analytics of air quality batches with apache spark and apache sedona. In *Proceedings of the 15th ACM International Conference on Distributed and Event-Based Systems*, DEBS '21, page 154–159, New York, NY, USA. Association for Computing Machinery.

Nieuwenhuijsen, M. J., Dadvand, P., Márquez, S., Bartoll, X., Barboza, E. P., Cirach, M., Borrell, C., and Zijlema, W. L. (2022). The evaluation of the 3-30-300 green space rule and mental health. *Environmental research*, 215:114387.

Wyrzykowski, B. and Mościcka, A. (2024). Implementation of the 3-30-300 green city concept: Warsaw case study. *Applied Sciences*, 14(22):10566.

Yu, J., Wu, J., and Sarwat, M. (2015). Geospark: A cluster computing framework for processing large-scale spatial data. In *Proceedings of the 23rd SIGSPATIAL international conference on advances in geographic information systems*, pages 1–4.

Zheng, Y., Lin, T., Hamm, N. A., Liu, J., Zhou, T., Geng, H., Zhang, J., Ye, H., Zhang, G., Wang, X., et al. (2024). Quantitative evaluation of urban green exposure and its impact on human health: A case study on the 3–30-300 green space rule. *Science of the Total Environment*, 924:171461.