

UNIVERSIDADE FEDERAL FLUMINENSE

Ayres Nishio da Silva Junior

**ANÁLISE DE CRITICALIDADES EM
SISTEMAS DE MEDIÇÃO DE GRANDE PORTE
VIA PARTICIONAMENTO E PROGRAMAÇÃO
EM GPU**

NITERÓI

2026

Ayres Nishio da Silva Junior

**ANÁLISE DE CRITICALIDADES EM
SISTEMAS DE MEDIÇÃO DE GRANDE PORTE
VIA PARTICIONAMENTO E PROGRAMAÇÃO
EM GPU**

Tese de Doutorado apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do Grau de Doutor em Computação. Área de concentração: Ciência da Computação

Orientador:

Julio Cesar Stacchini de Souza

Coorientador:

Milton Brown Do Coutto Filho

NITERÓI

2026

Ficha catalográfica automática - SDC/BEE
Gerada com informações fornecidas pelo autor

S586a Silva Junior, Ayres Nishio da
Análise de Criticalidades em Sistemas de Medição de
Grande Porte via Particionamento e Programação em GPU /
Ayres Nishio da Silva Junior. - 2026.
95 f.

Orientador: Julio Cesar Stacchini de Souza.
Coorientador: Milton Brown Do Coutto Filho.
Tese (doutorado)-Universidade Federal Fluminense, Instituto
de Computação, Niterói, 2026.

1. Estimação de Estado. 2. Análise de Criticalidades. 3.
Particionamento de Grafos. 4. Computação Paralela em GPUs.
5. Produção intelectual. I. Souza, Julio Cesar Stacchini de,
orientador. II. Do Coutto Filho, Milton Brown, coorientador.
III. Universidade Federal Fluminense. Instituto de
Computação. IV. Título.

CDD - XXX

Ayres Nishio da Silva Junior

Análise de Criticalidades em Sistemas de Medição de Grande Porte via Particionamento
e Programação em GPU

Tese de Doutorado apresentada ao Programa
de Pós-Graduação em Computação da Uni-
versidade Federal Fluminense como requisito
parcial para a obtenção do Grau de Dou-
tor em Computação. Área de concentração:
Ciência da Computação

Aprovada em Agosto de 2026.

BANCA EXAMINADORA

Prof. Julio Cesar Stacchini de Souza - Orientador, UFF

Prof. Milton Brown Do Coutto Filho - Coorientador, UFF

Prof. Esteban Walter Gonzalez Clua, UFF

Prof. Anselmo Antunes Montenegro, UFF

Prof. Djalma Mosqueira Falcão, COPPE/UFRJ

Prof. Igor Delgado de Melo, UFJF

Niterói

2026

*"Finding it, though, that's not the hard part. It's letting go."— Father Elijah, Fallout
New Vegas*

Agradecimentos

Esta tese marca o fim do doutorado que foi uma fase tão prazerosa e enriquecedora que foi difícil deixar para trás. Agradeço primeiramente aos meus pais, Ana Cristina e Ayres Nishio, pelo amor e suporte em todos os momentos. Aos meus orientadores Milton e Julio por todos os ensinamentos e auxílios. Aos meus colegas de doutorado Vinícius e Rafael por toda a parceria pela caminhada. Ao professor Marcio pela indicação ao mestrado que iniciou a jornada. Ao Fabio Padovani por toda a amizade e parceria durante o trabalho. Ao Cepel e ao INESC tec pelas bolsas e incentivos pela pesquisa.

Resumo

A Estimação de Estado encontra-se entre as principais ferramentas utilizadas na operação de sistemas elétricos de potência, e sua confiabilidade depende diretamente da qualidade do sistema de medição empregado. Nesse contexto, a Análise de Criticalidades permite identificar medidas e conjuntos de medidas cuja indisponibilidade compromete a observabilidade da rede e, por conseguinte, a capacidade de detecção e de identificação de erros grosseiros nesses dados. Entretanto, sua natureza combinatória implica um elevado custo computacional, o que limita sua aplicação em sistemas de grande porte. Esta tese tem como objetivo ampliar a escalabilidade da Análise de Criticalidades por meio da redução de seu custo computacional. Inicialmente, são propostas otimizações para sua implementação em unidades de processamento gráfico, reduzindo o tempo de execução em comparação à abordagem convencional. Em seguida, é apresentada a principal contribuição deste trabalho: uma metodologia inédita baseada no particionamento topológico do sistema de medição, que explora a natureza local das criticalidades para decompor o problema original em subproblemas menores e mais eficientes computacionalmente. A metodologia representa o sistema de medição por meio de um grafo e emprega técnicas de particionamento, complementadas por uma estratégia de sobreposição entre partições vizinhas para aprimorar a identificação de criticalidades nas regiões de fronteira. A abordagem proposta é avaliada em sistemas IEEE de 118 barras e no sistema polonês de 2.383 barras. Os resultados demonstram que a implementação em GPU supera, em desempenho, os principais métodos sequenciais da literatura e que a metodologia de particionamento reduz significativamente o número de combinações avaliadas e o tempo computacional, preservando elevadas taxas de identificação de criticalidades. Os experimentos também evidenciam que as criticalidades apresentam natureza predominantemente local, o que favorece a estratégia proposta.

Palavras-chave: Estimação de Estado; Análise de Criticalidades; Sistemas de Medição; Particionamento de Grafos; Computação Paralela em GPUs.

Abstract

Power System State Estimation (SE) is a fundamental application used in power system operation, and its reliability strongly depends on the quality of the underlying measurement system. In this context, Criticality Analysis identifies measurements and measurement groups whose unavailability compromises network observability and the SE's ability to detect and identify gross measurement errors. However, the combinatorial nature of Criticality Analysis results in high computational cost, limiting its application to large-scale measurement systems. This thesis aims to improve the scalability of Criticality Analysis by reducing its computational complexity. The primary contribution is a novel topology-based partitioning methodology that leverages the local nature of criticalities to decompose the original problem into smaller and computationally efficient subproblems. The proposed methodology models the measurement system as a graph and applies graph partitioning techniques combined with an overlap strategy between neighboring partitions to improve the identification of criticalities near partition boundaries. In addition, optimizations for the parallel implementation of Criticality Analysis on Graphics Processing Units (GPUs) are proposed to further reduce the execution time of the conventional approach. The methodology is validated using the IEEE 118-bus and the Polish 2,383-bus systems. Experimental results demonstrate that the optimized GPU implementation outperforms state-of-the-art sequential approaches, while the proposed partitioning methodology significantly reduces both the number of evaluated combinations and the computational time without compromising criticality identification rates. The results also provide evidence that critical measurement sets exhibit predominantly local behavior, supporting the proposed partitioning strategy.

Keywords: Power System State Estimation; Criticality Analysis; Topology-Based Partitioning; Measurement Systems; GPU Parallel Computing.

Lista de Figuras

1	Fluxograma de atuação da EE no SGE	13
2	Etapas da EE.	22
3	Linhas e colunas da matriz $\Omega_{(AB)}^{(2)}$ associadas a uma combinação de UMs. .	34
4	Exemplo Ilustrativo da AC de UMs.	35
5	Representação do fluxo da execução e dados para a aplicação da AC em GPUs.	37
6	Etapas da AC e fluxo de dados na implementação em GPU.	40
7	Etapas da AC e fluxo de dados na implementação em GPU.	44
8	Etapas da AC e fluxo de dados na implementação em GPU.	46
9	Tempo de execução em função da cardinalidade (k).	50
10	Sistema exemplo de 6 barras e 9 medidas.	53
11	Multigrafo de medição do sistema exemplo de 6 barras. Nesta modelagem, os vértices representam as barras do sistema e as arestas representam as medidas do plano.	56
12	Grafo da topologia do sistema exemplo de 6 barras.	57
13	Particionamento ótimo do sistema exemplo de 6 barras obtido por meio da formulação de programação inteira.	61
14	Árvore geradora do sistema exemplo de 6 barras. Vértices folha 2 e 6 destacados para inicialização do particionamento.	63
15	Fluxograma do algoritmo de particionamento do Esquema H.	65
16	Particionamento final obtido pelo Esquema H.	66
17	Matriz de adjacências $\mathbf{A}$ construída para o sistema exemplo de 6 barras para a execução do Esquema S.	67

18	Fluxograma da abordagem proposta da AC particionada.	71
19	Grupos de medidas formados pelo particionamento do sistema exemplo de 6 barras.	72
20	Exemplo de C_4 não identificada no sistema IEEE 118 barras.	79

Lista de Tabelas

1	Comparação dos tempos de execução e fatores de aceleração obtidos para diferentes plataformas computacionais.	48
2	Comparação entre o Algoritmo B&B e o Algoritmo Proposto em GPU . . .	50
3	Matriz de adjacências de medidas $\mathbf{M}$ do sistema exemplo de 6 barras. . . .	70
4	Representação das regiões Ω_1 e Ω_2	72
5	Distribuição das criticalidades identificadas em função da distância máxima.	77
6	Taxa de Identificação de criticalidades (%) — IEEE 118 Barras	78
7	Desempenho dos Esquemas de Particionamento — Sistema de Medição IEEE 118 Barras (C_k com $k \leq 4$)	81
8	Desempenho dos Esquemas de Particionamento — Sistema de Medição Polonês de 2.383 Barras (C_k com $k \leq 4$)	82

Sumário

1	Introdução	12
1.1	Considerações Gerais	12
1.2	Revisão Bibliográfica	15
1.3	Objetivo	19
1.4	Descrição dos Capítulos	19
1.5	Publicações	20
2	Análise de Criticalidades	21
2.1	Estimação de Estado	21
2.1.1	Etapas	21
2.1.2	Conceitos Básicos	23
2.1.3	Análise de Observabilidade	27
2.2	Análise de Criticalidades via Matriz de Covariância de Resíduos	29
2.2.1	Propriedades para a Identificação de Criticalidades	30
2.2.2	Etapas da Análise de Criticalidades	32
2.2.3	Criticalidades de Unidades de Medição	33
2.2.4	Complexidade da Análise de Criticalidades	34
2.3	Programação paralela em GPUs	35
2.3.1	Paradigmas da programação em GPU	36
2.4	Implementação da Análise de Criticalidades na GPU	37
2.4.1	Primeira Versão: Duas etapas executadas na GPU	40
2.4.1.1	Adaptação da Etapa 2: Poda	41

2.4.1.2	Adaptação da Etapa 3: Avaliação	42
2.4.2	Segunda Versão: Quatro etapas executadas na GPU	43
2.4.2.1	Adaptação da Etapa 1: Enumeração	43
2.4.2.2	Adaptação da Etapa 4: de Atualização	45
2.4.3	Comparação de desempenho entre AC na CPU e na GPU	47
2.4.4	Comparação com Branch & Bound	49
3	Estratégias de Particionamento	52
3.1	Metodologia	52
3.2	Definição do Problema de Particionamento	53
3.3	Modelagem via Grafos	55
3.3.1	Multigrafo Grafo do SM	55
3.3.2	Grafo da Topologia	56
3.3.3	Formulação via Programação Inteira	57
3.3.3.1	Variáveis de decisão	58
3.3.3.2	Função objetivo	58
3.3.3.3	Restrição de unicidade de grupos	59
3.3.3.4	Restrições de conectividade	59
3.3.3.5	Complexidade da Formulação	60
3.4	Algoritmos Heurísticos de Particionamento	62
3.4.1	Heurístico baseado em árvore geradora	62
3.4.1.1	Primeira etapa: Identificação das folhas	63
3.4.1.2	Segunda etapa: Particionamento	63
3.4.1.3	Terceira etapa: Balanceamento das partições	63
3.4.2	Clusterização Espectral	66
3.4.3	Modularidade Gulosa	68
3.4.4	K-means	68

3.5	Análise de Criticalidades Particionada	70
3.5.1	Metodologia para AC Particionada	70
3.5.2	Exemplo ilustrativo	72
4	Resultados Numéricos	75
4.1	Descrição da Simulação	75
4.2	Avaliação dos Métodos de Particionamento para AC	76
4.3	Simulação do Sistema IEEE de 118 Barras	77
4.3.1	Adjacência de medidas participantes de criticalidades	77
4.3.2	Taxa Média de Identificação de Criticalidades	78
4.3.3	Exemplo de Criticalidade não identificada	79
4.3.4	Avaliação de Desempenho	80
4.4	Simulação do Sistema Polonês de 2.383 Barras	81
4.5	Considerações Finais	83
5	Conclusões e Trabalhos Futuros	84
5.1	Principais Contribuições	85
5.2	Trabalhos Futuros	86
	REFERÊNCIAS	89

1 Introdução

1.1 Considerações Gerais

A operação dos sistemas elétricos de potência modernos é realizada em Centros de Operação de Sistemas (COS), com o auxílio de ferramentas computacionais integradas em um Sistema de Gerenciamento de Energia (SGE) ([Ali Abur, 2004](#)). Entre as principais funções do SGE, destaca-se a Estimação de Estado (EE), responsável por determinar o estado operativo mais provável da rede elétrica com base nas medidas fornecidas pelo Sistema de Medição (SM).

Em um regime de operação caracterizado pelo equilíbrio entre carga e geração, o estado operativo é definido pelas tensões nodais complexas, correspondentes a uma determinada configuração da rede elétrica ([Augusto, 2016](#)). Como essas grandezas não são usualmente observadas diretamente, sua estimação é realizada a partir de um conjunto redundante de medidas coletadas por diferentes unidades de medição (UMs), incluindo Unidades Terminais Remotas (UTRs), Dispositivos Eletrônicos Inteligentes (DEIs) e Unidades de Medição Fasorial (UMFs). Essas unidades fornecem medidas convencionais de fluxo e de injeção de potência, magnitudes de tensão e, no caso das UMFs, sincrofasores de tensão e de corrente.

As medidas coletadas são concentradas pelo sistema SCADA (*Supervisory Control and Data Acquisition*) e encaminhadas ao Estimador de Estado. A partir dessas medições, o estimador obtém a melhor estimativa do estado operativo da rede, atenuando os efeitos dos erros aleatórios inerentes ao processo de medição e possibilitando a detecção e identificação de erros grosseiros (EGs). O estado estimado é então utilizado como entrada para diversas funções do SGE, incluindo, por exemplo, a análise de contingências, o fluxo de potência ótimo e o despacho econômico, conforme ilustrado na Figura 1.

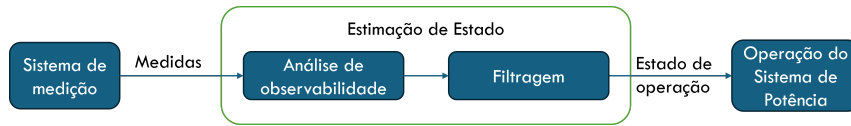


Figura 1: Fluxograma de atuação da EE no SGE

A EE é normalmente formulada pelo método dos mínimos quadrados ponderados (MQP) (Ali Abur, 2004), cujo desempenho depende diretamente da quantidade, do tipo e da distribuição das medidas disponíveis. Dessa forma, um SM adequado deve garantir observabilidade, qualidade da estimação, confiabilidade na detecção de erros grosseiros e robustez frente à indisponibilidade de medidas (Augusto, 2016). Entretanto, restrições econômicas, falhas de comunicação, ataques cibernéticos, alterações topológicas e indisponibilidades de equipamentos podem comprometer esses requisitos e reduzir a confiabilidade do processo de estimação.

Tradicionalmente, uma etapa inicial do estimador, denominada análise de observabilidade, verifica apenas se o conjunto de medidas disponível é suficiente para que o estado da rede possa ser estimado, produzindo uma resposta essencialmente binária: observável ou não observável. Embora essa informação seja indispensável, ela não permite avaliar o grau de vulnerabilidade do SM nem sua capacidade de manter a observabilidade e a detecção de erros grosseiros diante da perda de medidas.

Nesse contexto, a Análise de Criticalidades (AC) amplia a avaliação do SM ao identificar medidas ou conjuntos de medidas cuja indisponibilidade compromete a observabilidade da rede ou a capacidade de detecção e identificação de erros grosseiros. Assim, além de caracterizar a vulnerabilidade do SM, a AC fornece informações importantes para o planejamento de novos sistemas, estudos de confiabilidade e aplicações relacionadas à segurança operacional.

Entretanto, a identificação de criticalidades constitui um problema de natureza combinatória. O número de combinações de medidas a serem avaliadas cresce rapidamente com o tamanho do sistema e com a cardinalidade das criticalidades de interesse, tornando o custo computacional proibitivo em sistemas de grande porte.

Embora a AC seja um procedimento *offline*, tipicamente empregado na fase de projeto de sistemas de medição, os ganhos de desempenho são de grande interesse, pois permitem a análise de instâncias maiores do problema e viabilizam o uso da AC como subprocesso em outras aplicações (Braga Flor, V. et al., 2022). Contudo, apesar de sua importância, a aplicação prática da AC é severamente limitada pelo crescimento expo-

nencial do número de combinações de medidas a serem avaliadas. Mesmo abordagens recentes baseadas em algoritmos Branch-and-Bound (Augusto; Do Coutto Filho *et al.*, 2019) e em processamento paralelo em GPU (Nishio; Clua *et al.*, 2020) permanecem limitadas quando aplicadas a sistemas de medição de grande porte. Dessa forma, torna-se necessário investigar novas estratégias capazes de reduzir a dimensão efetiva do problema sem comprometer a qualidade da identificação das criticalidades.

Primeiramente, esta tese de doutorado propõe aprimorar a implementação paralela da AC em GPU, explorando as características da arquitetura paralela para maximizar o desempenho computacional. A AC apresenta alto potencial de paralelização, uma vez que a análise de cada combinação de medidas pode ser realizada de forma independente. Além disso, GPUs são projetadas para executar tarefas de baixo esforço computacional e altamente paralelizáveis, o que as torna uma escolha natural para acelerar a execução da AC.

Apesar dos resultados satisfatórios obtidos, a natureza combinatória do problema de AC continua sendo intratável em larga escala, o que ainda inviabiliza a análise de sistemas reais de grande porte.

A principal contribuição desta tese consiste em uma metodologia inédita de particionamento topológico do SM que reduz significativamente o número de combinações avaliadas pela AC. A metodologia explora a natureza local das criticalidades para decompor o problema original em subproblemas menores, permitindo ampliar significativamente a escalabilidade da AC.

Entre os métodos mais utilizados para o particionamento de redes elétricas (Zhang; Miao; Fan, 2018), quatro foram selecionados para compor a metodologia proposta para a AC particionada: (a) particionamento heurístico de grafos; (b) particionamento espectral clássico; (c) agrupamento *k-means*; e (d) detecção aglomerativa de comunidades baseada em modularidade gulosa (*greedy modularity*).

Embora os métodos (b), (c) e (d) sejam abordagens gerais amplamente conhecidas, sua aplicação a problemas de AC requer contextualização e adaptação adequadas. Os métodos de particionamento adotados são avaliados nos sistemas de potência IEEE de 118 barras e no sistema polonês de 2.383 barras, utilizados como prova de conceito, e apresentam resultados expressivos.

1.2 Revisão Bibliográfica

A análise de observabilidade para EE foi amplamente investigada na literatura, destacando os trabalhos clássicos de [Ali Abur \(2004\)](#) e [Monticelli \(2012\)](#). Existem duas abordagens principais para o problema de análise de observabilidade: a topológica, que se fundamenta na teoria de grafos para realizar uma análise puramente estrutural, e a numérica, que se baseia nas propriedades das matrizes do processo de estimação.

Dentre as abordagens topológicas, os principais trabalhos foram desenvolvidos por [Krumpholz, Clements e Davis \(1980\)](#), [K. Clements, G. Krumpholz e P. Davis \(1981\)](#) e [K. A. Clements, G. R. Krumpholz e P. W. Davis \(1982\)](#) e [Clements, Krumpholz e Davis \(1983\)](#). Particularmente no primeiro trabalho de [Krumpholz, Clements e Davis \(1980\)](#) foi estabelecida a análise residual do processo de estimação, levando em consideração a topologia da rede. Outras análises baseadas puramente na teoria de grafos para avaliação da observabilidade podem ser encontradas no trabalho de [Quintana, Simoes-Costa e Mandel \(1982\)](#), e notadamente na pesquisa de [Mori e Tsuzuki \(1991\)](#), em que se adotou o conceito de árvores geradoras mínimas (*minimum spanning trees*). Por fim, o trabalho de [Nucera e Gilles \(1991\)](#) faz uso de conceitos de otimização combinatória e de sequências ampliadas para a determinação de observabilidade de uma rede.

A abordagem topológica apresenta uma implementação computacional elaborada e requer um embasamento teórico mais complexo. Desta forma, trabalhos voltados à análise de observabilidade numérica foram desenvolvidos alternativamente. Os trabalhos de [Monticelli e Wu \(1985\)](#) e [Wu e Monticelli \(1985\)](#) abordam os conceitos de ilha observável e de ramo não observável, explorando também algoritmos para a avaliação da observabilidade pela fatoração triangular da matriz de Ganho. O mau condicionamento numérico desta matriz, motivou o desenvolvimento de uma nova abordagem baseada na transformação ortogonal da matriz Jacobiana ([Monticelli; Wu, 1986](#)). A partir de propriedades de alocação das medidas de fluxo e injeção e a topologia da rede, o algoritmo de [Chen \(1990\)](#) manipula apenas números inteiros na matriz Jacobiana de medidas, o que torna o processo de análise de observabilidade mais simples e ágil. [Gou e Abur \(2000\)](#) apresentam em seu trabalho uma análise algébrica que utiliza fatores triangulares da matriz de Ganho, tanto para a determinação de ilhas observáveis quanto para a determinação de quais pseudo-medidas devem ser inseridas para restaurar a observabilidade da rede. [Castillo et al. \(2005\)](#) apresentam uma técnica eficiente que se baseia no cálculo do espaço nulo da matriz Jacobiana. Já [Almeida, Asada e Garcia \(2008\)](#) utilizam da matriz de Gram, calculada a partir da Jacobiana, para realizar análise e verificar redundância. [Gou \(2006\)](#)

realiza a análise por meio do método da eliminação de Gauss para tornar o processo mais simples e eficiente. Por fim, [Solares *et al.* \(2009\)](#) também utilizam o método de Gauss, porém, aplicam a aritmética binária para evitar erros de arredondamento.

A capacidade de observação de um SM e a depuração de erros grosseiros estão intimamente relacionadas à presença de criticalidades. O primeiro trabalho relacionado a identificação generalizada de k -tuplas críticas foi proposto por [Clements e Davis \(1986\)](#), que apresentam a definição de C_{ks} de medidas e suas principais propriedades, como também, propõe um método geométrico baseado em geometria analítica para identificação das criticalidades.

Existem dois trabalhos desenvolvidos por Clements que utilizavam abordagem topológica para a identificação de medidas críticas (C_{meds}). O primeiro trabalho de [K. Clements, G. Krumpholz e P. Davis \(1981\)](#) utiliza o conceito de árvore geradora, para identificar C_{meds} e regiões de propagação de erro. O segundo, apresentado por [Clements, Krutnpholz e Davis \(1983\)](#), dá continuidade ao trabalho anterior e avalia a inclusão de pseudo-medidas no conjunto de medição. [Simoes Costa, Piazza e Mandel \(1990\)](#) estenderam o trabalho de [Quintana, Simoes-Costa e Mandel \(1982\)](#), e propuseram a utilização de matroides e de árvores geradoras para a determinação de medidas críticas e conjuntos críticos. Mais recentemente, a abordagem topológica voltou a ser abordada pelo trabalho de [Soares Lima *et al.* \(2022\)](#), nele o SM é modelado como multigrafo capaz de representar tanto as características de observabilidade quanto as de criticalidades do sistema.

Dentre as abordagens numéricas, [Ayres e Haley \(1986\)](#) utilizam a análise dos resíduos normalizados e dos elementos da matriz de covariância para identificar C_{conj} . O trabalho de [Exposito e Abur \(1997\)](#) apresentam um método generalizado de observabilidade que também é capaz de classificar C_{meds} e medidas redundantes. [Milton B.. Do Coutto Filho *et al.* \(2001\)](#) e [Do Coutto Filho, Stacchini de Souza e Schilling \(2007\)](#) desenvolveram um método que utiliza propriedades dos resíduos normalizados das medidas para a determinação de C_{meds} e C_{conj} . Outro trabalho relevante no sentido de determinação de tuplas de baixas cardinalidades foi realizado por [Do Coutto Filho, Stacchini de Souza e Augusto \(2014\)](#), no qual foi apresentada uma abordagem da AC considerando a situação mais provável, em que ocorre a indisponibilidade de uma UM contendo várias medidas.

Além das análises topológicas e numéricas, há também as análises híbridas que combinam a teoria dos grafos com métodos numéricos. O algoritmo numérico-simbólico de classificação de C_{meds} e C_{conj} apresentado por [Korres e Contaxis \(1991\)](#) emprega os conceitos de ilhas de fluxo, áreas de propagação residual e modelo reduzido da rede, para

isolar erros em menores regiões do sistema, possibilitando uma redução na busca por EGs.

Conforme destacado intensamente, devido à natureza combinatória do problema em estudo a quantidade de trabalhos relacionados a AC em cardinalidades elevadas ($k > 3$) é relativamente pequena. Contudo, diversos equipamentos da rede elétrica (medidores inteligentes) começaram a utilizar sistemas de comunicação digital de dados, tornando sistemas vulneráveis a ataques cibernéticos (He; Yan, 2016). O primeiro caso de um ataque desta natureza bem-sucedido em redes de potência pode ser encontrado em Zetter (2016). A possibilidade de invasores corromperem intencionalmente as medidas coletadas para a EE estimula o estudo de cardinalidades mais altas.

Considerando o perigo de ataques cibernéticos e a vulnerabilidade do SM pela presença de C_k -tuplas, Sou, Sandberg e Johansson (2012) propuseram um método de programação linear inteira mista para identificação de C_{ks} de medidas (as mais esparsas, i.e., de menor k possível) nas quais uma medida pré-estabelecida pertença. Posteriormente, Augusto, Filho. e Stacchini de Souza (2013) desenvolveram uma estratégia para identificar C_{ks} para $k > 3$, a partir dos resultados obtidos das cardinalidades inferiores. Em sequência, Augusto (2016) desenvolveu uma abordagem pela heurística de *branch-and-bound* (B&B) para enumerar de forma eficiente as combinações de medidas e identificar C_{ks} , para qualquer valor de k , seja de medidas, unidades de medição ou ramos da rede. Dando continuidade a este trabalho, Augusto, Do Coutto Filho *et al.* (2019) aprofundaram a abordagem de B&B para criticalidades de UMs.

Aprimorar o desempenho da AC tem sido uma linha de pesquisa recente, e por esta razão, existem poucos trabalhos publicados nesta direção. Entre eles, o artigo de Vinícius B. Braga Flor *et al.* (2021) aprimorou a implementação do método B&B na AC (Augusto; Do Coutto Filho *et al.*, 2019), explorando diferentes estratégias de busca, estruturas de dados e funções objetivo. A partir da enumeração exaustiva (força bruta), o trabalho de Nishio, Clua *et al.* (2020) utilizou de unidades de processamento gráfico (GPUs) para identificar paralelamente as criticalidades do sistema. Em seguida, o trabalho de Nishio, Do Coutto Filho, Stacchini De Souza *et al.* (2024) faz as adaptações necessárias para adaptar o método de GPU para realizar a AC de UMs e possibilitar uma comparação justa com outros métodos encontrados na literatura, incluindo o método de B&B. Os resultados obtidos demonstram que o método de GPU é o mais eficiente para a AC em sistemas de larga escala. Apesar dos avanços de desempenho alcançados pelos trabalhos anteriores, a natureza do problema permanece intratável, o que torna impraticável a realização da

análise em instâncias maiores. Para contornar esta limitação, este trabalho busca utilizar técnicas de particionamento para segmentar o problema em instâncias menores.

Técnicas de particionamento desempenham um papel crucial na operação, no controle e no gerenciamento de redes elétricas. Tradicionalmente, o particionamento da rede baseia-se em regiões geográficas, áreas de controle e experiência operacional. Diversos métodos amplamente utilizados são discutidos em [Massaoudi et al. \(2025\)](#), destacando seus pontos fortes e limitações. Os benefícios geralmente envolvem eficácia em determinados tipos de agrupamento, eficiência computacional, escalabilidade ou adequação a estruturas específicas da rede ([Guggari; V; Kadappa, 2025](#)). Por outro lado, as desvantagens frequentemente incluem alta demanda computacional, sensibilidade à escolha dos parâmetros e capacidade limitada de lidar com determinadas características dos dados.

Dentre os trabalhos relacionados ao particionamento de sistemas de potência, podem-se destacar os métodos hierárquicos que solucionam sistemas de grande porte "por partes" (tradução do inglês *piecewise*) ([Montenegro; Ramos; Bacha, 2016](#)). O trabalho de [Kuri, Gagnaire e Puech \(2005\)](#) apresenta um algoritmo, baseado na meta-heurística de recozimento simulado, capaz de dividir um modelo de rede de potência em várias sub-redes para otimizar o uso de sistemas computacionais paralelos para análise da rede. O trabalho de [Ding, El-Keib e Smith \(1994\)](#) utiliza algoritmos genéticos para decomposição da rede. Além destes, o trabalho de [Luccio e Sami \(1969\)](#) busca dividir a rede em diferentes grupos de componentes elétricos, interconectados por um número mínimo de condutores.

Diferentemente do que se encontra na literatura, este trabalho busca particionar o SM para ampliar os limites da AC. Com o sistema dividido em instâncias menores, a identificação de todas as criticalidades, referentes ao sistema completo, poderá ser obtida através da integralização dos resultados das análises dos subsistemas. Um particionamento eficaz é essencial para otimizar o desempenho de algoritmos paralelos e viabilizar o uso eficiente dos recursos computacionais ([Ayall et al., 2022](#)). As técnicas de particionamento incluem tanto algoritmos exatos, que encontram a partição ótima, quanto métodos heurísticos, que produzem soluções aproximadas rapidamente. Encontrar a partição ótima por meio de algoritmos exatos (por exemplo, programação dinâmica e *Branch and Bound* – B&B) é um problema NP-difícil, cuja complexidade cresce rapidamente à medida que aumenta o número de vértices/arestas no grafo do MS. Métodos exatos de particionamento de grafos são práticos apenas em casos limitados e, portanto, não são considerados neste trabalho. Alternativamente, dentre os métodos heurísticos, este trabalho considera quatro técnicas de particionamento que são amplamente utilizadas em redes de potência

(Zhang; Miao; Fan, 2018): particionamento heurístico de grafos; particionamento espectral clássico; agrupamento *k-means*; e detecção de comunidades baseada em modularidade gulosa.

1.3 Objetivo

A natureza combinatória da AC é o principal fator limitante à sua aplicação em sistemas de medição de grande porte. Nesse contexto, o objetivo principal desta tese é desenvolver uma metodologia baseada no particionamento topológico do SM, capaz de decompor o problema original em subproblemas menores e, conseqüentemente, ampliar a escalabilidade da AC. A metodologia proposta explora as características topológicas do SM para realizar a análise de forma independente em diferentes subsistemas. Ao final do processo, as criticalidades identificadas em cada partição são integradas e refinadas, de modo a reconstruir as criticalidades do sistema como um todo.

Como contribuição complementar, esta tese também propõe otimizações para a implementação paralela da AC em unidades de processamento gráfico (GPUs), dando continuidade aos trabalhos anteriores baseados em Branch-and-Bound (Augusto; Do Coutto Filho *et al.*, 2019) e processamento paralelo (Nishio; Clua *et al.*, 2020). As otimizações desenvolvidas permitem um melhor aproveitamento da arquitetura paralela das GPUs, resultando em desempenho superior ao dos principais métodos sequenciais encontrados na literatura. Entretanto, embora tais otimizações reduzam significativamente o tempo de execução, elas não eliminam a natureza combinatória do problema, que continua a limitar a aplicação da AC em sistemas de grande porte. Os resultados apresentados nesta tese demonstram que a metodologia de particionamento proposta supera essa limitação, reduzindo substancialmente o custo computacional da análise e ampliando os limites práticos de sua aplicação.

1.4 Descrição dos Capítulos

A presente tese está organizada da seguinte forma:

O Capítulo 1 introduz o problema da EE contextualizando sua aplicação na operação de sistemas de potência, bem como apresenta a AC, justificando a necessidade da segmentação do problema. Além disso, o capítulo também realiza uma revisão bibliográfica das análises de observabilidade, de criticalidades e de segmentação de redes de potência. O

Capítulo 2 apresenta os principais conceitos referentes à EE e AC, como também descreve o algoritmo paralelo, utilizado neste trabalho para realizar a AC e sua implementação na GPU. No Capítulo 3 é descrita a metodologia proposta para a segmentação de planos de medição. Em seguida, o Capítulo 4 apresenta e comenta os resultados consolidados obtidos pela metodologia desenvolvida. Finalmente, o Capítulo 5 conclui o trabalho, discutindo os resultados e apresentando sugestões de implementações para trabalhos futuros.

1.5 Publicações

Este trabalho dá continuidade à dissertação de mestrado de [Nishio \(2021\)](#), que originou a publicação de [Nishio, Clua *et al.* \(2020\)](#). Nesse estudo, foi realizada uma adaptação parcial do método de AC via matriz de covariâncias para execução em GPUs. Posteriormente, no artigo de [Nishio, Do Coutto Filho, Stachinni de Souza *et al.* \(2024\)](#), as quatro etapas da AC via matriz de Covariâncias foram adaptadas para implementação em GPUs, possibilitando ganhos significativos de desempenho e viabilizando sua aplicação em sistemas de maior porte. Na sequência, o trabalho apresentado de [Nishio, Do Coutto Filho, Stacchini De Souza *et al.* \(2024\)](#), feito em conjunto com o INESC TEC, faz as adaptações necessárias para adaptar o método de GPU para realizar a AC de UMs e possibilitar uma comparação justa outros métodos encontrados na literatura, incluindo o método de B&B. Os resultados obtidos comprovam que o método de GPU é o mais eficiente para a AC em sistemas de larga escala.

Devido aos resultados expressivos, essa metodologia foi empregada no trabalho de [Vinícius Braga Flor *et al.* \(2022\)](#), no qual o aumento de desempenho da AC viabilizou sua utilização como sub-processo no problema de alocação ótima de UMs. Além das contribuições voltadas ao aprimoramento computacional da AC, durante o doutorado o grupo de pesquisa também desenvolveu uma ferramenta educacional para auxiliar o ensino de EE [Milton B. Do Coutto Filho *et al.* \(2023\)](#). Essa ferramenta inclui módulos voltados ao ensino de análise de observabilidade, AC e fluxo de potência. Por essa razão, a metodologia de AC desenvolvida neste trabalho também foi incorporada à ferramenta, que é disponibilizada gratuitamente para a comunidade acadêmica.

2 Análise de Criticalidades

Este capítulo apresenta os conceitos fundamentais da Estimação de Estado (EE) e da Análise de Criticalidades (AC) que servem de base para a metodologia proposta nesta tese. Inicialmente, são revisados os principais aspectos da EE, incluindo suas etapas, a análise de observabilidade e o modelo linear adotado. Em seguida, são apresentados os conceitos relacionados à AC, bem como a implementação paralela em GPU empregada neste trabalho. Por fim, são discutidos os resultados que motivaram a adoção da implementação paralela na metodologia proposta.

2.1 Estimação de Estado

A EE realiza a tarefa de filtrar medidas cruas coletadas da rede elétrica para estimar o estado mais provável de operação do sistema, a ser utilizado por aplicativos computacionais que integram o SGE. O trabalho de [Schweppe e Wildes \(1970\)](#) foi o primeiro a abordar o problema da EE em sistemas de potência. Os principais conceitos utilizados neste capítulo encontram-se detalhados por [Ali Abur \(2004\)](#) e [Monticelli \(2012\)](#).

2.1.1 Etapas

Convencionalmente, a EE inclui quatro etapas: pré-processamento; configuração da rede; análise de observabilidade; filtragem e processamento dos erros grosseiros. A seguir, tais etapas são brevemente descritas:

- **Pré-processamento:** Verifica a consistência inicial das medições e do estado dos equipamentos de chaveamento, eliminando erros grosseiros evidentes e corrigindo inconsistências de configuração.
- **Configurador de rede:** Processa os dados de *status* de chaves e disjuntores para construir o modelo barra-ramo representativo da rede.

- **Análise de Observabilidade:** Verifica se a existência de medidas disponíveis em quantidade e posicionamento que permitam a obtenção da solução da EE, que se estende por rede observável. Identifica também ramos não observáveis e ilhas observáveis no sistema, caso existam.
- **Filtragem:** Considerada como núcleo ou principal etapa do processo de EE, esta etapa filtra erros de baixa magnitude, determinando uma estimação otimizada para o estado mais provável da rede elétrica, usualmente através do método dos mínimos quadrados ponderados (MQP).
- **Processamento de Erros Grosseiros:** Confronta as medidas estimadas pela etapa anterior com aquelas oriundas do sistema de medição, e define os chamados resíduos da estimação, com o objetivo de detectar/identificar a presença de erros grosseiros (EGs). É importante destacar que esta etapa requer uma redundância suficiente e posicionamento adequado de medidas para que possa produzir resultados confiáveis

A Figura 2 ilustra o encadeamento das etapas do processo de EE descritas anteriormente. Nas seções seguintes são apresentados apenas os conceitos diretamente relacionados à AC, uma vez que eles constituem a base da metodologia desenvolvida nesta tese.

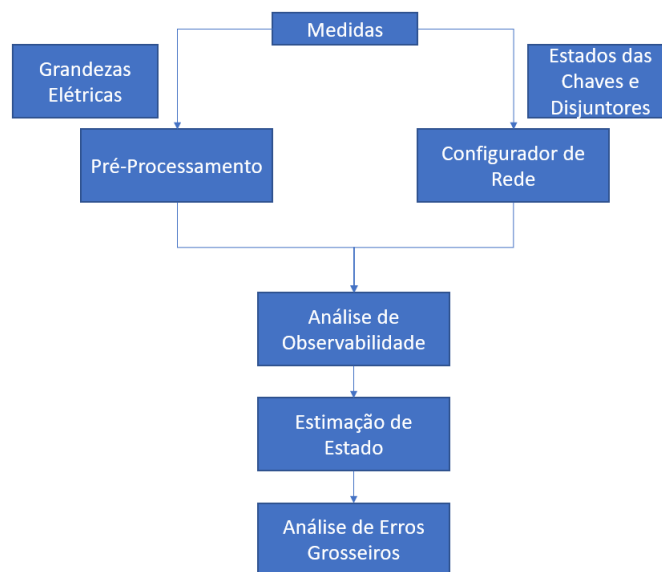


Figura 2: Etapas da EE.

2.1.2 Conceitos Básicos

Para uma determinada configuração da rede elétrica, o estado de operação e as medidas coletadas no sistema podem ser representados pelo modelo de medição:

$$\mathbf{z} = \mathbf{h}(\mathbf{x}) + \mathbf{e}, \quad (2.1)$$

em que $\mathbf{z}[m \times 1]$ representa o vetor de medidas coletadas, $\mathbf{h}(\mathbf{x})[m \times 1]$ representa o vetor de funções não lineares que relacionam as grandezas medidas ao vetor de estado $\mathbf{x}$, e $\mathbf{e}[m \times 1]$ representa o vetor de erros de medição. Assume-se que os erros de medição apresentam distribuição Gaussiana, com média zero. O parâmetro m corresponde ao número de medidas disponíveis no sistema.

No modelo de estimação de estado em regime permanente, considerando a formulação não linear em corrente alternada, o vetor de estado $\mathbf{x}$ é usualmente definido pelas magnitudes e pelos ângulos das tensões nas barras da rede. Considerando uma rede com N barras e uma barra de referência, o vetor de estado pode ser representado por

$$\mathbf{x} = \begin{bmatrix} \boldsymbol{\theta} \\ \mathbf{V} \end{bmatrix} = \begin{bmatrix} \theta_2 & \cdots & \theta_N & V_1 & \cdots & V_N \end{bmatrix}^T, \quad (2.2)$$

em que θ_i e V_i representam, respectivamente, o ângulo e a magnitude da tensão na barra i . O ângulo da barra de referência (θ_1) é considerado conhecido e, portanto, não faz parte do vetor de estado.

O vetor $\mathbf{h}(\mathbf{x})$ é constituído pelas funções que relacionam o estado da rede às grandezas efetivamente medidas. Dessa forma, dependendo do plano de medição considerado, $\mathbf{h}(\mathbf{x})$ pode ser organizado como

$$\mathbf{h}(\mathbf{x}) = \begin{bmatrix} \mathbf{P}_i(\mathbf{x}) \\ \mathbf{Q}_i(\mathbf{x}) \\ \mathbf{P}_{i,j}(\mathbf{x}) \\ \mathbf{Q}_{i,j}(\mathbf{x}) \\ \mathbf{V}_i^f(\mathbf{x}) \end{bmatrix}, \quad (2.3)$$

em que $\mathbf{P}_i$, $\mathbf{Q}_i$, $\mathbf{P}_{i,j}$ e $\mathbf{Q}_{i,j}$ representam, respectivamente, os vetores de funções associados às medidas de injeção de potência ativa e reativa e aos fluxos de potência ativa e reativa nas linhas de transmissão, enquanto $\mathbf{V}_i^f$ representa as funções associadas às medidas de magnitude de tensão.

Para a formulação em corrente alternada, considerando a matriz de admitâncias nodais $\mathbf{Y} = \mathbf{g} + j\mathbf{b}$, as funções de injeção de potência ativa e reativa na barra i são dadas,

respectivamente, por

$$P_i = V_i \sum_{j=1}^N V_j [g_{ij} \cos(\theta_i - \theta_j) + b_{ij} \sin(\theta_i - \theta_j)], \quad (2.4)$$

$$Q_i = V_i \sum_{j=1}^N V_j [g_{ij} \sin(\theta_i - \theta_j) - b_{ij} \cos(\theta_i - \theta_j)]. \quad (2.5)$$

De forma análoga, para uma linha que conecta as barras i e j , as funções associadas aos fluxos de potência ativa e reativa podem ser representadas por

$$P_{ij} = V_i^2 g_{ij} - V_i V_j [g_{ij} \cos(\theta_i - \theta_j) + b_{ij} \sin(\theta_i - \theta_j)], \quad (2.6)$$

$$Q_{ij} = -V_i^2 b_{ij} - V_i V_j [g_{ij} \sin(\theta_i - \theta_j) - b_{ij} \cos(\theta_i - \theta_j)]. \quad (2.7)$$

Por sua vez, para uma medida de magnitude de tensão na barra i , a função de medição é diretamente expressa por

$$V_i^f = V_i. \quad (2.8)$$

Assim, cada medida disponível no sistema corresponde a uma função específica do vetor de estado. Dessa forma, o vetor $\mathbf{h}(\mathbf{x})$ representa os valores teóricos das grandezas medidas para um determinado estado $\mathbf{x}$ da rede, enquanto o vetor $\mathbf{z}$ representa os respectivos valores obtidos por meio do sistema de medição. A diferença entre esses valores é atribuída aos erros de medição, representados pelo vetor $\mathbf{e}$ na Equação (2.1).

A partir do modelo de medição apresentado na Equação (2.1), o estado da rede elétrica pode ser determinado por meio do método dos Mínimos Quadrados Ponderados (MQP). Nesse método, busca-se determinar o vetor de estado $\mathbf{x}$ que minimiza a diferença entre as medidas obtidas no sistema e os respectivos valores calculados a partir do estado estimado. Essa diferença é ponderada de acordo com a incerteza associada a cada medida, representada pela matriz $\mathbf{R}$.

Assim, utilizando o modelo de medição da Equação (2.1), o problema de estimação de estado pode ser formulado como o seguinte problema de otimização, em que $J(\mathbf{x})$ representa a função objetivo do problema:

$$\min_{\mathbf{x}} J(\mathbf{x}) = [\mathbf{z} - \mathbf{h}(\mathbf{x})]^T \mathbf{R}^{-1} [\mathbf{z} - \mathbf{h}(\mathbf{x})], \quad (2.9)$$

Considerando que os erros de medição são estatisticamente independentes, a matriz $\mathbf{R}$ pode ser representada por uma matriz diagonal:

$$\mathbf{R} = \text{diag}(\sigma_1^2, \sigma_2^2, \dots, \sigma_m^2), \quad (2.10)$$

em que σ_i^2 representa a variância associada à i -ésima medida. Nesse caso, a função objetivo pode ser expressa como

$$J(\mathbf{x}) = \sum_{i=1}^m \frac{[z_i - h_i(\mathbf{x})]^2}{\sigma_i^2}. \quad (2.11)$$

Essa forma evidencia que medidas com menor variância, isto é, maior confiabilidade, possuem maior peso no processo de estimação.

Para determinar o estado estimado, a função objetivo $J(\mathbf{x})$ deve ser minimizada em relação ao vetor de estado. Como $\mathbf{h}(\mathbf{x})$ é, em geral, uma função não linear para o modelo de corrente alternada, a solução do problema de otimização não pode ser obtida diretamente. A condição necessária para a obtenção do mínimo é dada pelo anulamento do gradiente da função objetivo:

$$\nabla J(\mathbf{x}) = 0. \quad (2.12)$$

Para isso, define-se a matriz Jacobiana $\mathbf{H}(\mathbf{x})$ como

$$\mathbf{H}(\mathbf{x}) = \frac{\partial \mathbf{h}(\mathbf{x})}{\partial \mathbf{x}}, \quad (2.13)$$

em que cada elemento H_{ij} representa a derivada parcial da i -ésima função de medição em relação à j -ésima variável de estado:

$$H_{ij} = \frac{\partial h_i(\mathbf{x})}{\partial x_j}. \quad (2.14)$$

A partir da derivação da função objetivo em relação ao vetor de estado, obtêm-se as equações normais do problema de estimação:

$$\mathbf{H}^T(\mathbf{x})\mathbf{R}^{-1}[\mathbf{z} - \mathbf{h}(\mathbf{x})] = 0. \quad (2.15)$$

A Equação (2.15) corresponde a um sistema de equações não lineares, uma vez que tanto $\mathbf{h}(\mathbf{x})$ quanto $\mathbf{H}(\mathbf{x})$ dependem do vetor de estado. Consequentemente, sua solução requer um processo iterativo. Sendo assim, uma das abordagens mais utilizadas para resolver esse sistema é o método de *Newton-Raphson*. Nesse caso, considerando uma

estimação do estado na iteração k , $\mathbf{x}^{(k)}$, o vetor de estado é atualizado de acordo com

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \Delta\mathbf{x}^{(k)}, \quad (2.16)$$

em que $\Delta\mathbf{x}^{(k)}$ representa o incremento aplicado ao vetor de estado.

A cada iteração, a matriz Jacobiana é avaliada no estado atual $\mathbf{x}^{(k)}$ e o modelo de medição é linearizado em torno desse ponto. A partir da aplicação do método de Newton ao problema de estimação, obtém-se o seguinte sistema linear para o cálculo do incremento do vetor de estado:

$$[\mathbf{H}^T(\mathbf{x}^{(k)})\mathbf{R}^{-1}\mathbf{H}(\mathbf{x}^{(k)})] \Delta\mathbf{x}^{(k)} = \mathbf{H}^T(\mathbf{x}^{(k)})\mathbf{R}^{-1} [\mathbf{z} - \mathbf{h}(\mathbf{x}^{(k)})]. \quad (2.17)$$

A matriz que multiplica o incremento $\Delta\mathbf{x}^{(k)}$ no lado esquerdo da Equação (2.17) é denominada Matriz de Ganho ($\mathbf{G}$), sendo definida por

$$\mathbf{G}(\mathbf{x}) = \mathbf{H}^T(\mathbf{x})\mathbf{R}^{-1}\mathbf{H}(\mathbf{x}). \quad (2.18)$$

Assim, a Equação (2.17) pode ser escrita de forma compacta como

$$\mathbf{G}(\mathbf{x}^{(k)})\Delta\mathbf{x}^{(k)} = \mathbf{H}^T(\mathbf{x}^{(k)})\mathbf{R}^{-1} [\mathbf{z} - \mathbf{h}(\mathbf{x}^{(k)})]. \quad (2.19)$$

A solução desse sistema fornece o incremento $\Delta\mathbf{x}^{(k)}$, utilizado a Equação (2.16) para atualizar a estimação do vetor de estado de acordo com

Esse processo é repetido iterativamente, com a avaliação da matriz Jacobiana e da função de medição a cada nova estimação do estado, até que um critério de convergência seja satisfeito. Um critério usual consiste em verificar se a maior componente, em valor absoluto, do incremento do vetor de estado é inferior a uma tolerância previamente estabelecida:

$$|\Delta\mathbf{x}^{(k)}|_{\infty} < \varepsilon. \quad (2.20)$$

Ao final do processo iterativo, o vetor $\mathbf{x}$ obtido representa a estimação do estado de operação da rede elétrica que melhor se ajusta, segundo o critério de mínimos quadrados ponderados, ao conjunto de medidas disponíveis.

2.1.3 Análise de Observabilidade

Uma rede elétrica é considerada observável se existir um número suficiente de medidas, diversificadas e bem distribuídas pela rede para que a filtragem do estado seja possível em toda a rede. Matematicamente, a análise de observabilidade destina-se à verificação da possibilidade de solução do sistema de equações estabelecido em (2.9). Alternativamente, o sistema é considerado topologicamente observável se for possível determinar unicamente todos os fluxos de potência na rede partindo-se das medidas disponibilizadas (Monticelli, 2012).

Este trabalho adota a análise de observabilidade por meio de um modelo linear, com desacoplamento $P\theta$ (potência ativa-ângulo de fase). Supõe-se que as medidas de fluxo e injeção de potência, ativa e reativa, sejam tomadas aos pares, como também, que ao menos uma medida de ângulo do fasor de tensão esteja presente. Sendo assim, o seguinte modelo é utilizado:

$$\mathbf{z}_a = \mathbf{H}_a \boldsymbol{\theta} + \mathbf{e}_a, \quad (2.21)$$

em que $\mathbf{H}_a[m_a \times n]$ representa a matriz Jacobiana das medidas ativas, $\boldsymbol{\theta}[n \times 1]$ o vetor de ângulos das tensões das barras, $\mathbf{e}_a[m_a \times 1]$ o vetor de medidas ativas, m_a o total de medidas ativas.

Em um sistema numericamente observável a matriz $\mathbf{H}_a$ deve apresentar posto completo, com pelo menos uma medida de ângulo presente. A determinação da observabilidade depende apenas da topologia da rede e da configuração das medidas do sistema (número, tipo e localização das medidas) (Gou; Abur, 2000). Portanto, costuma-se assumir que todas as medidas do esquema possuam igual peso de participação no processo de estimação ($\mathbf{R}_a = \mathbf{I}$, matriz identidade) e que os ramos da rede estejam representados somente por susceptâncias-série unitárias.

A partir destas considerações, os elementos de $\mathbf{H}_a$, associados a uma dada medida l , passam a ser definidos da seguinte maneira (Gou; Abur, 2000):

- Fluxo de Potência Ativa no ramo i-k:

$$\mathbf{H}_a(l, i) = 1; \quad \mathbf{H}_a(l, k) = -1 \quad (2.22)$$

- Injeção de Potência Ativa na barra i :

$$\mathbf{H}_a(l, i) = nb_i; \quad \mathbf{H}_a(l, k) = -1 \quad (2.23)$$

onde nb_i corresponde ao número de barras conectadas à barra i .

Quando unidades de medição fasorial (UMFs) estão presentes no sistema de medição, medidas de ângulo de tensão e de corrente podem ser incorporadas ao modelo linear. Nesse caso, considerando a aproximação $P\theta$, essas medidas podem ser representadas na matriz $\mathbf{H}_a$ conforme descrito a seguir (Do Coutto Filho; Stacchini de Souza; Augusto, 2014):

- A medida de ângulo da tensão na barra i :

$$\theta_i \rightarrow \mathbf{H}_a(l, i) = 1 \quad (2.24)$$

- A parte real da corrente entre as barras i e k :

$$I_{ik} \rightarrow \mathbf{H}_a(l, i) = 1; \quad \mathbf{H}_a(l, k) = -1 \quad (2.25)$$

A verificação da observabilidade pode ser realizada por meio da fatoração da matriz de ganho ($\mathbf{G}_a$), construída com base no modelo linear, de forma semelhante à apresentada anteriormente. O algoritmo descrito em (Augusto, 2016) e (Gou; Abur, 2000) apresenta detalhadamente esta verificação.

A partir do modelo linear definido em (2.21), a matriz de ganho pode ser definida por

$$\mathbf{G}_a = \mathbf{H}_a^\top \mathbf{R}_a^{-1} \mathbf{H}_a. \quad (2.26)$$

Como, para a análise de observabilidade topológica, assume-se que todas as medidas possuem o mesmo peso, adota-se $\mathbf{R}_a = \mathbf{I}$. Consequentemente,

$$\mathbf{G}_a = \mathbf{H}_a^\top \mathbf{H}_a. \quad (2.27)$$

Uma rede é considerada observável quando a matriz $\mathbf{H}_a$ possui posto de coluna completo. De forma equivalente, a matriz de ganho $\mathbf{G}_a$ possui posto completo e é não singular. Assim, durante a fatoração de $\mathbf{G}_a$, a ausência de pivôs nulos indica que o sistema é observável.

2.2 Análise de Criticalidades via Matriz de Covariância de Resíduos

Durante a operação da rede, podem ocorrer indisponibilidades de medidas, levando a redundância do SM a níveis críticos, que representam risco de perda iminente de observabilidade e comprometimento da confiabilidade da EE.

A AC tem como objetivo revelar as vulnerabilidades do SM, identificando elementos, tomados individualmente ou em grupos que são essenciais para que o sistema seja observado com um todo. Sendo assim, adotam-se as seguintes definições:

- Medida Crítica (C_{1s}): quando ocorre a indisponibilidade destas, o sistema torna-se inobservável.
- k -Tupla Crítica (C_k): grupo de k medidas cuja indisponibilidade simultânea leva o sistema à inobservabilidade.

A AC pode ser realizada a partir das propriedades dos resíduos obtidos no processo de estimação de estado. Embora os resíduos permitam detectar a presença de erros grosseiros, sua análise individual não é suficiente para identificar todas as situações de criticalidade. Em particular, medidas pertencentes a uma mesma tupla crítica podem apresentar comportamento indistinguível, sendo necessário considerar também a relação existente entre seus resíduos.

Essa relação pode ser obtida a partir da matriz Ω_a , associada aos resíduos do modelo linear utilizado na análise de observabilidade (Braga Flor, Vinicius B. *et al.*, 2023). Considerando que todas as medidas possuem o mesmo peso, $\mathbf{R}_a = \mathbf{I}$, a matriz é definida por

$$\Omega_a = \mathbf{I} - \mathbf{H}_a \mathbf{G}_a^{-1} \mathbf{H}_a^\top, \quad (2.28)$$

Os elementos diagonais de $\Omega_a[m_a \times m_a]$ estão associados às variâncias dos resíduos, enquanto os elementos fora da diagonal representam a relação entre resíduos de diferentes medidas.

A partir desses elementos, define-se o coeficiente de correlação residual entre as medidas i e j como

$$\gamma_{ij} = \frac{\Omega_a(i,j)}{\sqrt{\Omega_a(i,i)\Omega_a(j,j)}}, \quad (2.29)$$

em que $\Omega_a(i,j)$ representa a relação entre os resíduos das medidas i e j , enquanto $\Omega_a(i,i)$ e $\Omega_a(j,j)$ correspondem às respectivas variâncias. O coeficiente γ_{ij} varia entre -1 e 1 , sendo que valores próximos de ± 1 indicam forte correlação e valores próximos de zero indicam ausência de correlação linear.

No contexto da AC, o coeficiente de correlação deve ser analisado conjuntamente com os resíduos normalizados. Para medidas pertencentes a um mesmo conjunto crítico, os resíduos normalizados podem apresentar valores numericamente iguais, mesmo quando não são nulos. Assim, para duas medidas i e j , a condição

$$r_a[i] = r_a[j] \quad \text{e} \quad \gamma_{ij} = 0 \quad (2.30)$$

caracteriza o comportamento associado a medidas que pertencem a uma mesma C_k . Nessa situação, um eventual erro grosseiro introduzido em uma das medidas não pode ser identificado de forma inequívoca pelo teste residual, uma vez que as medidas apresentam resíduos indistinguíveis.

Para uma medida individualmente crítica, C_1 , essa propriedade assume uma forma particular. Nesse caso, a indisponibilidade da medida torna o sistema inobservável e seu resíduo normalizado é identicamente nulo, resultando em

$$r_a[i] = 0 \quad \text{e} \quad \Omega_a(i,i) = 0. \quad (2.31)$$

Dessa forma, a análise conjunta dos resíduos normalizados e da matriz Ω_a permite identificar medidas individualmente críticas e estabelecer relações entre medidas que podem pertencer a criticalidades. Essas propriedades constituem a base para a identificação de criticalidades por meio da análise dos resíduos da estimação de estado.

2.2.1 Propriedades para a Identificação de Criticalidades

As propriedades apresentadas nas seções anteriores estabelecem as relações entre observabilidade, resíduos e criticalidades e constituem a base teórica para a identificação de C_k por meio da matriz Ω_a . A partir dessas propriedades, podem ser estabelecidos os seguintes resultados ([Augusto, 2016](#)):

1. Dependência linear das colunas de Ω_a : as colunas de Ω_a associadas às medidas pertencentes a uma mesma C_k são linearmente dependentes.

A matriz Ω_a relaciona os resíduos das diferentes medidas do sistema. Seus elementos diagonais estão associados à parcela do resíduo de uma determinada medida, enquanto os elementos fora da diagonal representam a relação entre os resíduos de diferentes medidas. Dessa forma, as colunas de Ω_a representam como o resíduo associado a cada medida se relaciona com os resíduos de todo o conjunto de medidas.

Quando um conjunto de medidas pertence a uma mesma C_k , essas medidas apresentam uma relação de dependência associada à observabilidade da rede. Consequentemente, as informações relacionadas aos seus resíduos também não são independentes, fazendo com que as colunas de Ω_a correspondentes a essas medidas sejam linearmente dependentes. Assim, uma ou mais dessas colunas podem ser expressas como uma combinação linear das demais.

Para um conjunto candidato de k medidas, essa propriedade pode ser verificada pela singularidade da submatriz $\tilde{\Omega}_a[k \times k]$, formada pelas linhas e colunas de Ω_a correspondentes às medidas analisadas.

2. Minimalidade das C_k : uma C_k não pode conter uma C_j , com $j < k$.

Por definição, uma C_k é um conjunto minimal de k medidas cuja indisponibilidade simultânea torna o sistema inobservável. Portanto, se um subconjunto de $j < k$ medidas de uma dada C_k também fosse crítico, a indisponibilidade desse subconjunto já seria suficiente para provocar a inobservabilidade, contradizendo a definição de minimalidade de C_k .

As duas propriedades apresentadas fornecem os critérios utilizados neste trabalho para a identificação de criticalidades. A Propriedade 1 permite identificar conjuntos candidatos a partir da dependência linear das colunas da matriz Ω_a . Para um conjunto candidato de k medidas, a dependência pode ser verificada por meio da singularidade da submatriz $\tilde{\Omega}_a[k \times k]$ correspondente.

Entretanto, essa condição, isoladamente, não garante que o conjunto candidato seja uma C_k , pois, ele pode conter uma criticalidade de menor cardinalidade. A Propriedade 2 estabelece, portanto, a condição de minimalidade necessária para classificar o conjunto. Assim, um conjunto de k medidas será considerado uma C_k somente quando apresentar a dependência indicada pela Propriedade 1 e nenhum de seus subconjuntos próprios apresentar uma criticalidade.

A partir deste ponto, por simplicidade de notação, será utilizada a matriz Ω para representar a matriz de covariância dos resíduos, sem distinção entre Ω_a e Ω , salvo quando essa distinção for necessária.

Com base nas propriedades apresentadas, pode-se estabelecer um procedimento sistemático para a identificação das C_k presentes em um sistema de medição. O procedimento explora a dependência linear das submatrizes de Ω , a minimalidade dos conjuntos críticos e os limites de cardinalidade estabelecidos anteriormente. A metodologia desenvolvida neste trabalho é organizada em quatro etapas, descritas a seguir.

2.2.2 Etapas da Análise de Criticalidades

O processo da AC adotado nesta tese baseia-se em quatro etapas principais (Nishio; Clua *et al.*, 2020), que são realizadas em cada iteração até atingir o valor k_{max} preestabelecido. O Algoritmo 1 apresenta a implementação mais recente e eficiente das etapas da AC descritas no artigo de Nishio, Do Coutto Filho, Stacchini De Souza *et al.* (2024). A entrada do algoritmo consiste na matriz de covariância das medidas Ω , no número total de medidas m e na cardinalidade máxima k_{max} . A saída do algoritmo é o conjunto de criticalidades encontradas (*SolSet*). Na etapa 1 (Enumeração) todas as combinações pretendidas de k medidas são listadas. Na etapa 2 (Poda), como uma C_k não pode incluir uma C_j de cardinalidade inferior ($j < k$), o algoritmo exclui todas as que não atendem a esta propriedade da solução final. Na etapa 3 (Avaliação) são demarcadas todas as combinações de k medidas que tornam o sistema inobservável, quando indisponíveis simultaneamente. Por fim, na etapa 4 (Atualização), as combinações confirmadas na etapa anterior são adicionadas ao *SolSet*.

Algoritmo 1: Entrada, Saída e Etapas envolvidas na AC.

Entrada: Ω, m, k_{max}

Saída : *SolSet*

```

1 para  $k \leftarrow 1$  até  $k_{max}$  faça
2   Etapa 1: Enumeração
3   Etapa 2: Poda
4   Etapa 3: Avaliação
5   Etapa 4: Atualização

```

Esta versão do algoritmo aprimora a proposta original de Nishio, Clua *et al.* (2020) ao realizar a etapa de poda antes da etapa de avaliação. Isto é vantajoso, pois a avaliação envolve a análise da singularidade de matrizes e apresenta maior custo computacional, a

redução prévia do número de combinações a serem analisadas diminui significativamente o esforço necessário nessa fase. Consequentemente, o tempo total de execução do algoritmo é reduzido.

2.2.3 Criticalidades de Unidades de Medição

Embora as criticalidades possam envolver medições espalhadas pela rede — isto é, fornecidas por Unidades de Medição (UMs) instaladas em diferentes barras (nós da rede) — o evento dominante de indisponibilidade é a inacessibilidade simultânea de todas as medições pertencentes à mesma barra da rede. Notavelmente, a falta de medidas ocorre em regiões da rede centradas nas criticalidades de UMs (C_{k-UMs}) (Do Coutto Filho; Stacchini de Souza; Tafur, 2013). Sendo assim, diversos artigos utilizam uma análise orientada à nós para tratar as C_{k-UMs} . Como é o caso de Vinicius B. Braga Flor *et al.* (2023) que utilizam as C_{k-UMs} para quantificar a vulnerabilidade do sistema de medição e propor medidas preventivas, considerando a probabilidade de ocorrência de ataques cibernéticos. Alternativamente, este trabalho utiliza deste conceito para viabilizar a comparação dos resultados da implementação da análise da GPU com os resultados obtidos pelo algoritmo Branch & Bound apresentado por Augusto, Do Coutto Filho *et al.* (2019).

Por definição, um grupo de UMs de cardinalidade k torna-se crítico — denominado C_{k-UMs} — somente quando a indisponibilidade simultânea de todas essas UMs torna o sistema inobservável; como uma tupla mínima de cardinalidade k , a indisponibilidade de j UMs desse grupo (para $j < k$) não leva à perda de observabilidade. Em outras palavras, uma $C_{(k-1)-UM}$ não é um subconjunto de uma C_{k-UM} , e o mesmo vale para $C_{(k-2)-UM}$, $\dots$, C_{1-UM} .

A principal diferença da abordagem orientada a UMs na identificação de criticalidades de medidas reside na construção das submatrizes de covariância empregadas na Etapa 3 (Avaliação) do algoritmo. A implementação da AC orientada a UMs considera todas as medidas das UMs envolvidas. Ou seja, a submatriz de covariância $\mathbf{\Omega}^{(k)}$ associada a uma combinação de k UMs é obtida ao extrair de $\mathbf{\Omega}$ todas as linhas e colunas correspondentes às medidas das UMs dessa combinação.

A Figura 3 ilustra a construção da submatriz de covariância, matriz auxiliar $\mathbf{\Omega}_{(AB)}^{(2)}$, que representa o par (A, B) de UMs. Na segunda iteração, isto é, na busca por criticalidades de cardinalidade $k = 2$, a UM_A fornece três medidas — a_1 , a_2 e a_3 — e a UM_B fornece duas medidas — b_1 e b_2 . As linhas e colunas de $\mathbf{\Omega}_{(AB)}^{(2)}$ são organizadas conforme mostrado na Fig. 3. Se a submatriz $\mathbf{\Omega}_{(AB)}^{(2)}$ for singular, então o par (A, B) de UMs é declarado uma

C_{2-UM} .

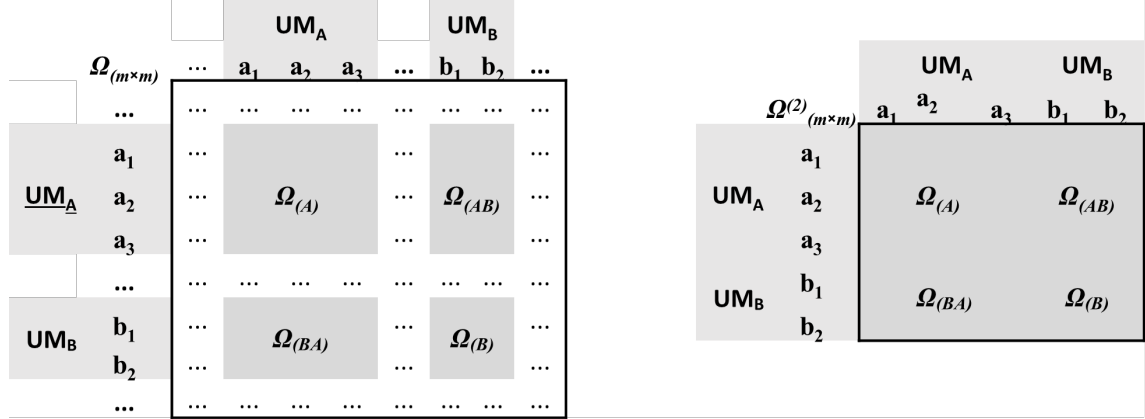


Figura 3: Linhas e colunas da matriz $\Omega_{(AB)}^{(2)}$ associadas a uma combinação de UMs.

Para facilitar a leitura, a Figura 4 ilustra a execução dos três passos da AC para identificar, por exemplo, triplas críticas de 5 UMs. Cada UM é representada por uma letra da sequência A, B, C, D e E. O primeiro passo enumera todas as $\binom{5}{3} = 10$ combinações possíveis de UMs, numeradas consecutivamente, a saber: (A, B, C), (A, B, D), (A, B, E), (A, C, D), (A, C, E), (A, D, E), (B, C, D), (B, C, E), (B, D, E) e (C, D, E). Considerando um ambiente de programação paralela, essas combinações são processadas simultaneamente por meio de *threads* dedicadas a cada uma delas.

Assumindo que (B, C) já tenha sido identificada como uma C_{2-UM} no Passo 3 da iteração anterior ($k = 2$) e que esse resultado foi devidamente armazenado. O Passo 2 utiliza essa informação para determinar que as combinações (A, B, C) , (B, C, D) e (B, C, E) contêm o par (B, C) . Esses triplas são candidatas não promissoras, sendo descartadas (podadas) e, portanto, não participam do Passo 3.

O algoritmo constrói, no Passo 3, as submatrizes de covariância extraídas da matriz Ω e as triangulariza para verificar a condição de observabilidade. Se, por exemplo, as triplas (A, B, E) e (A, C, D) estiverem associadas a submatrizes singulares $\Omega_{(3)}^{(3)}$ e $\Omega_{(4)}^{(3)}$, então são classificados como C_{3-UM} .

2.2.4 Complexidade da Análise de Criticalidades

A AC é um problema de natureza combinatorial, no qual, o número de combinações de medidas (m_{comb}) que precisam ser avaliadas, em um processo enumerativo exaustivo é descrito pela equação 2.32, em que m representa o número de medidas presente no SM e k_{max} representa a cardinalidade máxima a ser analisar.

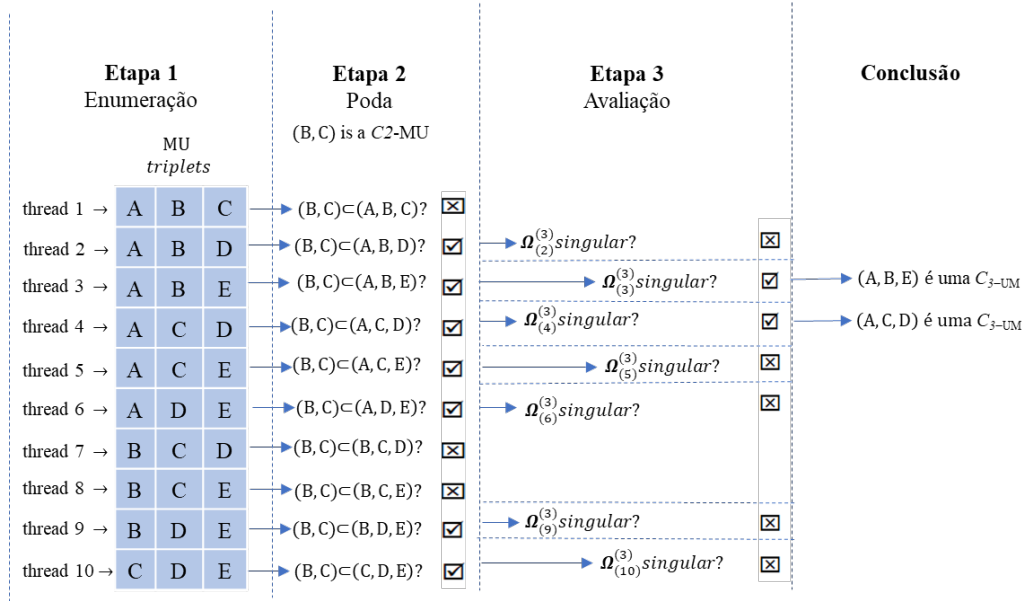


Figura 4: Exemplo Ilustrativo da AC de UMs.

$$m_{\text{comb}} = \sum_{k=1}^{k_{\text{max}}} \frac{m!}{(k!(m-k)!)} \quad (2.32)$$

Como pode ser visto, ocorre um crescimento de ordem fatorial do número de combinações em relação ao tamanho do SM. Além disso, a complexidade do problema é influenciada diretamente apenas por m e k . O tamanho do sistema (n^0 de barras) está relacionado indiretamente à complexidade do problema, pois caracteriza a dimensão do SM para alcançar a observabilidade e a distribuição de criticidades do sistema.

2.3 Programação paralela em GPUs

A computação paralela permite reduzir o tempo de execução de cálculos por meio da realização simultânea de múltiplas operações, sendo particularmente vantajosa em problemas que apresentam elevado grau de paralelismo. Nesse contexto, destacam-se as Unidades Centrais de Processamento (CPUs) e as Unidades de Processamento Gráfico (GPUs). As CPUs são constituídas por um número relativamente reduzido de núcleos de propósito geral, enquanto as GPUs possuem uma arquitetura altamente paralela, composta por um grande número de unidades de processamento capazes de executar simultaneamente um elevado número de threads. Assim, as GPUs são particularmente adequadas a aplicações nas quais um grande conjunto de operações pode ser executado de forma independente ou com baixo grau de dependência entre si.

As GPUs foram originalmente desenvolvidas para aplicações de processamento gráfico. Entretanto, com o surgimento da arquitetura *Compute Unified Device Architecture* (CUDA), uma plataforma de computação paralela e interface de programação de aplicações (*Application Programming Interface* – API) desenvolvida pela *NVIDIA*, tornou-se possível utilizar GPUs para acelerar aplicações científicas e de engenharia. A programação em CUDA possui características específicas que permitem distribuir e coordenar a execução de tarefas paralelas, denominadas *threads*, entre os diversos núcleos de processamento da GPU.

2.3.1 Paradigmas da programação em GPU

A avaliação simultânea da vasta quantidade de combinações possíveis é a principal motivação para a implementação paralela da CA. Como esta análise envolve a execução de muitas operações simples, a implementação em GPU é adequada para acelerar a identificação de criticalidades. Ao abordar um determinado problema com uma implementação algorítmica, é fundamental considerar os paradigmas de programação para GPUs.

Primeiramente, algoritmos para programação paralela são heterogêneos, isto é, parte do código é destinada à GPU, responsável pela execução de funções com elevado paralelismo, enquanto outra parte é atribuída à CPU, que executa operações mais pesadas com baixo paralelismo. A CPU também é responsável por coordenar o fluxo da execução do código e a transferência de dados entre as arquiteturas. Uma implementação paralela adequada busca explorar o melhor das arquiteturas de CPU e GPU, identificando as etapas que apresentam melhor desempenho em cada uma delas. Além disso, a transferência de dados entre arquiteturas distintas é um gargalo de desempenho relevante em implementações que envolvem múltiplas arquiteturas. Uma implementação eficiente minimiza o número de transferências de dados entre a CPU e a GPU durante a execução do código.

A Figura 5 apresenta um código paralelo típico que utiliza programação em GPU. A CPU transfere as variáveis utilizadas durante a execução do código na GPU. Por meio de uma função específica, denominada *Kernel*, a CPU inicia a execução das etapas altamente paralelizáveis na GPU. Em seguida, a GPU invoca um grande número de *threads* que executam, em paralelo, uma tarefa simples. Finalmente, a CPU recupera os resultados da GPU para análise posterior.

A divisão e organização do trabalho entre as *threads* são definidas pelo programador. Cada *thread* possui um identificador (*Id*) associado à porção dos dados a ser processada. Esse identificador é determinado a partir de uma hierarquia de dois níveis: *threadId*,

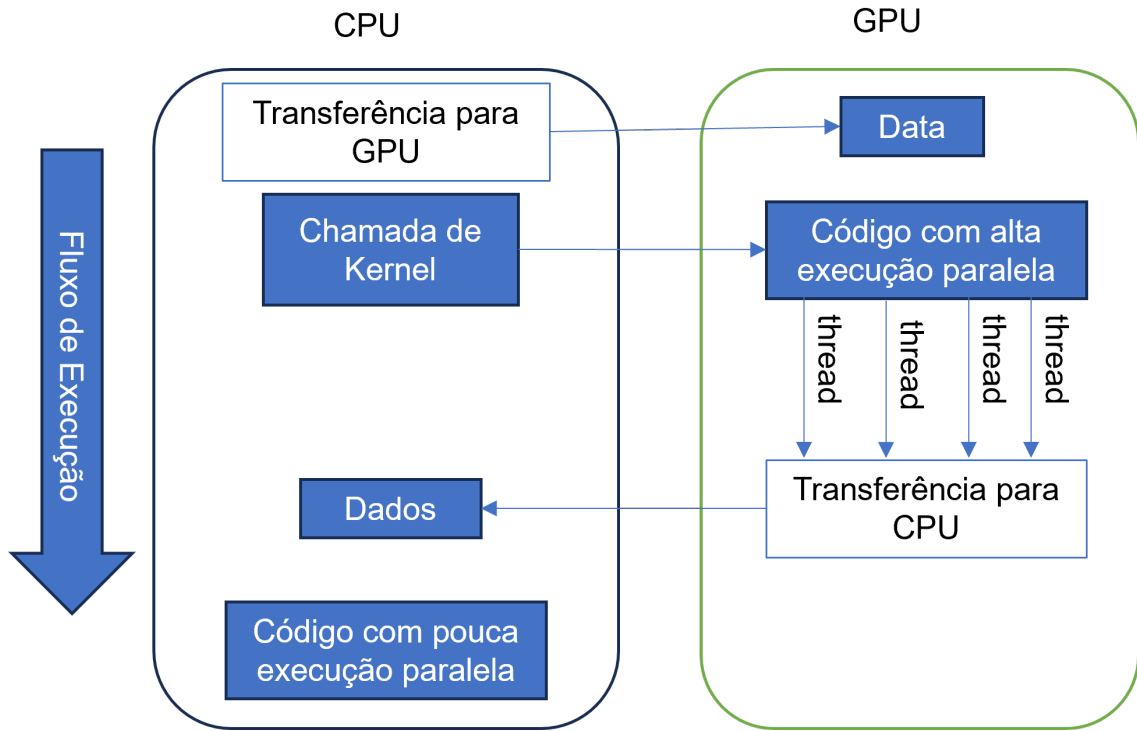


Figura 5: Representação do fluxo da execução e dados para a aplicação da AC em GPUs.

que identifica a *thread* dentro de um bloco, e *blockId*, que identifica o bloco dentro da grade. Dessa forma, o identificador global de uma *thread* pode ser obtido por meio da Equação (2.33).

$$Id = threadId + blockDim \times blockId \quad (2.33)$$

2.4 Implementação da Análise de Criticalidades na GPU

A partir dos conceitos apresentados, a implementação da AC na arquitetura da GPU é favorável. Pois, a etapa mais onerosa, a de avaliação, consiste na inversão de matrizes de pequena dimensão para cada possível combinação de medidas (final da Seção 2.2.1). Esta etapa é bem adaptável para a GPU, pois, é essencialmente um número massivo de operações simples que podem ser realizadas simultaneamente. Além disso, todas as quatro etapas possuem uma versão paralela, o que possibilita implementá-las na GPU. Consequentemente, a transferência de dados entre arquiteturas distintas é minimizada, pois apenas duas transferências de dados são realizadas durante a execução: a matriz Ω que é inicialmente transferida da CPU para a GPU; e no final da implementação, a lista de criticalidades encontradas é retornada para a CPU.

De forma mais detalhada, o Algoritmo 2 apresenta o código principal da implementação da CA em GPU. O algoritmo recebe como entrada a matriz de covariância dos resíduos Ω , a cardinalidade máxima de interesse $k_{\max}$ e o número total de medidas m do sistema de medição. Como saída, produz o conjunto solução (*SolSet*), contendo todos as combinações críticas identificadas.

Algoritmo 2: *Kernel* Principal da Análise de Criticalidade na GPU

```

1.   $tpb \leftarrow 1024$ 
2.   $n_{\max comb} \leftarrow \binom{m}{k_{\max}}$ 
3.   $\text{cudaMalloc}(\text{Combs}, n_{\max}^{comb} \cdot k_{\max} \cdot \text{sizeof}(\text{int}))$ 
4.   $\text{cudaMalloc}(\text{isCrit}, n_{\max}^{comb} \cdot \text{sizeof}(\text{int}))$ 
5.   $\text{cudaMalloc}(\text{pointer}, n_{\max}^{comb} \cdot \text{sizeof}(\text{int}))$ 
6.   $\text{cudaMalloc}(\Omega, m^2 \cdot \text{sizeof}(\text{double}))$ 
7.   $\text{cudaMemcpy}(\Omega, \text{cudaMemcpyHostToDevice})$ 
8.  for  $k = 1$  to  $k_{\max}$ 
9.       $n_{comb} \leftarrow \binom{m}{k}$ 
10.      $gridSize \leftarrow \lceil n_{comb}/tpb \rceil$ 
11.      $\text{Combs} = \text{Etapa1} \ll \text{gridSize}, tpb \gg (m, k)$ 
12.      $\text{isCrit} = \text{Etapa2} \ll \text{gridSize}, tpb \gg (k, \text{Combs}, \text{isCrit})$ 
13.      $\text{isCrit} = \text{Etapa3} \ll \text{gridSize}, tpb \gg (k, \text{Combs}, \Omega)$ 
14.      $\text{pointer} = \text{Scan} \ll \text{gridSize}, tpb \gg (k, \text{isCrit})$ 
15.      $\text{SolSet} = \text{Compact} \ll \text{gridSize}, tpb \gg (k, \text{pointer}, \text{SolSet})$ 
16. end for
17.  $\text{cudaMemcpy}(\text{SolSet}, \text{cudaMemcpyDeviceToHost})$ 

```

Inicialmente, na linha 1, é definido o número de *threads* por bloco (*threads per block* – *tpb*), parâmetro utilizado para configurar a execução dos *kernels* na GPU. Em seguida, na linha 2, é calculado o número máximo de combinações que poderá ser analisado durante a execução, $n_{\max comb} = \binom{m}{k_{\max}}$, o qual é utilizado para dimensionar as estruturas de dados alocadas na memória global da GPU.

As linhas de 3 a 6 realizam a alocação de memória das principais estruturas utilizadas pelo algoritmo na GPU. A matriz *Combs* armazena as combinações de medidas geradas na Etapa 1. O vetor binário *isCrit* é utilizado para indicar se uma determinada combinação é candidata a criticalidade. O vetor *pointer* armazena os índices gerados pela operação *Scan*, enquanto a matriz Ω contém a matriz de covariância dos resíduos utilizada na avaliação das combinações.

Na linha 7, a matriz Ω é transferida da memória principal da CPU para a memória global da GPU por meio da função *cudaMemcpy*. Após essa transferência, todas as etapas subsequentes são executadas diretamente na GPU.

O laço principal, compreendido entre as linhas 8 e 16, é executado para cada cardinalidade k , variando de 1 até $k_{\max}$. Para cada valor de k , é inicialmente calculado o número de combinações a serem avaliadas, $n_{comb} = \binom{m}{k}$, conforme indicado na linha 9. Em seguida, na linha 10, é determinado o número de blocos de *threads* necessários para processar essas combinações em paralelo, $gridSize = \left\lceil \frac{n_{comb}}{tpb} \right\rceil$. A linha 11 executa o *kernel* da Etapa 1, responsável por gerar todas as combinações de medidas de cardinalidade k e armazená-las na matriz *Combs*. Na linha 12, é executado o *kernel* da Etapa 2, que realiza a poda das combinações contendo criticalidades já identificadas em iterações anteriores. Em seguida, a linha 13 executa o *kernel* da Etapa 3, responsável por avaliar as combinações remanescentes e identificar os candidatos a criticalidades por meio da análise de singularidade da matriz de covariância correspondente.

As linhas 14 e 15 implementam a Etapa 4. Inicialmente, a primitiva paralela *Scan* é utilizada para gerar um vetor de ponteiros contendo as posições de armazenamento das criticalidades identificadas. Posteriormente, a primitiva *Compact* utiliza essas posições para copiar apenas os elementos válidos para o conjunto solução *SolSet*, produzindo uma estrutura de dados compacta contendo exclusivamente as criticalidades identificadas.

Por fim, na linha 17, o conjunto solução *SolSet* é transferido da memória da GPU para a CPU, tornando os resultados disponíveis para processamento e análise posteriores.

As principais variáveis utilizadas pelo algoritmo são resumidas a seguir:

- m : número total de medidas do sistema;
- k : cardinalidade atualmente analisada;
- $k_{\max}$: cardinalidade máxima considerada na análise;
- n_{comb} : número de combinações de cardinalidade k ;
- $n_{\max comb}$: número máximo de combinações consideradas durante a execução;
- tpb : número de *threads* por bloco;
- $gridSize$: número de blocos lançados na GPU;
- Ω : matriz de covariância dos resíduos;
- *Combs*: matriz que armazena as combinações de medidas geradas;
- *isCrit*: vetor binário que indica as combinações candidatas a criticalidade;

2.4.1.1 Adaptação da Etapa 2: Poda

Primeiramente, este capítulo explica em detalhes a implementação dos *kernels* das etapas 2 e 3. A etapa 2 de Poda é executada pelo Algoritmo 3, que compara todas as criticalidades previamente armazenadas no conjunto solução (*SolSet*) com as combinações presentes em *Combs*. Caso uma determinada combinação contenha alguma criticalidade já identificada, o valor correspondente em *isCrit* é atualizado para zero, removendo-a da lista de candidatos.

Algoritmo 3: Kernel da Etapa 2: Poda

```

Input:  $k$ ,  $Combs$ ,  $isCrit$ 
Output:  $isCrit$ 
1.  $Tid \leftarrow threadIdx.x + blockDim.x \times blockIdx.x$ 
2. if  $Tid < \binom{m}{k}$ 
3.     if  $isCrit[Tid] = 1$ 
4.         for each  $j \in SolSet$ 
5.             if  $j \subset Combs[Tid]$ 
6.                  $isCrit[Tid] \leftarrow 0$ 
7.             end if
8.         end for
9.     end if
10. end if
  
```

Na implementação paralela, múltiplas *threads* executam simultaneamente o laço externo do algoritmo, permitindo que as verificações sejam realizadas de forma concorrente e explorando o paralelismo massivo oferecido pela GPU.

Para cada combinação válida, a linha 3 verifica o valor correspondente no vetor *isCrit*. Apenas as combinações marcadas com valor igual a 1 são processadas, uma vez que representam candidatos que ainda não foram descartados em etapas anteriores.

O laço das linhas 4 a 8 percorre todas as criticalidades previamente identificadas e armazenadas em *SolSet*. Para cada elemento j pertencente a esse conjunto solução, é realizada uma verificação de inclusão (linha 5), avaliando se a criticalidade j é um subconjunto da combinação atualmente analisada, representada por $Combs[Tid]$. Quando essa condição é satisfeita, conclui-se que a combinação em análise não pode corresponder a uma nova criticalidade minimal, pois já contém uma previamente identificada. Dessa forma, o valor correspondente em *isCrit* é atualizado para zero (linha 6), removendo a combinação da lista de candidatos que serão avaliados nas etapas subsequentes.

2.4.1.2 Adaptação da Etapa 3: Avaliação

A versão paralela da etapa 3 é realizada pelo Algoritmo 4. Ele é responsável pela avaliação das combinações de medidas geradas e previamente filtradas pela Etapa 2 de Poda. O objetivo dessa etapa é identificar quais combinações constituem uma C_k .

Algoritmo 4: Kernel da Etapa 3: Avaliação

```

Input:  $k, Combs, \Omega_1$ 
Output:  $isCrit$ 
1.  $Tid \leftarrow threadIdx.x + blockDim.x \times blockIdx.x$ 
2. if  $Tid < \binom{m_1}{k}$ 
3.    $build\_ \Omega_{Tid}^k(\Omega, k, Combs)$ 
4.   if  $\Omega_{Tid}^k$  is singular
5.      $isCrit[Tid] \leftarrow 1$ 
6.   else
7.      $isCrit[Tid] \leftarrow 0$ 
8.   end if
9. end if

```

Inicialmente, na linha 1, cada *thread* calcula seu identificador global (Tid), que determina a combinação a ser processada. Em seguida, a linha 2 verifica se o identificador corresponde a uma combinação válida, isto é, se $Tid < \binom{m_1}{k}$.

Para cada combinação válida, a linha 3 constrói a submatriz de covariância dos resíduos Ω_{Tid}^k , obtida a partir da matriz Ω e da combinação armazenada na linha correspondente da matriz $Combs$.

Na linha 4 é realizada a verificação de singularidade de Ω_{Tid}^k através da eliminação de Gauss. Conforme a fundamentação teórica apresentada no Capítulo 2, a singularidade de Ω_{Tid}^k constitui a condição necessária para que a combinação analisada seja classificada como uma C_k .

Caso a matriz seja singular, o elemento correspondente no vetor binário $isCrit$ recebe o valor 1 (linha 5). Caso contrário, é atribuído o valor 0 (linha 7), descartando a combinação das etapas subsequentes.

A principal característica dessa etapa é que cada combinação é analisada de forma completamente independente das demais. Consequentemente, cada *thread* executa a construção e a avaliação de uma única submatriz, permitindo que milhares de verificações de singularidade sejam realizadas simultaneamente na GPU. Como a análise de singularidade representa a operação computacionalmente mais custosa da AC, sua paralelização é res-

ponsável por grande parte do ganho de desempenho obtido pela implementação proposta.

Observa-se que a construção da submatriz e o teste de singularidade são realizados localmente por cada *thread*, eliminando dependências de memória entre combinações distintas e tornando essa etapa particularmente adequada à execução em arquiteturas de programação paralela.

Esta versão apresenta desempenho superior ao do algoritmo sequencial, porém, ela possui uma baixa escalabilidade. Pois, conforme apresentado na Figura 6, como as etapas de enumeração e de atualização são implementadas na CPU, ocorrem múltiplas transferências de dados entre a CPU e a GPU durante a execução. Durante o laço (de $k=1$ até k_{max}), a CPU envia à GPU as combinações de medidas ($Combs$) a serem avaliadas e recebe de volta as combinações críticas (C_{ks}) identificadas. O que impacta o desempenho da implementação por resultar momentos de sincronização e de transferência de dados entre arquiteturas distintas.

2.4.2 Segunda Versão: Quatro etapas executadas na GPU

A versão seguinte do algoritmo, apresentada na Figura 7, implementa todas as etapas da AC na GPU para minimizar as transferências de dados entre arquiteturas distintas, e consequentemente, aprimorar a escalabilidade das instâncias (Nishio; Do Coutto Filho; Stachinni de Souza *et al.*, 2024). Contudo, a implementação das duas etapas remanescentes em uma versão paralela para GPU não é trivial, principalmente devido às dependências existentes entre as operações realizadas de forma sequencial.

2.4.2.1 Adaptação da Etapa 1: Enumeração

Na implementação anterior, a geração das combinações (Etapa 1) era realizada pelo algoritmo *Cooler* (Ruskey; Williams, 2009), que obtém cada combinação de forma sequencial a partir da combinação anterior. Embora essa abordagem seja adequada para a execução em CPU, a dependência entre combinações dificulta sua paralelização eficiente em GPU.

Como alternativa, o Algoritmo 5 (Er, 1988) apresenta uma nova estratégia para a implementação da Etapa 1 da AC em GPU, responsável pela geração de todas as combinações possíveis de cardinalidade k dentre as m medidas disponíveis. Diferentemente do algoritmo *Cooler*, a abordagem proposta permite determinar cada combinação de forma independente. Essa característica elimina as dependências entre *threads* e, consequentemente, permite explorar de forma mais eficiente o paralelismo massivo proporcionado pela

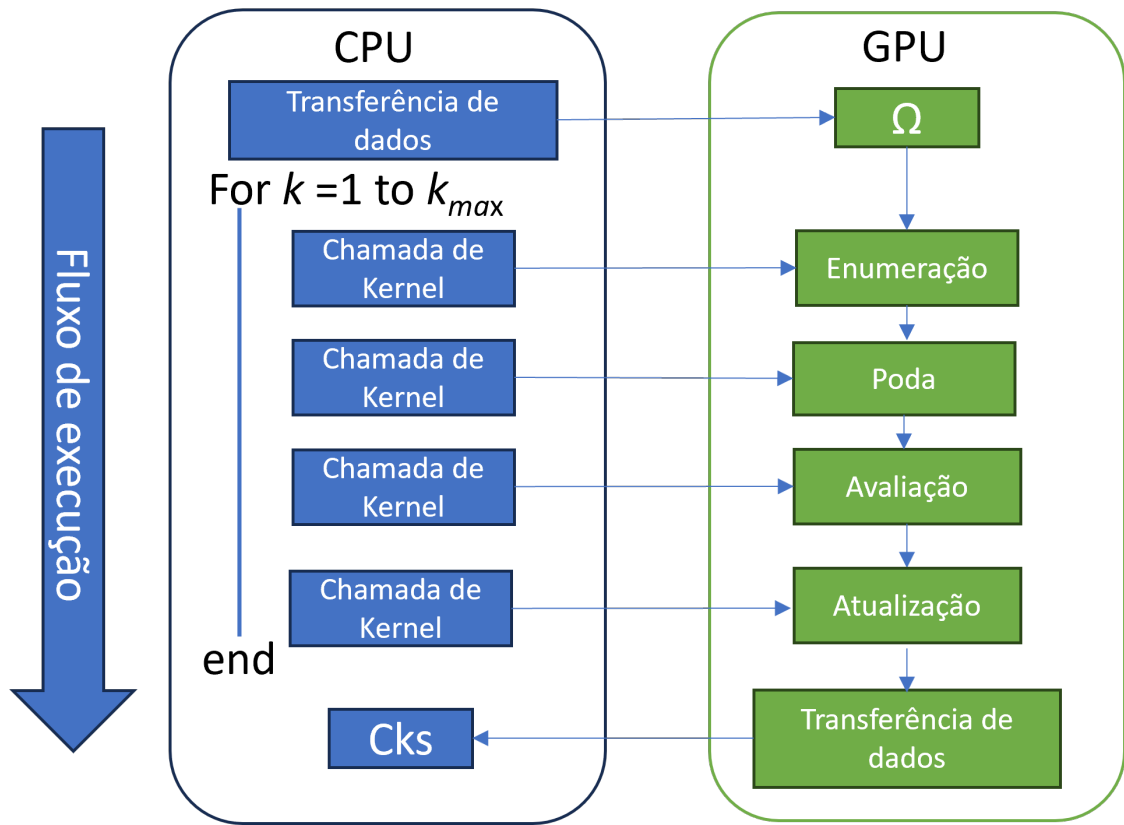


Figura 7: Etapas da AC e fluxo de dados na implementação em GPU.

GPU.

Inicialmente, na linha 1, cada *thread* calcula seu identificador global (*Thread ID*), denotado por *Tid*, a partir dos índices do bloco e da própria *thread*. Esse identificador define qual combinação (de acordo com a ordem lexicográfica) será processada pela *thread*.

Na linha 2, verifica-se se o identificador calculado corresponde a uma combinação válida, isto é, se $Tid < \binom{m}{k}$. Dessa forma, apenas as *threads* associadas a combinações existentes executam as etapas subsequentes do algoritmo.

As variáveis auxiliares *a*, *b* e *c*, inicializadas na linha 3, são utilizadas para converter o índice lexicográfico *Tid* na combinação correspondente. Em particular, a variável *a* armazena o índice da combinação a ser gerada, enquanto *b* e *c* controlam o número de elementos ainda disponíveis e o número de posições restantes a serem preenchidas.

O laço das linhas 4 a 16 percorre os possíveis índices de medidas, determinando quais deles pertencem à combinação associada à *thread*. A cada iteração, a variável *c* é decrementada (linha 5), representando a redução do número de elementos ainda necessários para completar a combinação. Além disso, os cálculos combinatórios (binomiais) das linhas 2 e 7 foram substituídos por valores pré-computados (*hard-coded*), reduzindo o custo

Algoritmo 5: Kernel da Etapa 1: Enumeração de Combinações

```

Input:  $m_1, k$ 
Output:  $Combs$ 
1.   $Tid \leftarrow threadIdx.x + blockDim.x \times blockIdx.x$ 
2.  if  $Tid < \binom{m}{k}$ 
3.       $a \leftarrow Tid; \quad b \leftarrow m; \quad c \leftarrow m - k$ 
4.      for  $j = 0$  to  $m$ 
5.           $c \leftarrow c - 1$ 
6.          if  $c \geq 0$ 
7.               $d \leftarrow \binom{m_1-1-j}{c}$ 
8.              if  $c < a$ 
9.                   $Combs[Tid][k - b] \leftarrow j$ 
10.                  $b \leftarrow b - 1; \quad c \leftarrow c + 1; \quad a \leftarrow a + d$ 
11.             end if
12.          else
13.               $Combs[Tid][k - b] \leftarrow j$ 
14.               $b \leftarrow b - 1$ 
15.          end if
16.      end for
17.  end if

```

computacional associado a essas operações.

Ao final da execução, cada *thread* produz exatamente uma linha da matriz *Combs*. Como consequência, todas as $\binom{m}{k}$ combinações podem ser geradas simultaneamente por diferentes *threads*, explorando o paralelismo massivo da GPU. A matriz *Combs* resultante é então utilizada nas etapas subsequentes da análise para avaliar a criticidade das combinações geradas.

2.4.2.2 Adaptação da Etapa 4: de Atualização

A etapa 4 de atualização também apresenta desafios para uma implementação paralela. Como o número de combinações avaliadas é muito menor que o número de criticalidades identificadas, a etapa de atualização utiliza a estrutura de dados de uma pilha para compactar as criticalidades encontradas. Porém, a manipulação desta estrutura é complexa e foi resolvida utilizando duas primitivas da programação paralela o *Scan* e *Compact*.

A Figura 8 ilustra o emprego das primitivas paralelas para a filtragem e o agrupamento das combinações críticas identificadas durante a execução da CA. O vetor **Combs** armazena as combinações de medidas analisadas, enquanto o vetor binário **isCrit** indica o resultado da avaliação de cada combinação, assumindo valor igual a 1 quando a combinação é

classificada como crítica e 0 caso contrário.

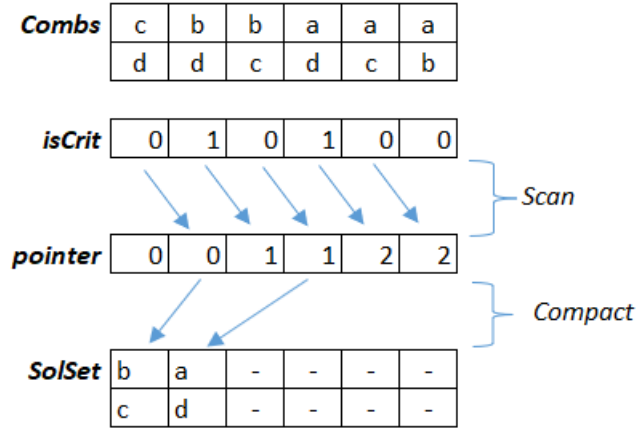


Figura 8: Etapas da AC e fluxo de dados na implementação em GPU.

Inicialmente, a operação *scan* é aplicada ao vetor **isCrit**. Essa primitiva calcula, para cada posição, a soma acumulada dos elementos anteriores do vetor. Como resultado, obtém-se o vetor **pointer**, cujos elementos indicam a posição que cada combinação crítica deverá ocupar no vetor de saída. No exemplo apresentado, o vetor **isCrit** = [0,1,0,1,0,0] gera o vetor **pointer** = [0,0,1,1,2,2].

Em seguida, a operação *compact* utiliza simultaneamente os vetores **isCrit** e **pointer** para transferir apenas as combinações críticas para o vetor de saída **SolSet**. Cada *thread* associada a uma combinação verifica o valor correspondente em **isCrit**; caso esse valor seja igual a 1, a combinação é copiada para a posição indicada pelo vetor **pointer**. Dessa forma, as combinações críticas tornam-se armazenadas de forma contígua em memória, eliminando lacunas e descartando automaticamente as combinações não críticas.

No exemplo da Figura 8, apenas as combinações {b,d} e {a,c} são identificadas como críticas. Após a execução da operação *compact*, essas combinações são armazenadas nas duas primeiras posições do conjunto solução **SolSet**, enquanto as demais posições permanecem vazias. Esse procedimento equivale ao comportamento de uma pilha compactada, na qual apenas os elementos de interesse são preservados. Em arquiteturas GPU, essa estratégia é amplamente empregada por permitir a seleção eficiente de elementos em paralelo, reduzindo movimentações desnecessárias de memória e evitando a utilização de operações atômicas para gerenciamento de índices globais.

2.4.3 Comparação de desempenho entre AC na CPU e na GPU

Este capítulo apresenta os resultados numéricos obtidos pelos artigos para comprovar a eficiência da implementação paralela da AC em GPU, a qual será utilizada nos capítulos seguintes para a execução da AC particionada.

A Tabela 1 apresenta os tempos de execução obtidos por Nishio, Do Coutto Filho, Stachinni de Souza *et al.* (2024) para a AC nos sistemas IEEE de 14, 30 e 118 barras, considerando uma implementação paralela em $CPU_{multithread}$ (*Intel Core i5-9300H*), desenvolvida em C++ com a API *OpenMP*, e duas plataformas GPU distintas.

A primeira plataforma consiste em uma GPU *NVIDIA GeForce GTX 1650*, dotada de 896 núcleos de processamento (*cores*) e capacidade computacional (*compute capability*) 7.5. Nessa plataforma foram executadas tanto a implementação original, em que apenas duas etapas da AC são realizadas na GPU ($GPU_{GTX}^{2etapas}$), quanto a implementação proposta nesta tese, na qual as quatro etapas do algoritmo são executadas integralmente na GPU ($GPU_{GTX}^{4etapas}$). A utilização da mesma plataforma em ambas as implementações permite uma comparação direta entre as duas abordagens.

A segunda plataforma corresponde ao supercomputador *NVIDIA DGX* da UFF, equipado com uma GPU *Tesla P100*, que possui 3.584 núcleos de processamento e capacidade computacional 6.0. Nessa plataforma, foi executada apenas a implementação de quatro etapas ($GPU_{DGX}^{4etapas}$), permitindo avaliar o desempenho da abordagem proposta em hardware de maior capacidade computacional.

Além dos tempos absolutos, este capítulo apresenta os fatores de aceleração (*speed-ups*) obtidos em relação à implementação em CPU.

Observa-se que, para os sistemas IEEE 14 e 30 barras, os tempos de execução já se encontram na ordem de milissegundos, independentemente da plataforma utilizada. Ainda assim, as GPUs proporcionam reduções significativas no tempo de processamento. No sistema IEEE 14 barras, por exemplo, o tempo de execução é reduzido de 224 ms na CPU para 16 ms na GPU GTX e apenas 4 ms na plataforma DGX. Resultados semelhantes são observados para o sistema IEEE 30 barras, no qual o tempo de execução passa de aproximadamente 4 s para 290 ms e 80 ms, respectivamente.

A Tabela 1 também compara as implementações $GPU_{GTX}^{2etapas}$ e $GPU_{GTX}^{4etapas}$. Observa-se que, para o sistema IEEE de 14 barras, ambas as abordagens apresentam desempenho semelhante, com uma pequena vantagem para a versão de duas etapas. Entretanto, à medida que o porte do sistema aumenta, a implementação em quatro etapas torna-se

progressivamente mais eficiente, alcançando aceleração de aproximadamente 4 vezes no sistema IEEE de 30 barras e de 26 vezes no sistema IEEE de 118 barras. Esse comportamento decorre da necessidade de sucessivas transferências de dados entre a CPU e a GPU na implementação de duas etapas. Embora esse custo seja pouco significativo para sistemas de pequeno porte, ele cresce com o volume de dados processados e passa a representar uma parcela expressiva do tempo total de execução. Utilizando o *Nvidea visual profiler* foi possível constatar que 40% do tempo de execução foi gasto na transferência de dados no caso de 118 barras. Ao manter todas as etapas da AC na GPU, a implementação proposta elimina esse gargalo de comunicação, reduzindo a sobrecarga de transferência de dados e proporcionando melhor escalabilidade para sistemas de maior dimensão

À medida que o porte do sistema aumenta, os benefícios da execução em GPU tornam-se ainda mais evidentes. Para o sistema IEEE 118 barras, a implementação em CPU requer aproximadamente 830 s para concluir a análise, enquanto a $GPU_{GTx}^{4etapas}$ reduz esse tempo para 44,86 s e a plataforma DGX para apenas 15 s. Essa redução substancial evidencia a capacidade das arquiteturas GPU de explorar o elevado grau de paralelismo presente na avaliação simultânea de combinações de medidas.

Os fatores de aceleração apresentados na Tabela 1 corroboram essa análise. Em comparação com a CPU multithread, a $GPU_{GTx}^{4etapas}$ proporciona ganhos de desempenho entre 13 e 19 vezes, enquanto a plataforma DGX alcança acelerações superiores a 50 vezes para todos os sistemas analisados. A relativa estabilidade desses fatores de aceleração indica que a implementação proposta apresenta boa escalabilidade, mantendo sua eficiência mesmo com o crescimento do tamanho do problema.

Tabela 1: Comparação dos tempos de execução e fatores de aceleração obtidos para diferentes plataformas computacionais.

Plataforma	14 barras	30 barras	118 barras
$CPU_{multithread}$	< 1 s (224 ms)	≈ 4 s (3.970 ms)	≈ 14 min (829,90 s)
$GPU_{GTx}^{2etapas}$	< 1 s (18 ms)	4s s (4,44 ms)	$\approx 20min$ s (1180,86 s)
$GPU_{GTx}^{4etapas}$	< 1 s (16 ms)	< 1 s (290 ms)	≈ 45 s (44,86 s)
$GPU_{DGX}^{4etapas}$	< 1 s (4 ms)	< 1 s (80 ms)	15 s
$GPU_{GTx}^{2etapas} / GPU_{GTx}^{4etapas}$	0.8×	4×	26×
$CPU_{multithread} / GPU_{GTx}^{4etapas}$	16×	13×	19×
$CPU_{multithread} / GPU_{DGX}^{4etapas}$	51×	52×	55×

2.4.4 Comparação com Branch & Bound

Finalmente, a versão mais recente do algoritmo de AC em GPU, apresentada na Seção 2.4.2 e em Nishio, Do Coutto Filho, Stacchini De Souza *et al.* (2024), compara os resultados de desempenho da implementação da AC em GPU com o algoritmo sequencial Branch & Bound (B&B) apresentado por (Braga Flor, Vinícius B. *et al.*, 2021). Contudo, uma vez que o trabalho de Vinícius B. Braga Flor *et al.* (2021) utiliza uma abordagem orientada a UMs, o algoritmo da GPU foi adaptado para permitir a comparação dos resultados.

Este trabalho compara os tempos de execução de dois novos algoritmos paralelos AC orientados a UMs, um deles implementado em CPU e o outro em GPU, com o algoritmo sequencial Branch-and-Bound (B&B) apresentado por Vinícius B. Braga Flor *et al.* (2021). Os estudos utilizaram o sistema IEEE de 118 barras, que contém 176 medições de fluxo de ramo fornecidas por 99 UMs. Além disso, todos os algoritmos realizaram a AC até $k = 5$. Para possibilitar uma comparação justa na mesma máquina, o algoritmo sequencial B&B foi reimplementado em C++ utilizando a biblioteca Armadillo e executado na mesma máquina utilizada para os testes em CPU e GPU. Os testes em GPU são executados em uma NVIDIA Titan X com arquitetura Pascal e capacidade computacional de 6.1, pertencente ao INESC TEC.

Primeiramente, é pertinente comparar o esforço computacional despendido pelos algoritmos analisados na etapa mais custosa da AC (Avaliação). Uma forma justa de realizar essa comparação é avaliar o número de combinações de UMs (sub-matrizes $\Omega_{(i)}^{(k)}$) processadas, uma vez que esse é um dos índices de desempenho mais relevantes do problema.

O algoritmo proposto em GPU, inicialmente, para cada cardinalidade k , forma uma lista com todas as combinações de UMs disponíveis na etapa de enumeração. As combinações que contêm UMs críticas de cardinalidade $(k-1)$ são descartadas na etapa 2 de Poda, reduzindo a carga computacional do algoritmo. Por outro lado, o algoritmo apresentado em Vinícius B. Braga Flor *et al.* (2021) faz uso da conhecida estratégia de busca em profundidade (DFS) (Neapolitan, 2015) para explorar o espaço de soluções. Essa abordagem desperdiça tempo avaliando desnecessariamente combinações de UMs pouco promissoras — isto é, aquelas que contêm criticalidades de ordem inferior — exigindo um processo de poda pós-avaliação.

A Tabela 2.4.4 compara o número de submatrizes avaliadas, $\Omega_{(i)}^{(k)}$, para cada cardinalidade k (até 5), pelos algoritmos estudados. O processo de poda favorece um melhor desempenho da CA, reduzindo significativamente o número de verificações de observabili-

dade. Pode-se concluir que a sequência de etapas enumeração–poda–avaliação, executada pelo algoritmo proposto em GPU, avalia aproximadamente oito milhões de submatrizes de covariância a menos do que o algoritmo B&B (Braga Flor, Vinícius B. *et al.*, 2021).

Tabela 2: Comparação entre o Algoritmo B&B e o Algoritmo Proposto em GPU

Card. (k)	Sem Poda	B&B	Proposto GPU
1	99	99	99
2	4.851	4.029	3.402
3	156.849	107.161	85.145
4	3.764.376	2.046.665	1.574.694
5	71.523.144	29.963.362	22.390.026
Total	75.449.319	32.121.316	24.053.366

O desempenho dos algoritmos AC depende fortemente do limite superior estabelecido para a cardinalidade k dos tuplas críticas de UMs. A figura 2.4.4 apresenta a relação dos tempos de execução (em segundos — escala logarítmica), obtidos com $k \leq 5$, considerando a implementação do algoritmo B&B e dos algoritmos paralelos propostos em CPU e GPU. Os valores de tempo de execução são calculados de forma cumulativa, uma vez que a busca por C_{5-UMs} depende das criticidades de ordens inferiores.

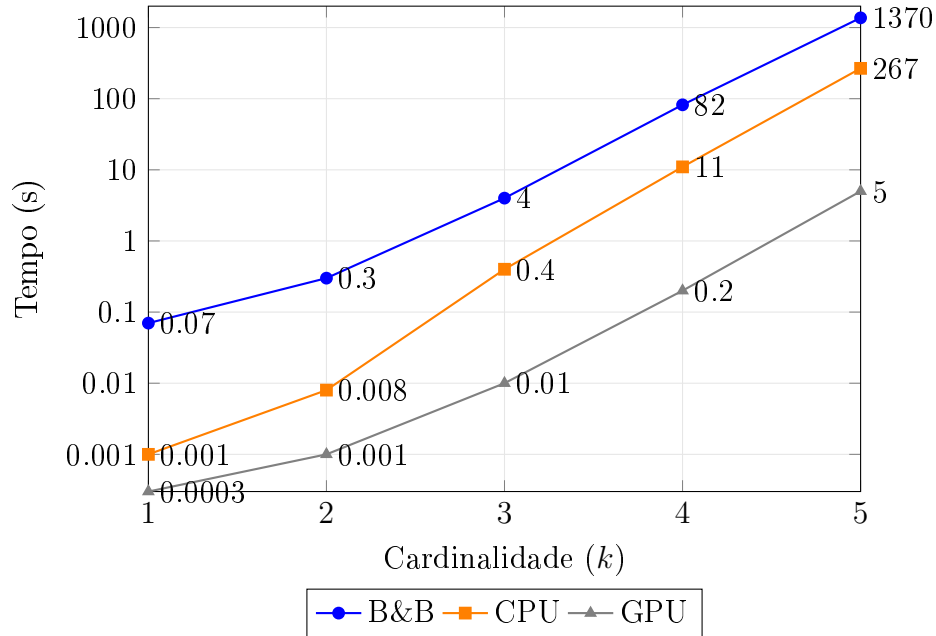


Figura 9: Tempo de execução em função da cardinalidade (k).

Como pode ser observado, o algoritmo paralelo em GPU supera seus concorrentes, apresentando um tempo total de execução de 5,40s, enquanto o algoritmo paralelo em CPU requer 84,78s (ou 1,41 min) e o algoritmo sequencial B&B necessita de 1.456s (ou 24,27 min). Esses resultados demonstram que o algoritmo paralelo proposto em GPU

é 269 vezes mais rápido do que a abordagem sequencial B&B. Além disso, a GPU é a arquitetura mais adequada para executar o ExtCA orientado a UMs, uma vez que a execução paralela na GPU é 17 vezes mais rápida do que na CPU.

Apesar dos resultados promissores, que viabilizaram o estudo de casos de grande porte em um tempo viável, a natureza do problema não é alterada pela implementação na GPU. A análise continua intratável para casos reais de maior porte, pois o tempo de execução cresce em proporção fatorial com o número de medidas no SM.

Com o objetivo de solucionar estas questões, este trabalho apresentará, nos próximos capítulos, a exploração das características topológicas da análise para segmentar o SM em grupos de medidas de menor dimensão. A análise proposta na GPU pode ser realizada em cada grupo, de forma concorrente, em múltiplas GPUs. O algoritmo realiza a análise em todos os grupos e, por fim, o resultado da AC do sistema como um todo será a união de todas as criticalidades identificadas.

3 Estratégias de Particionamento

Conforme comentado no capítulo anterior, a Análise de Criticalidades (AC) é um problema de natureza combinatória intratável. Partindo deste princípio, este trabalho propõe metodologias de particionamento do sistema de medição (SM) para viabilizar a análise em casos de maior porte. Neste capítulo, serão apresentadas as principais estratégias de modelagem e particionamento avaliadas durante a pesquisa.

3.1 Metodologia

A metodologia proposta parte da hipótese de que as criticalidades apresentam comportamento predominantemente local. Dessa forma, ao particionar o SM em grupos topologicamente conectados, espera-se que a maior parte das criticalidades permaneça integralmente contida nos grupos. Essa característica permite substituir uma única análise combinatória de grande porte por diversas análises menores, reduzindo significativamente o custo computacional.

Primeiramente, como o objetivo é aprimorar o desempenho da AC, é necessário destacar os principais fatores que influenciam o tempo de execução da AC::

1. A dimensão da rede elétrica, considerando o número de barras (nós elétricos) e suas interconexões (ramos da rede);
2. A quantidade, o tipo e a localização das medidas disponíveis no SM;
3. A cardinalidade alvo das tuplas de medidas a serem avaliadas quanto à observabilidade;
4. restrições de tempo impostas pela aplicação.

Uma vez estabelecida a representação do problema, a metodologia proposta é organizada em um conjunto de etapas sequenciais. A estratégia proposta consiste em substituir

uma única execução da AC sobre todo o sistema por diversas execuções independentes realizadas sobre subsistemas menores. Inicialmente, o SM é representado por meio de uma estrutura matemática apropriada, permitindo a aplicação de algoritmos de particionamento. Em seguida, o sistema é dividido em subsistemas balanceados, nos quais a AC é executada de forma independente. Por fim, as criticalidades identificadas em cada partição são refinadas e integradas para reconstruir o conjunto de criticalidades do sistema completo. As seções seguintes descrevem detalhadamente cada uma dessas etapas.

3.2 Definição do Problema de Particionamento

A metodologia proposta representa o SM como um grafo no qual a proximidade topológica entre as medidas é preservada. Essa representação permite aplicar algoritmos clássicos de particionamento de grafos para dividir o problema em subproblemas menores. Antes de apresentar os algoritmos de particionamento, é necessário formalizar o problema considerado nesta tese e definir a representação matemática utilizada para o sistema de medição

Sem perda de generalidade, considera-se o seguinte exemplo ilustrativo para facilitar a compreensão do problema. A Figura 10 apresenta um SM composto por 9 medidas: 6 medidas de fluxo de potência em ramos (P_{i-j}) e 3 medidas de injeção de potência em barras (P_i). Ressalta-se que medidas de corrente em ramos e ângulos de fase de tensão em barras, provenientes de unidades de medição fasorial (PMUs), também poderiam ser incorporadas ao modelo.

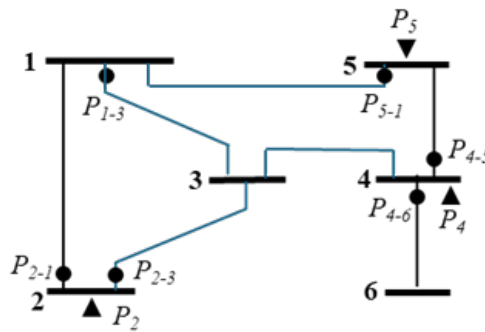


Figura 10: Sistema exemplo de 6 barras e 9 medidas.

O número de medidas disponíveis desempenha um papel fundamental na avaliação da complexidade da CA, conforme definido pela equação 2.32. O particionamento balanceado

da rede contribui para essa análise ao reduzir o número de combinações de medidas, m_{comb} que precisam ser avaliadas em cada grupo.

Ao adotar $k = 4$ e considerar o sistema completo com 9 medidas, o número de combinações de medidas analisadas pela AC é $m_{\text{comb}} = 255$. Em contrapartida, ao particionar o sistema em dois grupos desbalanceados — G_1 , contendo as barras #1, #2, #3 e #5, e G_2 , contendo as barras #4 e #6 — observa-se uma redução significativa em m_{comb} . Especificamente, G_1 , com 6 medidas, resulta em $m_{\text{comb}} = 56$, enquanto G_2 , com 3 medidas, produz $m_{\text{comb}} = 7$. Em conjunto, as 63 combinações representam uma redução de aproximadamente 75,7%.

Alternativamente, ao particionar a rede em dois grupos mais balanceados — G_1 , contendo as barras #1, #2 e #3, e G_2 , contendo as barras #4, #5 e #6 — cada um com 4 e 5 medidas, respectivamente, obtém-se $m_{\text{comb}} = 45$. A redução de 255 para 45 corresponde a aproximadamente 82,4%, evidenciando a eficácia dessa estratégia de particionamento. Adicionalmente, quando os subgrafos são desbalanceados, o custo computacional da AC tende a ser dominado pelo tempo de execução associado ao subgrafo com maior número de medidas.

Considerando que o número de medidas e sua distribuição ao longo da rede constituem fatores de fundamental importância para a execução da CA, o objetivo do particionamento consiste em obter subgrafos balanceados. Dessa forma, busca-se reduzir o número de combinações a serem avaliadas em cada partição, diminuindo, conseqüentemente, o custo computacional e o tempo de execução da CA.

O número de partições (grupos), denotado por n_g , pode ser definido em função da quantidade total de medidas (m) disponíveis no sistema. Uma vez que se buscam grupos balanceados, uma escolha natural para n_g seria o valor ideal de medidas igualmente distribuídas pelos grupos ($\frac{m}{n_g}$), tal que permita a execução do problema em um tempo razoável. Além disso, a disponibilidade de recursos computacionais, particularmente unidades de processamento em paralelo, pode influenciar diretamente a escolha de n_g , permitindo a distribuição simultânea do processamento entre diferentes partições.

Sendo assim, o SM deve ser modelado como um grafo para permitir a aplicação de técnicas de particionamento que dividam o problema original em subproblemas menores e mais facilmente tratáveis.

3.3 Modelagem via Grafos

Inicialmente, constrói-se um grafo conexo $G(V, E)$, em que V representa o conjunto de vértices e E corresponde ao conjunto de arestas que conectam estes vértices.

O problema consiste em particionar o grafo $G = (V, E)$ em n_g subgrafos conexos $G_1, G_2, \dots, G_{n_g}$, com $n_g > 1$, de forma que a distribuição das medidas entre as partições seja a mais balanceada possível.

Além do critério de balanceamento, como a proximidade física das medidas é um fator relevante para o problema, impõe-se a restrição de conectividade, segundo a qual cada subgrafo (G_i) deve constituir uma componente conexa. Assim, para quaisquer vértices $(u, v \in G_i)$, deve existir um caminho entre (u) e (v) inteiramente contido em (G_i) .

Nesta Seção são apresentadas duas formas de representar o SM. Primeiramente, é descrita a modelagem por multigrafo, amplamente utilizada na literatura e diretamente relacionada ao problema da AC, que está presente nos trabalhos de [Simoes Costa, Piazza e Mandel \(1990\)](#) e de [Soares Lima *et al.* \(2022\)](#). Em seguida, propõe-se uma modelagem simplificada baseada em um grafo da topologia, adotada nesta tese por facilitar a aplicação de algoritmos clássicos de particionamento. Finalmente, apresenta-se uma formulação em programação inteira para formalizar matematicamente o problema.

3.3.1 Multigrafo Grafo do SM

Uma representação do SM que utiliza multigrafos pode ser encontrada por [Soares Lima *et al.* \(2022\)](#). Diferentemente de um grafo simples, o multigrafo permite múltiplos ramos entre seus nós terminais. Do ponto de vista da EE, o multigrafo representa a correspondência entre ramos e medidas. Esse multigrafo é chamado de grafo de medições, ou mais especificamente, multigrafo de medições.

A figura 11 representa o multigrafo de medição do sistema de 6 barras e 9 medidas. Na modelagem utilizada, cada barra do sistema é representada por um vértice no multigrafo $\mathcal{G}$, já as arestas são adicionadas da seguinte forma:

- Para cada medida de fluxo de potência P_{k-m} , no ramo $k - m$, há uma aresta entre os vértices k e m de G . A Figura 11 ilustra a representação de uma medida de fluxo de potência P_{1-3} no ramo 1 - 3, como aresta de fluxo.
- Para cada medida de injeção de potência P_k , na barra k , há um conjunto de arestas

com um extremo em k e outro extremo nas barras vizinhas de k . A Figura 11 exemplifica a representação de uma medida de injeção P_4 , na barra 1, cujas barras adjacentes são 3, 5 e 6, como arestas de injeção.

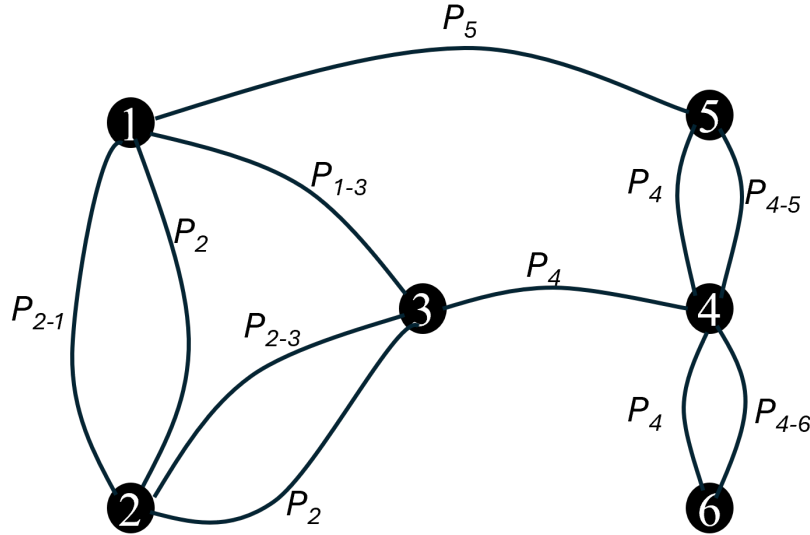


Figura 11: Multigrafo de medição do sistema exemplo de 6 barras. Nesta modelagem, os vértices representam as barras do sistema e as arestas representam as medidas do plano.

Uma propriedade relevante desta modelagem é a possibilidade de expressar a observabilidade da rede. Além da robustez da redundância do SM. Uma rede observável possui uma representação conexa em multigrafo de medição, ou seja, existe um caminho entre quaisquer dois pares de vértices que não repita arestas associadas à mesma medida.

Embora o multigrafo represente explicitamente a correspondência entre medidas e elementos da rede, o objetivo desta tese é dividir o sistema em regiões topologicamente conectadas e balanceadas. Então, uma representação baseada apenas na topologia elétrica torna-se suficiente e reduz significativamente a complexidade da modelagem.

3.3.2 Grafo da Topologia

Nesta modelagem, constrói-se um grafo conexo $G(V,E)$, em que V representa o conjunto de vértices associados às barras do sistema elétrico, e E representa o conjunto de arestas que correspondem aos ramos da rede. A cada vértice é atribuído um peso não negativo, correspondente ao número de medidas instaladas naquela barra. Vale destacar que medidas de fluxo de potência P_{k-m} , associadas a ramos $k-m$, são consideradas instaladas na barra k .

A Figura 10 apresenta o grafo da topologia do sistema de 6 barras e 9 medidas, conforme a modelagem do grafo da topologia. Nela, os balões representam o peso atribuído a cada vértice.

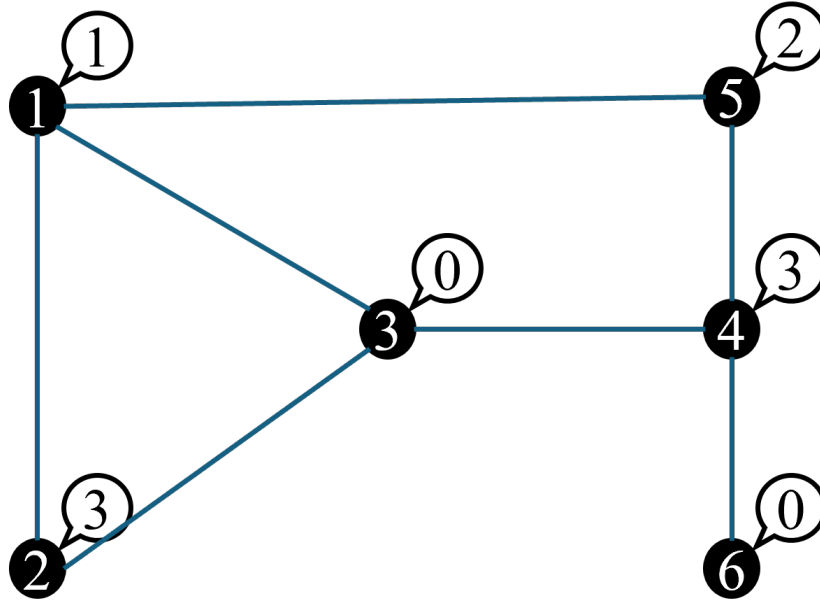


Figura 12: Grafo da topologia do sistema exemplo de 6 barras.

A modelagem apresentada permite aplicar algoritmos de particionamento originalmente desenvolvidos para grafos ao problema da AC, constituindo a base da metodologia proposta. Uma vez definida a representação do sistema por meio de um grafo topológico, torna-se possível formalizar o problema de particionamento como um modelo de programação inteira (PI).

3.3.3 Formulação via Programação Inteira

Embora a resolução ótima do problema por PI seja limitada a instâncias de menor porte, a formulação possui papel importante por formalizar matematicamente o problema de particionamento e servir como referência para a avaliação das heurísticas propostas.

Sendo assim, considere o grafo da topologia $G = (V, E)$, onde cada vértice $i \in V$ possui um peso associado w_i , correspondente ao número de medidas relacionadas à barra i do sistema elétrico. O objetivo consiste em particionar o conjunto de vértices em n_g grupos conexos, de forma que os pesos totais associados aos grupos sejam aproximadamente balanceados.

3.3.3.1 Variáveis de decisão

A formulação proposta busca obter partições balanceadas e conexas. Para isso são definidas as seguintes variáveis de decisão. Inicialmente, define-se a variável binária de pertencimento:

$$x_{ig} = \begin{cases} 1, & \text{se o vértice } i \text{ pertence ao grupo } g \\ 0, & \text{caso contrário} \end{cases} \quad (3.1)$$

3.3.3.2 Função objetivo

Define-se inicialmente o peso médio ideal das partições como:

$$\bar{W} = \frac{\sum_{i \in V} w_i}{n_g} \quad (3.2)$$

O peso total associado ao grupo g é definido por:

$$W_g = \sum_{i \in V} w_i x_{ig} \quad (3.3)$$

Com o objetivo de quantificar o desbalanceamento entre os grupos, define-se a variável contínua não negativa d_g , correspondente ao desvio absoluto entre o peso do grupo g e o peso médio ideal $\bar{W}$:

$$d_g = |W_g - \bar{W}|, \quad \forall g \quad (3.4)$$

Uma vez que o módulo não é uma função linear, a linearização desse desvio é obtida por meio das seguintes restrições:

$$d_g \geq W_g - \bar{W}, \quad \forall g \quad (3.5)$$

$$d_g \geq \bar{W} - W_g, \quad \forall g \quad (3.6)$$

A função objetivo do problema consiste em minimizar o desbalanceamento total entre os grupos:

$$\min \sum_{g=1}^{n_g} d_g \quad (3.7)$$

3.3.3.3 Restrição de unicidade de grupos

Para garantir que cada vértice seja alocado em um dos grupos, impõe-se a seguinte restrição de unicidade:

$$\sum_{g=1}^{n_g} x_{ig} = 1, \quad \forall i \in V \quad (3.8)$$

Esta restrição também pode ser adaptada para permitir que um vértice possa pertencer a mais de um grupo, caso seja necessário considerar sobreposições entre partições. Contudo, para alcançar um resultado ótimo igual ao dos métodos heurísticos propostos sem sequência, optou-se por manter a restrição de unicidade.

3.3.3.4 Restrições de conectividade

Considerando que a AC é um problema de natureza local. A restrição de conectividade é crucial para garantir que cada partição seja formada por elementos topologicamente próximos.

A conectividade de cada partição (g) é garantida pela existência de uma árvore geradora (*spanning tree*) em seu subgrafo induzido. Como uma árvore geradora corresponde a um subgrafo acíclico que conecta todos os vértices da partição, sua presença assegura que o grupo seja conexo. Para modelar essa condição, introduz-se a variável binária auxiliar (y_e^g), associada às arestas do grupo (g). Dessa forma, a conectividade é imposta indiretamente ao exigir que as arestas selecionadas por (y_e^g) constituam uma árvore geradora. A inicialização dessa variável é definida por:

$$y_e^g = \begin{cases} 1, & \text{se a aresta } e \text{ pertence à estrutura auxiliar do grupo } g \\ 0, & \text{caso contrário} \end{cases} \quad (3.9)$$

Para cada grupo g avaliado, a aresta auxiliar somente pode ser selecionada se ambos os vértices extremos pertencem a g . Tal condição é definida pelas seguintes restrições:

$$y_{ij}^g \leq x_{ig}, \quad \forall (i,j) \in E, \forall g \quad (3.10)$$

$$y_{ij}^g \leq x_{jg}, \quad \forall (i,j) \in E, \forall g \quad (3.11)$$

Essas restrições garantem consistência entre as variáveis de vértice x_{ig} e as variáveis auxiliares de aresta y_e^g . Além disso, é importante frisar que y_e^g não representam necessariamente todas as arestas presentes no subgrafo induzido pela partição g . Essas variáveis são auxiliares e têm a finalidade exclusiva de avaliar a restrição de conectividade.

A Equação (3.12) define a cardinalidade característica de uma árvore geradora. Para cada grupo g , impõe-se que o número de arestas auxiliares selecionadas, que definam uma árvore geradora dentro do grupo, seja igual ao número de vértices pertencentes ao grupo menos uma unidade:

$$\sum_{e \in E} y_e^g = \sum_{i \in V} x_{ig} - 1, \quad \forall g \quad (3.12)$$

Esta condição é necessária, porém não suficiente para garantir conectividade, uma vez que estruturas desconexas contendo ciclos também podem satisfazer essa igualdade. Sendo assim, também são adicionadas restrições de eliminação de subciclos. Para este fim, considere um subconjunto arbitrário $S \subseteq V$ e seja $E(S)$ o conjunto de arestas internas ao subconjunto S . Então, impõe-se:

$$\sum_{e \in E(S)} y_e^g \leq |S| - 1, \quad \forall S \subseteq V, \forall g \quad (3.13)$$

As restrições apresentadas em (3.13) impedem a formação de ciclos na estrutura auxiliar associada às variáveis y_e^g .

A combinação das Equações (3.12) e (3.13) garante que as arestas auxiliares selecionadas formem uma árvore geradora que conecte todos os vértices pertencentes do grupo. Garantindo, assim, a conectividade de cada partição.

3.3.3.5 Complexidade da Formulação

Utilizando a formulação de programação inteira apresentada, é possível obter uma solução ótima para o particionamento do SM apresentado na Figura 13; em que o grafo G é

particionado em dois subgrafos G_1 definido pelas barras 1, 2 e 3 e G_2 , definido pelas barras 4, 5 e 6. Este particionamento alcançou o balanceamento ótimo, retornando dois grupos com 4 e 5 medidas, respectivamente.

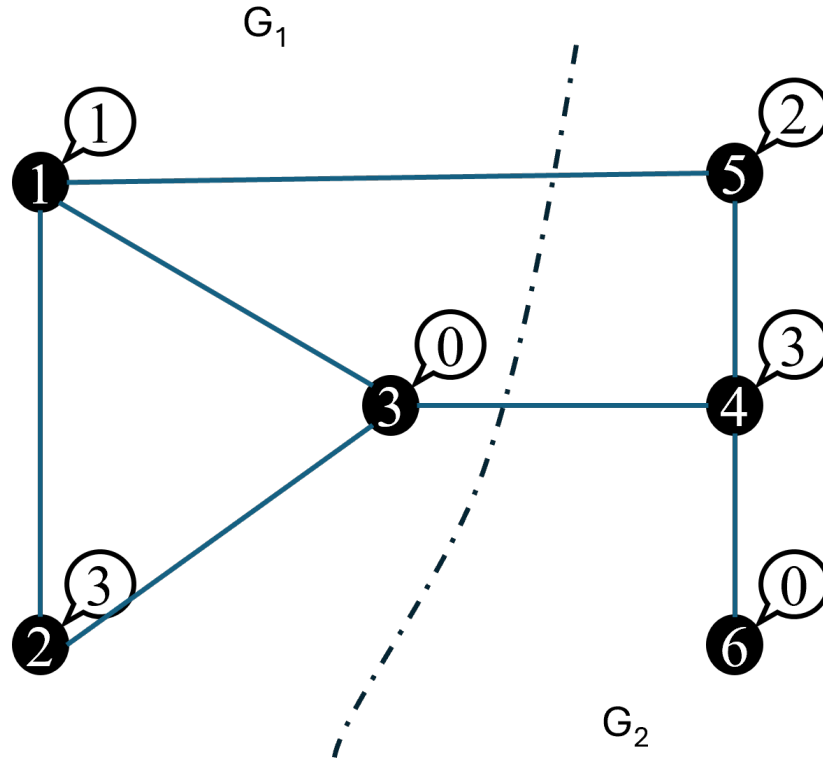


Figura 13: Particionamento ótimo do sistema exemplo de 6 barras obtido por meio da formulação de programação inteira.

No entanto, é importante destacar que o número de variáveis e restrições cresce exponencialmente com o tamanho do grafo, tornando a resolução computacionalmente inviável para instâncias de grande porte. Esta alta complexidade é atribuída à restrição de conectividade, que precisa considerar todos os subconjuntos possíveis de vértices para garantir a ausência de ciclos. Definindo uma complexidade computacional de $O(2^{|V|})$. Sendo assim, para avaliar o particionamento de um sistema de 118 barras, seria necessário considerar $2^{118} = 3,8 \times 10^{35}$ subconjuntos de vértices, o que é impraticável. Esse resultado confirma que a formulação é NP-difícil, o que torna sua implementação em sistemas de grande porte custosa.

Como o problema da AC já possui alta complexidade computacional, a adição de uma formulação de programação inteira, de natureza combinatorial, para o particionamento não viabilizaria a análise. Por este motivo, torna-se atrativa a utilização de algoritmos heurísticos que, embora não garantam a obtenção de soluções ótimas, apresentam um

custo computacional significativamente menor, permitindo a resolução de instâncias de grande porte em tempo razoável.

3.4 Algoritmos Heurísticos de Particionamento

Utilização de algoritmos heurísticos de particionamento é uma abordagem prática para lidar com a complexidade computacional em casos de grande porte. Como a qualidade do particionamento influencia diretamente a eficiência e a precisão da AC particionada, foram selecionadas quatro estratégias heurísticas com características distintas. Além do algoritmo heurístico desenvolvido nesta tese (Esquema H), foram considerados três métodos amplamente utilizados na literatura de particionamento de redes elétricas ([Ayall et al., 2022](#)), permitindo comparar diferentes critérios de formação das partições:

- **Esquema H:** método heurístico baseado em árvore geradora, desenvolvido nesta tese para gerar partições balanceadas e conectadas.
- **Esquema S:** particionamento espectral baseado na minimização do número de cortes do grafo.
- **Esquema K:** agrupamento k-means aplicado aos vértices da rede.
- **Esquema M:** detecção de comunidades por meio da maximização da modularidade.

3.4.1 Heurístico baseado em árvore geradora

O Esquema H é um método heurístico de particionamento de grafos desenvolvido durante o período de pesquisa, com o objetivo de gerar partições de tamanho aproximadamente uniforme. Este algoritmo cria um particionamento inicial a partir da árvore geradora de G , seguido por um processo iterativo de realocação de vértices para aprimorar o balanceamento das partições.

O algoritmo de particionamento em grupos balanceados utiliza o grafo G apresentado na Seção 3.3.2 e é composto por três etapas principais, Identificação das folhas, Particionamento e Balanceamento, descritas detalhadamente a seguir.

3.4.1.1 Primeira etapa: Identificação das folhas

Inicialmente, o algoritmo constrói uma árvore geradora de G . Em seguida, identificam-se os vértices folha (nós de grau 1) dessa árvore. Esses vértices são então utilizados para a inicialização do particionamento.

Como exemplo, a Figura 14 apresenta a árvore geradora do grafo ilustrado na Figura 12. Nesse caso, os vértices #2, #5 e #6 são classificados como folhas. Considerando o particionamento da rede em dois subgrafos, G_1 e G_2 , os vértices folha #2 e #6 são selecionados arbitrariamente para inicializar o algoritmo.

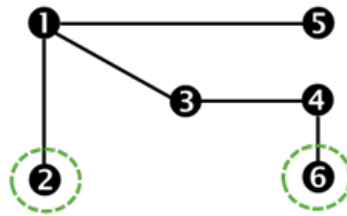


Figura 14: Árvore geradora do sistema exemplo de 6 barras. Vértices folha 2 e 6 destacados para inicialização do particionamento.

3.4.1.2 Segunda etapa: Particionamento

Nesta etapa, o algoritmo obtém um particionamento inicial, associando cada vértice folha identificado a um subgrafo distinto. Depois disso, cada vértice restante do grafo G é alocado no subgrafo correspondente à folha mais próxima. Neste trabalho, a distância é calculada com base no número de arestas entre os vértices.

No exemplo considerado, o vértice #3 é alocado ao subgrafo G_1 , por estar mais próximo da folha #2. Os vértices #4 e #6 são atribuídos ao subgrafo G_2 , uma vez que a folha #5 é a mais próxima. Já o vértice #1 apresenta a mesma distância das folhas #2 e #5, sendo, portanto, alocado arbitrariamente ao subgrafo G_2 .

Como resultado dessa etapa, observa-se que o subgrafo G_1 contém 5 medidas, enquanto G_2 reúne 9 medidas, o que caracteriza um particionamento desbalanceado do grafo G .

3.4.1.3 Terceira etapa: Balanceamento das partições

Nesta etapa, o algoritmo busca aprimorar o particionamento por meio da realocação iterativa de vértices entre os subgrafos, com o objetivo de reduzir o desbalanceamento.

Para tal, propõe-se o seguinte índice de balanceamento, I_b :

$$I_b = \sum_{i=1}^{n_g} \left| m_i - \frac{m}{n_g} \right| \quad (3.14)$$

em que n_g representa o número de subgrafos nos quais G foi particionado, e m_i denota o número de medidas no i -ésimo subgrafo.

A Figura 15 apresenta um diagrama de blocos simplificado da estratégia de balanceamento proposta. Primeiramente, calcula-se o índice de balanceamento corrente, I_b . Posteriormente, o algoritmo identifica o subgrafo com o menor número de medidas. A partir disso, um barramento de fronteira é selecionado (de forma arbitrária) para ser transferido para esse subgrafo, aumentando assim o número de medidas desse subgrafo.

Após a transferência, calcula-se um novo índice de balanceamento, denotado por $I_b^{(\text{nov})}$. O algoritmo então verifica se houve melhoria no balanceamento, o que ocorre quando $I_b^{(\text{nov})} < I_b$. Caso essa condição seja satisfeita, a modificação é mantida.

O processo de avaliação e realocação é repetido para todas as transferências possíveis, até que não sejam observadas melhorias significativas no índice de balanceamento. As iterações prosseguem até que não ocorram mais atualizações nas partições, sendo o número máximo de iterações, i_{count} , definido conforme critérios previamente estabelecidos.

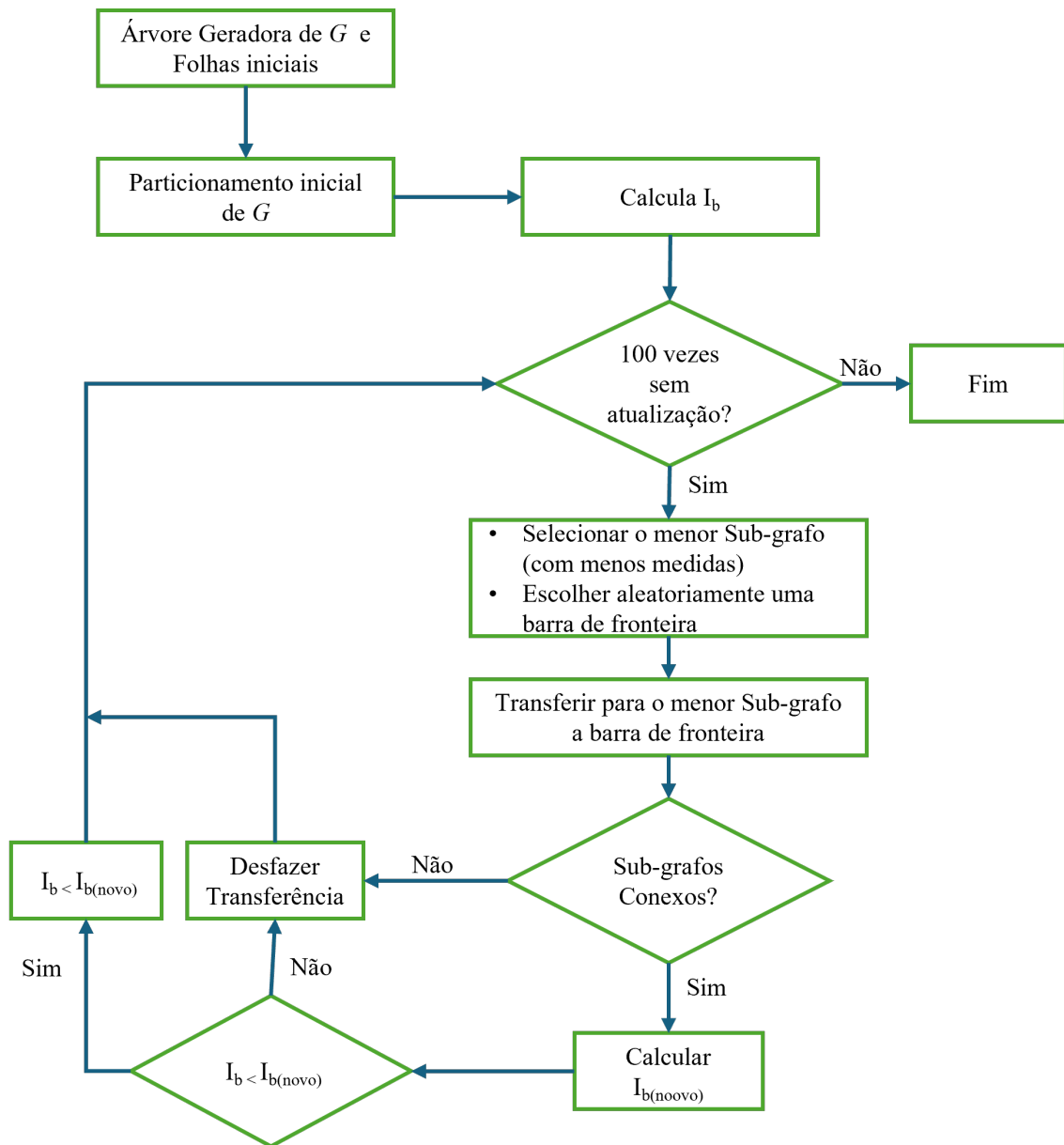


Figura 15: Fluxograma do algoritmo de particionamento do Esquema H.

Dando continuidade ao exemplo, na terceira etapa, o objetivo é em balancear os subgrafos previamente formados, G_1 e G_2 . A partir da Equação 3.14, obtém-se $I_b = 3$, o que indica que o particionamento inicial é desbalanceado.

Os vértices de fronteira #3 e #5 são identificados como candidatos à realocação. No entanto, a transferência do vértice #3 não resulta em melhoria no balanceamento, uma vez que provoca a desconexão de vértices, tornando os vértices #5 e #6 isolados, o que viola a condição de conectividade dos subgrafos.

Por outro lado, a realocação do vértice #5 resulta em um particionamento balanceado. Após essa operação, os subgrafos G_1 e G_2 passam a conter 4 e 5 medidas, respectivamente.

Essa nova distribuição de vértices reduz o índice de balanceamento para $I_b^{(\text{novo})} = 1$.

Dessa forma, obtém-se uma partição de balanceamento ótima que também foi alcançada pela formulação de programação inteira, conforme ilustrado na Figura 16.

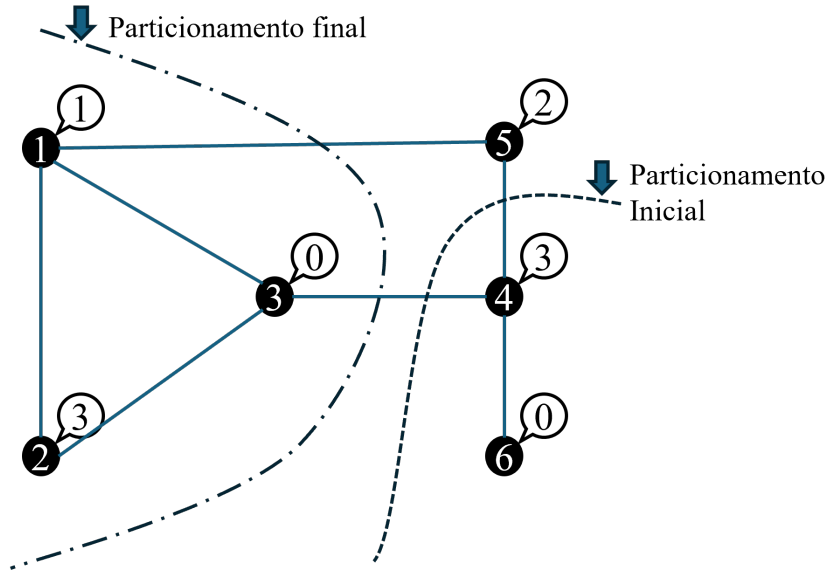


Figura 16: Particionamento final obtido pelo Esquema H.

3.4.2 Clusterização Espectral

A utilização de análise espectral para o particionamento de redes elétricas apresenta diversas vantagens, dentre as quais destacam-se: a capacidade de obter soluções de alta qualidade global, em contraste com métodos puramente locais; a identificação de sub-redes fracamente conectadas, contribuindo para o aprimoramento do controle e da eficiência; a robustez do método, uma vez que a etapa espectral não depende de condições iniciais; e a obtenção de partições fundamentadas nas propriedades intrínsecas da rede (Shi; Malik, 2000).

Durante a aplicação do particionamento espectral, constrói-se inicialmente a matriz de adjacência ($\mathbf{A}$) de G e, a partir desta, a matriz Laplaciana correspondente. Os autovalores e autovetores da matriz Laplaciana são então calculados e utilizados para extrair informações sobre a estrutura topológica do grafo, permitindo a identificação de partições com elevada coesão interna e com um número reduzido de conexões entre grupos distintos.

A matriz $\mathbf{A}$ é quadrada ($n \times n$), simétrica e possui elementos nulos na diagonal principal. Ela representa a conectividade do grafo, ponderadas de acordo com as medidas envolvidas em cada ramo da rede. Sabendo disso, $\mathbf{A}$ é construída de acordo com as

seguintes regras:

- Se existir uma aresta entre os vértices i e j , então:

$$A(i,j) = [\text{número de medidas de fluxo no ramo } (i - j)] + [\text{número de medidas de injeção nos vértices } i \text{ e } j] \quad (3.15)$$

- Caso contrário:

$$A(i,j) = 0 \quad (3.16)$$

A Figura 17 apresenta a matriz $\mathbf{A}$ construída para o sistema ilustrado na Figura 10, após a aplicação do Esquema S. Para a aresta $(4 - 5)$, há a medida de fluxo P_{4-5} , além das medidas de injeção P_4 e P_5 , totalizando 3 medidas; portanto, $A(4,5) = 3$. Por outro lado, $A(1,6) = 0$, uma vez que não há aresta conectando os vértices #1 e #6. A matriz $\mathbf{A}$ constitui a entrada comum dos Esquemas S e M.

	#1	#2	#3	#4	#5	#6
#1	0	2	1	0	2	0
#2	2	0	2	0	0	0
#3	1	1	0	0	0	0
#4	0	0	0	0	3	2
#5	2	0	0	3	0	0
#6	0	0	0	2	0	0

Figura 17: Matriz de adjacências $\mathbf{A}$ construída para o sistema exemplo de 6 barras para a execução do Esquema S.

Neste trabalho, utiliza-se o módulo de *spectral clustering* da biblioteca de código aberto Scientific Python (SciPy). O código-fonte da biblioteca está disponível em: <https://github.com/scipy/scipy>. Esse módulo recebe como entrada a matriz $\mathbf{A}$ e o número desejado de clusters (partições), denotado por n_g , e realiza o particionamento do grafo com base na conectividade topológica dos vértices e no grau de afinidade entre eles.

No exemplo considerado, a aplicação do *spectral clustering* resultou no mesmo particionamento balanceado obtido pelo Esquema H, conforme ilustrado na Figura 16.

3.4.3 Modularidade Gulosa

O método de Modularidade Gulosa (Esquema M) é uma abordagem aglomerativa para detecção de comunidades em redes (Clauset; Newman; Moore, 2004). Essa abordagem utiliza a modularidade como métrica para avaliar a qualidade da divisão da rede em comunidades, enfatizando o peso das conexões internas em cada comunidade. Neste trabalho, utiliza-se a mesma matriz $\mathbf{A}$ do Esquema S para definir o peso de cada aresta.

A modularidade, denotada por Q , em sua forma padrão, é definida por:

$$Q = \frac{1}{2w} \sum_{i,j} \left(A_{ij} - \frac{s_i s_j}{2w} \right) \delta(c_i, c_j) \quad (3.17)$$

em que $\mathbf{A}(i,j)$ representa o peso da aresta entre os nós i e j ; s_i e s_j correspondem à força dos nós i e j , definida como a soma dos pesos das arestas incidentes; $w = \frac{1}{2} \sum_{i,j} A(i,j)$ é o peso total das arestas da rede; c_i e c_j indicam as comunidades às quais pertencem os nós i e j ; e $\delta(c_i, c_j)$ é a função delta de *Kronecker*, que assume valor 1 quando $c_i = c_j$ e 0 caso contrário.

O algoritmo inicia considerando cada nó como uma comunidade individual e, de forma gulosa, realiza a união de pares de comunidades com o objetivo de maximizar o valor de Q . Esse processo é repetido até que não sejam mais possíveis uniões que aumentem a modularidade, resultando em uma estrutura de comunidades localmente ótima. O procedimento também pode ser interrompido ao se atingir um número máximo de partições previamente definido.

Neste trabalho, utiliza-se a função de modularidade gulosa disponível na biblioteca de código aberto NetworkX. O código-fonte pode ser acessado em: <https://github.com/networkx/networkx>. Esse módulo recebe como entrada a matriz $\mathbf{A}$ e o número máximo de partições, realizando o particionamento com base na conectividade topológica dos vértices e no grau de afinidade entre eles.

No exemplo considerado, a aplicação do método de modularidade gulosa resultou no mesmo particionamento balanceado apresentado na Figura 16.

3.4.4 K-means

O algoritmo *k-means* é um método de particionamento amplamente utilizado, conhecido por sua simplicidade conceitual e facilidade de implementação (exigindo poucos parâme-

tros de ajuste, principalmente o número de clusters), além de apresentar elevada eficiência computacional e bom desempenho no processamento de grandes conjuntos de dados numéricos. Suas etapas principais são descritas a seguir:

1. **Inicialização:** seleção de k centroides iniciais. Esses centroides podem ser escolhidos aleatoriamente a partir dos dados ou por meio de métodos mais elaborados, como o *k-means++*, visando melhorar a convergência.
2. **Atribuição:** cada ponto de dados é associado ao centroide mais próximo, geralmente utilizando a distância Euclidiana. Esse processo resulta na formação de k clusters, nos quais cada ponto pertence ao cluster cujo centroide é o mais próximo.
3. **Atualização:** os centroides de cada cluster são recalculados como a média dos pontos de dados a ele atribuídos.
4. **Iteração:** as etapas de atribuição e atualização são repetidas até que não haja mais mudanças significativas nos centroides, indicando convergência. O algoritmo tende a encontrar um mínimo local da função objetivo, definida como a soma dos quadrados das distâncias entre os pontos de dados e seus respectivos centroides. Entretanto, não há garantia de obter o ótimo global.
5. **Parada:** o algoritmo é encerrado quando os centroides se estabilizam ou quando é atingido o número máximo de iterações previamente definido.

Para a aplicação do *k-means* no contexto da CA, define-se uma matriz de adjacência de medidas $\mathbf{M}(m \times m)$. Cada elemento $M(i, j)$ representa a distância entre as medidas i e j , definida como o número mínimo de ramos que as separam na rede. Os elementos da diagonal principal satisfazem $M(i, i) = 0$, uma vez que correspondem à mesma medição.

Para ilustrar a métrica de distância utilizada na aplicação do *k-means*, a Fig. 3 apresenta a matriz $\mathbf{M}$ associada ao sistema no exemplo do sistema de seis barras. Esta matriz é simétrica e seus elementos correspondem às distâncias entre as medidas. Por exemplo, a distância entre as medidas P_{1-3} e P_{4-5} é igual a 2, pois estão separadas por dois ramos na rede. De maneira similar, a distância entre as medidas P_{1-3} e P_{2-1} é igual a 1, pois estão em ramos vizinhos. Por outro lado, a distância entre as medidas P_{1-3} e P_1 é igual a 0, pois ambas estão instaladas na mesma barra.

Neste trabalho, utiliza-se a biblioteca *Scikit-learn* em Python (Pedregosa *et al.*, 2018) para a implementação do algoritmo *k-means*. São adotadas a configuração padrão de

	P_{1-3}	P_{2-1}	P_{2-3}	P_{4-5}	P_{4-6}	P_{5-1}	P_2	P_4	P_5
P_{1-3}	0	1	1	2	2	1	1	2	1
P_{2-1}		0	0	2	2	2	0	2	2
P_{2-3}			0	2	2	2	0	2	2
P_{4-5}				0	0	1	2	0	1
P_{4-6}					0	1	2	0	1
P_{5-1}						0	2	1	0
P_2							0	2	2
P_4								0	1
P_5									0

Tabela 3: Matriz de adjacências de medidas $\mathbf{M}$ do sistema exemplo de 6 barras.

inicialização, na qual os centroides iniciais são definidos aleatoriamente pela biblioteca. As medidas são então agrupadas com base na proximidade entre si na rede. Quando esse procedimento é aplicado à topologia do sistema em estudo, obtém-se um particionamento idêntico àquele produzido pelos Esquema H, S e M.

3.5 Análise de Criticalidades Particionada

Após o particionamento do sistema de medição, a AC passa a ser executada independentemente em cada grupo formado. Essa estratégia permite substituir uma única análise de grande porte por diversas análises menores, preservando a possibilidade de processamento em paralelo. Para isso, a metodologia proposta utiliza a matriz de covariâncias Ω , da qual são extraídas submatrizes correspondentes a cada partição(grupo) G_i . Essas submatrizes podem ser processadas paralelamente com o objetivo de detectar singularidades, e consequentemente, criticalidades presentes em cada grupo.

3.5.1 Metodologia para AC Particionada

Um aspecto fundamental da metodologia proposta é o tratamento das medidas localizadas nas fronteiras entre partições. Como essas medidas influenciam simultaneamente diferentes grupos, elas são compartilhadas entre as partições adjacentes, reduzindo a probabilidade de perda de criticalidades durante a análise particionada.

Sendo assim, as medidas de fluxo de potência associadas aos ramos de fronteira devem ser incluídas em ambos os grupos adjacentes. De forma análoga, as medidas de injeção nas barras de fronteira são incorporadas às análises dos grupos vizinhos correspondentes.

A Figura 18 apresenta um diagrama simplificado da abordagem proposta de AC particionada. O método recebe como entradas: o SM, o número desejado de partições n_g e as cardinalidades das tuplas de interesse. A partir dessas informações, o grafo do SM é construído, a matriz Ω é calculada e o grafo é particionado em $G_1, G_2, \dots, G_{n_g}$, utilizando um dos esquemas de particionamento adotados (H, S, K ou M).

Para cada partição, a matriz Ω é decomposta nas submatrizes correspondentes $\Omega_1, \Omega_2, \dots, \Omega_{n_g}$. A seguir, a AC é realizada em paralelo para cada submatriz. Por fim, as criticalidades do SM são todas aquelas obtidas pelos resultados das análises particionadas.

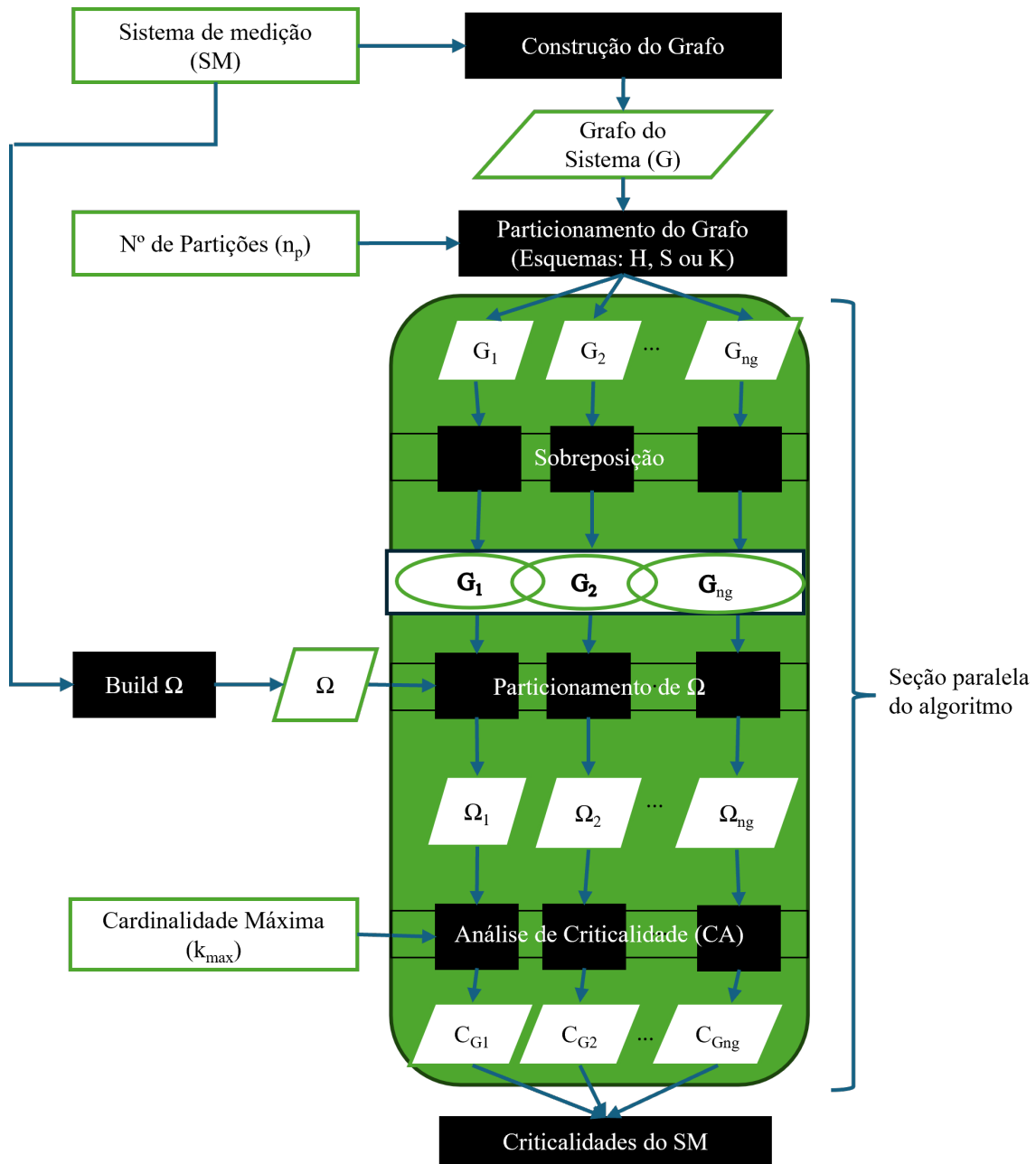


Figura 18: Fluxograma da abordagem proposta da AC particionada.

3.5.2 Exemplo ilustrativo

Para ilustrar o funcionamento da metodologia proposta, considera-se novamente o sistema exemplo de seis barras apresentado na Figura 10. Inicialmente, são apresentadas as criticalidades obtidas pela AC convencional. Em seguida, os resultados são comparados aos obtidos pela AC particionada.

$$G_1 = \{P_{1-3}, P_{2-1}, P_{2-3}, P_2, P_4, P_{5-1}, P_5\}$$

$$G_2 = \{P_{4-5}, P_{4-6}, P_4, P_{5-1}, P_5\}$$

A Figura 19 apresenta a representação dessas partições.

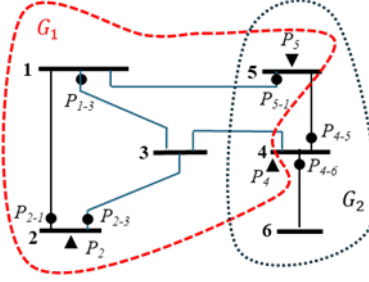


Figura 19: Grupos de medidas formados pelo particionamento do sistema exemplo de 6 barras.

	P1-3	P2	P2-1	P2-3	P5	P5-1	P4	P4-5	P4-6
P1-3	Ω_1								
P2									
P2-1									
P2-3									
P5									
P5-1									
P4									
P4-5								Ω_2	
P4-6									

Tabela 4: Representação das regiões Ω_1 e Ω_2 .

As criticalidades identificadas pela AC, sem o particionamento do SM, são descritas

a seguir. Esses resultados serão utilizados como referência para avaliar o impacto do particionamento sobre a identificação das criticalidades.

- Não há criticalidades do tipo C_1 ;
- Uma criticalidades do tipo C_2 :

$$(P_{4-6}, P_4)$$

- Oito criticalidades do tipo C_3 :

$$\begin{aligned} &(P_{5-1}, P_4, P_5), (P_{4-6}, P_{5-1}, P_5), (P_{4-5}, P_4, P_5), \\ &(P_{4-5}, P_{5-1}, P_5), (P_{4-5}, P_{5-1}, P_4), (P_{4-5}, P_{4-6}, P_5), \\ &(P_{4-5}, P_{4-6}, P_{5-1}), (P_{2-1}, P_{2-3}, P_2); \end{aligned}$$

- Seis criticalidades do tipo C_4 :

$$\begin{aligned} &(P_{1-3}, P_{2-3}, P_2, P_4), (P_{1-3}, P_{2-3}, P_{4-6}, P_2), \\ &(P_{1-3}, P_{2-1}, P_2, P_4), (P_{1-3}, P_{2-1}, P_{4-6}, P_2), \\ &(P_{1-3}, P_{2-1}, P_{2-3}, P_4), (P_{1-3}, P_{2-1}, P_{2-3}, P_{4-6}). \end{aligned}$$

Ao processar em paralelo as submatrizes Ω_1 (associada às medidas de G_1) e Ω_2 (contendo as medidas de G_2), três criticalidades do tipo C_4 — $(P_{1-3}, P_{2-3}, P_{4-6}, P_2)$, $(P_{1-3}, P_{2-1}, P_{4-6}, P_2)$ e $(P_{1-3}, P_{2-1}, P_{2-3}, P_{4-6})$ — não são identificadas.

Esse comportamento ocorre porque algumas medidas pertencentes a essas criticalidades estão distribuídas em grupos distintos, sendo, portanto, analisadas somente separadamente. Entretanto, observa-se que a tripla crítica (P_{5-1}, P_4, P_5) , formada na fronteira entre os grupos, é corretamente identificada.

O impacto da não identificação dessas criticalidades do tipo C_4 tende a ser limitado. Especificamente, os resultados da EE poderiam ser comprometidos apenas no caso de ocorrência simultânea de erros grosseiros nas medidas associadas aos nós 1, 2 e 4 — um evento que pode ser considerado pouco provável, tanto pela baixa probabilidade da ocorrência simultânea desses eventos, quanto pela distância física entre as barras. Nessa situação, tais erros permaneceriam não detectados (Clements; Davis, 1986).

Neste capítulo foi proposta uma metodologia de particionamento do SM baseada em grafos, composta por quatro estratégias de particionamento e uma etapa de refinamento

das criticalidades. No capítulo seguinte, essa metodologia será avaliada experimentalmente, considerando diferentes topologias de sistemas elétricos de potência e comparando os resultados obtidos com a abordagem particionada à AC realizada aplicada ao sistema completo.

4 Resultados Numéricos

O Capítulo 3 apresentou a metodologia proposta para reduzir o custo computacional da Análise de Criticalidade (AC) por meio do particionamento do sistema de medição. Neste capítulo, essa metodologia é avaliada experimentalmente quanto à capacidade de preservar a identificação das criticalidades e de reduzir o esforço computacional da análise.

4.1 Descrição da Simulação

O objetivo deste capítulo é avaliar o impacto do particionamento do Sistema de Medições (SM) sobre o desempenho da AC, todos os experimentos utilizam a implementação paralela em GPU desenvolvida no Capítulo 2 ([Nishio; Do Coutto Filho; Stachinni de Souza et al., 2024](#)), tendo sido construídas para comparar diretamente a AC convencional (efetuada com o sistema completo) com a AC particionada por meio de cada um dos quatro esquemas de particionamento propostos no Capítulo 3. No procedimento de identificação de criticalidades implementado em GPU, considera-se a latência entre arquiteturas para que a avaliação do desempenho computacional reflita eventuais atrasos de comunicação ou de processamento introduzidos pelo hardware.

São adotados os sistemas-teste IEEE de 118 barras ([Nishio; Do Coutto Filho; Stachinni de Souza et al., 2024](#)) e o sistema Polonês de 2.383 barras ([Snodgrass et al., 2022; Nishio; Do Coutto Filho; Stacchini De Souza et al., 2024](#)). As simulações são realizadas em um computador equipado com 32 GB de memória RAM, processador Intel Core i5-9300H e GPU GTX 1650. A biblioteca *NetworkX*, em *Python*, é utilizada nas operações relacionadas a grafos. Todos os algoritmos foram executados utilizando a mesma plataforma computacional, permitindo que as comparações reflitam exclusivamente diferenças entre as metodologias de particionamento.

A avaliação experimental foi organizada em cinco etapas. Inicialmente, investiga-se a distribuição espacial das criticalidades. Em seguida, avalia-se a taxa de identificação obtida pelos diferentes métodos de particionamento. Posteriormente, apresenta-se um

exemplo representativo de criticalidades não identificadas. Na sequência, compara-se o desempenho computacional das abordagens e, por fim, valida-se a metodologia no sistema polonês de 2.383 barras.

4.2 Avaliação dos Métodos de Particionamento para AC

Um método de particionamento pode ser avaliado tanto pela qualidade da solução produzida quanto pelo tempo computacional necessário para obtê-la. Neste trabalho, são analisados os seguintes aspectos:

- capacidade de identificação de C_{ks} ;
- número de combinações m_{comb} avaliadas pela AC;
- número de partições (n_p) e suas interações;
- distribuição do esforço computacional entre os grupos;
- tempo de execução da AC.

Quanto à capacidade de identificação de C_{ks} , calcula-se a seguinte métrica, denominada taxa de identificação de C_{ks} , representada por RoI_k (em percentual):

$$RoI_k = \frac{n_{C_k}^p}{n_{C_k}} \times 100 \quad (4.1)$$

em que $n_{C_k}^p$ representa o número de C_{ks} identificados pela AC particionada, e n_{C_k} o número de C_{ks} identificados pela AC não particionada.

A escolha de n_p para o grafo associado ao SM depende de fatores como a estrutura do grafo, a aplicação considerada e os recursos computacionais disponíveis. Grafos de grande porte podem requerer um número maior de partições para reduzir o custo computacional por partição. Em ambientes de computação paralela, deseja-se um número de partições igual ou múltiplo do número de processadores ou núcleos disponíveis.

Neste trabalho, com base em diversas simulações para ajuste de n_p , considerando principalmente o tamanho do SM, o balanceamento de carga computacional e o tempo de execução, adota-se, como prova de conceito, 3, 7 e 10 partições para o sistema IEEE de 118 barras, utilizando os esquemas H, K e S. Para o sistema Polonês de 2.383 barras, são adotadas 15 partições. O Esquema M utiliza um limite superior para n_p como entrada.

4.3 Simulação do Sistema IEEE de 118 Barras

Com o objetivo de garantir uma avaliação imparcial dos esquemas de particionamento adotados para a AC, são geradas 100 configurações distintas de sistemas de medição para monitorar a rede IEEE de 118 barras, todas com um mesmo nível de redundância.

Em cada configuração, medidas de fluxo de potência em ramos e de injeção de potência em barras são alocadas aleatoriamente até atingir um nível de redundância de 60%, o que, em média, resulta em 333 medidas por configuração. Ressalta-se que os esquemas de particionamento adotados permanecem aplicáveis a sistemas de medição convencionais (SCADA) e aos que contêm PMUs ([Ahmed *et al.*, 2022](#)).

4.3.1 Adjacência de medidas participantes de criticalidades

As medidas que integram uma C_k tendem a estar associadas a barras adjacentes na rede elétrica. Para isso, define-se $d_{\max}$ como a distância máxima (ou diâmetro) da C_k , correspondente ao número máximo de arestas (linhas) no caminho mínimo entre quaisquer duas barras que possuam medidas do grupo crítico. Uma distância nula indica que todos os elementos da C_k pertencem à mesma barra."

A Tabela 5 detalha a distribuição das criticalidades em função da distância máxima, $d_{\max} \in \{0, \dots, 6\}$. Nesta representação, n_{C_k} denota o número de C_{ks} cuja dispersão espacial na rede corresponde a uma distância $d_{\max}$.

Tabela 5: Distribuição das criticalidades identificadas em função da distância máxima.

$d_{\max}$	n_{C_2}	n_{C_3}	n_{C_4}
0	106	21	3
1	238	763	855
2	31	335	1213
3	0	18	120
4	0	5	21
5	0	0	1
6	0	0	1
Total	375	1142	2214

Observa-se que as tuplas críticas mais relevantes, isto é, as de menor ordem, se concentram predominantemente em distâncias topológicas curtas. Por exemplo, aproxima-

mente 92% das C_2 ocorrem para distâncias iguais a 0 e 1, evidenciando um comportamento essencialmente local. De forma análoga, verifica-se tendência semelhante para as C_3 , das quais cerca de 98% apresentam $d_{\max}$ entre 0 e 3, e para as C_4 , em que aproximadamente 99% possuem $d_{\max}$ entre 0 e 4. Esses resultados fornecem evidências quantitativas de que a maioria das criticalidades está concentrada em regiões topologicamente próximas, corroborando a hipótese de localidade que fundamenta a metodologia proposta.

4.3.2 Taxa Média de Identificação de Criticalidades

A taxa de acerto média $\overline{RoI}_k$ é calculada a partir da equação 4.1 dos 100 sistemas de medição simulados. A Tabela 6 apresenta os valores de $\overline{RoI}_k$ ($k \leq 4$), juntamente com seus respectivos desvios-padrão σ_k , para os esquemas avaliados (H, S, K, M). Adicionalmente, a Tabela 6 apresenta o número médio de C_{ks} identificadas pela AC não particionada, representado por $\overline{n_{C_k}}$, para cada ordem k . A Tabela 6 é organizada em três blocos, correspondentes aos números de partições $n_p = 3$, $n_p = 7$ e $n_p = 10$.

Tabela 6: Taxa de Identificação de criticalidades (%) — IEEE 118 Barras

$n_p = 3$		$\overline{RoI}_k \pm \sigma_k$ (%)			
k	$\overline{n_{C_k}}$	H	S	K	M
1	1	100 $\pm$ 0,0	100 $\pm$ 0,0	100 $\pm$ 0,0	100 $\pm$ 0,0
2	4	100 $\pm$ 0,0	100 $\pm$ 0,0	100 $\pm$ 0,0	100 $\pm$ 0,0
3	12	99,2 $\pm$ 3,7	99,2 $\pm$ 4,1	99,8 $\pm$ 1,3	99,6 $\pm$ 1,9
4	23	98,9 $\pm$ 2,5	99,2 $\pm$ 2,4	99 $\pm$ 2,3	99,6 $\pm$ 1,4

$n_p = 7$		$\overline{RoI}_k \pm \sigma_k$ (%)			
k	$\overline{n_{C_k}}$	H	S	K	M
1	1	100 $\pm$ 0,0	100 $\pm$ 0,0	100 $\pm$ 0,0	100 $\pm$ 0,0
2	4	100 $\pm$ 0,0	100 $\pm$ 0,0	100 $\pm$ 0,0	100 $\pm$ 0,0
3	12	98,6 $\pm$ 3,7	99,2 $\pm$ 4,1	99,8 $\pm$ 1,3	99,6 $\pm$ 1,9
4	23	96,9 $\pm$ 5,1	98,4 $\pm$ 4,9	97,5 $\pm$ 4,0	98 $\pm$ 4,8

$n_p = 10$		$\overline{RoI}_k \pm \sigma_k$ (%)			
k	$\overline{n_{C_k}}$	H	S	K	M
1	1	100 $\pm$ 0,0	100 $\pm$ 0,0	100 $\pm$ 0,0	100 $\pm$ 0,0
2	4	100 $\pm$ 0,0	100 $\pm$ 0,0	100 $\pm$ 0,0	100 $\pm$ 0,0
3	12	98,1 $\pm$ 4,4	99,1 $\pm$ 3,1	98,5 $\pm$ 4,9	99,3 $\pm$ 2,6
4	23	94,5 $\pm$ 6,9	97,4 $\pm$ 4,2	96,2 $\pm$ 5,9	98,7 $\pm$ 4,8

Todos os esquemas identificam 100% dos casos de C_1 e apresentam elevados valores de $\overline{RoI}_k$ (superiores a 96%), com baixa dispersão (entre 1,3 e 6,9). Esses resultados demonstram a viabilidade da AC particionada, uma vez que são obtidos resultados aproximados

de alta qualidade, independentemente do método de particionamento adotado.

Para criticalidades de menor ordem, observam-se valores ligeiramente superiores de $\overline{RoI}_k$. Isso ocorre porque criticalidades de maior cardinalidade tendem a envolver medidas espacialmente mais dispersas entre diferentes partições, aumentando a probabilidade de não identificação de determinadas C_k .

4.3.3 Exemplo de Criticalidade não identificada

Considerando a natureza local da ocorrência de criticalidades, é razoável supor que as C_{ks} não identificadas tendam a se concentrar em regiões de fronteira entre partições, o que evidencia a relevância da sobreposição de medidas nessas áreas. Com base nessa hipótese, a Figura 20 ilustra esse comportamento.

Considere, sem perda de generalidade, a interseção entre dois grupos, G_1 e G_2 , definidos pelo esquema S quando aplicado a uma das cem configurações simuladas do sistema IEEE de 118 barras. De acordo com a estratégia de particionamento adotada neste trabalho, as interseções entre grupos são determinadas com base em medidas de fluxo e de injeção. Ao executar a AC sem particionamento do SM, identifica-se a quádrupla $(P_{34}, P_{34-43}, P_{43}, P_{44-45})$ como uma criticalidade do tipo C_4 . Entretanto, essa criticalidade não é identificada pela AC particionada.

Observa-se que a medida P_{44-45} pertence ao grupo G_2 , enquanto as demais medidas que compõem o C_4 pertencem ao grupo G_1 . Como a AC particionada é executada separadamente em G_1 e G_2 , tais medidas não são analisadas simultaneamente, resultando na perda de identificação dessa criticalidade.

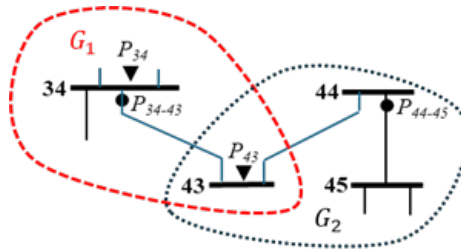


Figura 20: Exemplo de C_4 não identificada no sistema IEEE 118 barras.

Adicionalmente, para evidenciar a importância da forma de interação entre as partições do SM, considera-se a situação em que a etapa de sobreposição das medidas de fronteira é desconsiderada. Observa-se que todas as taxas de identificação se tornam significativamente inferiores às aquelas apresentadas na Tabela 6, especialmente para um maior

número de grupos. Por exemplo, para 10 grupos formados pelos esquemas H , K e S , os valores de RoI_4 , expressos por $\mu_4(\sigma_4)$, são, respectivamente, $70,0 \pm 12,5$, $78,5 \pm 11,6$ e $53,3 \pm 12,5$.

4.3.4 Avaliação de Desempenho

Esta subseção apresenta a análise do desempenho computacional da AC particionada. Os esquemas considerados foram aplicados às 100 simulações de SM, cujos resultados são apresentados na Tabela 7.

A terceira coluna da Tabela 7 apresenta o número médio de medidas no maior grupo ($\overline{m}_{\max}$) de cada esquema, considerando a divisão do SM em 3, 7 e 10 partições. Observa-se que o esquema S tende a produzir grupos com menor número de medidas, característica favorável à execução eficiente da AC particionada em ambiente de programação paralela, uma vez que o tempo total do processamento simultâneo dos grupos é influenciado, predominantemente, pelo tempo de execução do maior grupo.

A quarta coluna da Tabela 7 apresenta o tempo médio de execução ($\bar{t}$) da AC particionada, paralelizada e otimizada em GPU. Esse valor corresponde, essencialmente, ao tempo de processamento da AC no maior grupo de medidas gerado pelo respectivo esquema de particionamento. Para referência, o tempo médio de execução da AC aplicada ao sistema completo, sem particionamento, é de aproximadamente 8,86 s. A quinta coluna apresenta o percentual desse tempo requerido pela abordagem particionada. Os resultados apresentados na quarta e quinta colunas da Tabela 7, em conjunto com a métrica RoI_k , evidenciam a vantagem computacional do particionamento.

A expressiva redução do espaço de busca, entendido como o número de combinações avaliadas, constitui uma das principais vantagens do particionamento. Nesse contexto, a sexta coluna apresenta o valor médio de $\overline{m}_{\text{comb}}$ analisado por cada esquema. Tais valores são, no mínimo, duas ordens de grandeza inferiores aos requeridos pela AC sem particionamento do SM, caso em que uma média de $5,09 \times 10^8$ combinações deve ser avaliada. Por fim, a última coluna apresenta o percentual do espaço de busca explorado pela AC particionada em relação ao correspondente caso sem particionamento.

Os resultados mostram que o aumento de n_p reduz o número de combinações avaliadas e, conseqüentemente, o tempo de execução. Entretanto, esse ganho computacional ocorre em detrimento da capacidade de identificação de criticalidades, uma vez que parte delas pode ocorrer em regiões de fronteira entre partições.

Tabela 7: Desempenho dos Esquemas de Particionamento — Sistema de Medição IEEE 118 Barras (C_k com $k \leq 4$)

n_p	Esquema	Maior grupo	Tempo (s)	(%)	m_{comb}	Esp. Busca (%)
3	H	138	0,304	3,4	$2,92 \times 10^7$	5,4
3	S	135	0,285	3,2	$2,78 \times 10^7$	5,1
3	K	156	0,444	5,0	$3,77 \times 10^7$	7,0
3	M	146	0,400	4,5	$3,20 \times 10^7$	5,9
7	H	91	0,149	1,7	$6,63 \times 10^6$	1,2
7	S	77	0,134	1,5	$3,99 \times 10^6$	0,7
7	K	88	0,142	1,6	$6,07 \times 10^6$	1,1
7	M	83	0,130	1,5	$4,53 \times 10^6$	0,8
10	H	73	0,116	1,3	$2,83 \times 10^6$	0,5
10	S	64	0,114	1,3	$2,08 \times 10^6$	0,4
10	K	72	0,110	1,2	$3,29 \times 10^6$	0,6
10	M	72	0,117	1,3	$3,03 \times 10^6$	0,6

Dessa forma, n_p tende a ser escolhido tão pequeno quanto possível, porém buscando preservar a capacidade de identificação de criticalidades. A escolha final de n_p deve, portanto, equilibrar precisão analítica e custo computacional, considerando também os recursos de processamento disponíveis e eventuais restrições de tempo para a análise.

4.4 Simulação do Sistema Polonês de 2.383 Barras

O sistema polonês é uma das maiores redes empregadas na literatura em estudos conceituais de estimação de estado (EE) (Ahmed *et al.*, 2022). O SM adotado nas simulações é composto por 5.846 medidas, distribuídas de modo a garantir a observabilidade. Esse conjunto inclui 4.241 medidas de fluxo de potência em ramos e 1.605 medidas de injeção de potência em barras.

Com base em diversas simulações, a divisão do SM em 15 partições mostrou-se uma escolha adequada. Assim como na Seção 4.3, emprega-se computação paralela baseada em GPU para a execução da AC tanto no SM particionado quanto no não particionado.

A execução da AC não particionada para identificação de C_k de ordens mais elevadas em sistemas de grande porte torna-se intratável, uma vez que o crescimento combinatório de m_{comb} resulta em tempos computacionais proibitivos. Por exemplo, sem particionamento do sistema Polonês, a identificação de C_k para $k \leq 4$ requer a avaliação de $1,34 \times 10^{12}$ combinações de medidas. Nessa condição, mesmo com processamento paralelo em GPU, a

AC demandaria dias para ser concluída, o que configura um tempo de execução inaceitável e característico da intratabilidade do problema.

Isso ocorre porque, para $k = 4$, o valor de m_{comb} cresce drasticamente e, consequentemente, também o tempo de execução da AC. Dessa forma, o índice RoI_4 e o tempo relativo entre a AC particionada e não particionada não podem ser computados. Em contraste, para $k \leq 3$, a AC torna-se viável na rede polonesa. Sem particionamento, são identificados 745 casos críticos em 29 minutos. Com particionamento, a taxa de identificação de criticalidades permanece muito elevada, enquanto os tempos de execução para os esquemas H, S, K e M são, respectivamente, 0,61 s, 1,01 s, 1,73 s e 0,55 s.

A Tabela 8 apresenta o desempenho da AC particionada, para $k \leq 4$, no sistema de medição Polonês.

Tabela 8: Desempenho dos Esquemas de Particionamento — Sistema de Medição Polonês de 2.383 Barras (C_k com $k \leq 4$)

$n_p = 15$		Esquemas			
Esquemas		H	S	K	M
Número de C_{ks} identificadas	C_1	12	12	12	12
	C_2	173	173	173	173
	C_3	558	559	544	557
Taxa de identificação	$RoI_1(\%)$	100	100	100	100
	$RoI_2(\%)$	100	100	100	100
	$RoI_3(\%)$	99.6	99.8	97.1	99.5
	$RoI_4(\%)$	—	—	—	—
Métricas de desempenho	Maior grupo	628	721	860	578
	Tempo	2.3 min	4.7 min	9.9 min	1.7 min
	m_{comb}	3.9×10^{10}	5.8×10^{10}	1.2×10^{11}	2.6×10^{10}
	Espaço(%)	3%	4%	9%	2%

De forma análoga aos resultados apresentados na Seção 4.3.2, observa-se uma redução de duas ordens de grandeza no espaço de busca das combinações de medidas. Por exemplo, ao utilizar o Esquema M, são identificados 1.600 casos críticos em apenas 1,7 min, explorando cerca de 2% do espaço total de busca.

Com base nos resultados obtidos até o momento, espera-se uma elevada taxa de identificação de C_{4s} no sistema de medição polonês. Ressalta-se que mesmo uma lista parcial de C_4 — ainda que não contemple todas as criticalidades de ordem superior — permanece altamente relevante para análises diagnósticas voltadas à identificação de fragilidades do

sistema de medição.

4.5 Considerações Finais

Os resultados experimentais apresentados neste capítulo permitiram avaliar a metodologia proposta sob diferentes perspectivas, incluindo a distribuição espacial das criticalidades, a capacidade de identificação dos diferentes métodos de particionamento e o desempenho computacional obtido.

Inicialmente, verificou-se que as criticalidades apresentam comportamento predominantemente local, sendo seus elementos geralmente concentrados em regiões topologicamente próximas da rede elétrica. Esse resultado fornece embasamento experimental à hipótese adotada nesta tese e justifica a utilização do particionamento do SM como estratégia para reduzir a complexidade da AC.

A comparação entre os diferentes métodos de particionamento demonstrou que todos alcançaram elevadas taxas de identificação de criticalidades. Entretanto, o algoritmo heurístico desenvolvido nesta tese apresentou, de forma consistente, o melhor compromisso entre a qualidade do particionamento e o custo computacional, evidenciando a importância de considerar simultaneamente critérios de conectividade e de balanceamento das partições.

As simulações também mostraram que as poucas criticalidades não identificadas estão, em sua maioria, associadas às regiões de fronteira entre partições. Esse comportamento confirma que a principal limitação do particionamento decorre da separação de elementos pertencentes à mesma criticalidade, justificando a adoção do mecanismo de compartilhamento de medidas em regiões de fronteira dos grupos.

Finalmente, os resultados de desempenho evidenciaram uma redução significativa no número de combinações avaliadas e, conseqüentemente, do tempo de execução da AC, permitindo sua aplicação em sistemas de medição de grande porte. A validação realizada no sistema IEEE de 118 barras e, principalmente, no sistema polonês de 2.383 barras demonstra que a metodologia proposta amplia de forma significativa os limites práticos da AC, preservando elevada qualidade dos resultados e reduzindo substancialmente seu custo computacional.

5 Conclusões e Trabalhos Futuros

Esta tese abordou o problema da identificação de criticalidades em planos de medição destinados à Estimação de Estado (EE) em sistemas elétricos de potência. A relevância desse problema decorre do fato de que a presença de criticalidades indica o grau de observabilidade da rede, o que leva à redução da confiabilidade dos resultados obtidos pela EE, caracterizada pela capacidade de detecção e processamento de medidas espúrias. Apesar de sua importância prática, a Análise de Criticalidade (AC) requer a avaliação de um número extremamente elevado de combinações de medidas, o que pode limitar sua aplicação em sistemas de grande porte.

Com o objetivo de ampliar os limites de aplicação da AC, esta tese investigou duas estratégias complementares para reduzir o custo computacional da análise. A primeira consistiu no aprimoramento da implementação em GPU, explorando de forma mais eficiente o paralelismo inerente ao problema. Para isso, todas as etapas remanescentes da AC foram implementadas em GPU, reduzindo significativamente o tempo de processamento e permitindo a análise de sistemas de maior porte. Os resultados obtidos demonstraram que a implementação proposta supera, em desempenho, a implementação sequencial do método *Branch and Bound* descrita na literatura.

Entretanto, embora as otimizações alcançadas em GPUs reduzam significativamente o tempo de execução, essas otimizações não alteram a natureza combinatória do problema. Assim, a principal contribuição desta tese consistiu na proposição de uma nova abordagem baseada no particionamento do sistema de medição, explorando a natureza local das criticalidades. Essa estratégia permite dividir o problema original em instâncias menores, reduzindo substancialmente o número de combinações avaliadas em cada etapa da análise.

A fundamentação teórica desenvolvida ao longo da tese permitiu identificar os principais fatores que influenciam o desempenho da AC. Observou-se que o número de medidas do plano constitui o principal fator associado à complexidade computacional, enquanto a topologia da rede afeta diretamente a distribuição espacial das criticalidades. Nesse

contexto, a matriz de covariância Ω mostrou-se uma representação adequada para integrar simultaneamente as características topológicas da rede e a distribuição das medidas, tornando-se um elemento central para o desenvolvimento da estratégia de particionamento.

Com base nessa representação, foi proposta uma modelagem do sistema de medição como um grafo ponderado, o que possibilita a aplicação de técnicas de particionamento de grafos. Verificou-se que o balanceamento do número de medidas entre os grupos é um requisito essencial para maximizar a eficiência da análise particionada. Para esse fim, foi desenvolvida uma função objetivo específica para quantificar a qualidade do particionamento, bem como um procedimento heurístico capaz de produzir soluções eficientes para sistemas de grande porte. Adicionalmente, foram avaliados diferentes métodos de particionamento, incluindo o particionamento espectral, o método k -means e o algoritmo *Greedy Modularity*.

Após o estabelecimento das partições, a matriz Ω foi decomposta em submatrizes correspondentes aos grupos obtidos. Além disso, foi introduzida uma estratégia de sobreposição de medidas entre partições vizinhas, considerando medidas de fluxo e de injeção localizadas nas regiões de fronteira. Essa estratégia aumentou a capacidade de identificação de criticalidades distribuídas entre diferentes grupos, preservando elevadas taxas de identificação sem comprometer os ganhos computacionais obtidos pelo particionamento.

A metodologia proposta foi validada em sistemas de medição de diferentes portes, incluindo múltiplas instâncias do sistema IEEE de 118 barras e o sistema polonês, representativo de aplicações em larga escala. Os resultados confirmaram a hipótese da natureza local das criticalidades, evidenciando que a maioria das medidas que formam grupos críticos encontra-se espacialmente concentrada. Além disso, todos os métodos de particionamento avaliados produziram resultados satisfatórios, alcançando o particionamento ótimo nos sistemas de menor porte e reduzindo significativamente o número de combinações analisadas. Como consequência, foram obtidas elevadas taxas de identificação de criticalidades, baixo desvio padrão entre diferentes instâncias e reduções expressivas no tempo computacional.

5.1 Principais Contribuições

Em síntese, esta tese contribui para o avanço da AC em dois aspectos complementares. O primeiro consiste na evolução da implementação paralela em GPU, proporcionando

ganhos de desempenho computacional em relação à abordagem tradicional. O segundo, e principal, corresponde à proposição de uma metodologia inédita de particionamento topológico do sistema de medição, que explora a natureza local das criticalidades para reduzir o espaço combinatório e ampliar a escalabilidade da AC. Essa abordagem torna viável a análise de sistemas substancialmente maiores do que aqueles tratados pelas estratégias convencionais, permitindo a identificação coordenada de medidas críticas individuais, pares críticos, triplas críticas, quádruplas críticas e combinações de ordem superior, sempre que computacionalmente viável. Além disso, os resultados da AC podem subsidiar análises complementares das fragilidades do sistema de medição, como a identificação de UMs críticas e a avaliação da criticidade de linhas de comunicação associadas ao processo de medição.

5.2 Trabalhos Futuros

Os resultados desta tese demonstram o potencial da metodologia proposta para ampliar a aplicabilidade da AC em sistemas de medição de grande porte. Entretanto, diferentes aspectos ainda podem ser explorados com o objetivo de aumentar sua eficiência computacional, aprimorar as taxas de identificação de criticalidades e ampliar sua escalabilidade. Nesse contexto, destacam-se as seguintes possibilidades para trabalhos futuros:

- **Execução distribuída em múltiplas GPUs.** A independência entre as partições torna natural a utilização de múltiplas GPUs para executar a AC de forma distribuída, reduzindo ainda mais o tempo de processamento. Além disso, características da arquitetura de interconexão entre GPUs podem ser exploradas para otimizar a comunicação entre partições quando o compartilhamento de informações for necessário.
- **Particionamento por programação inteira.** A obtenção de soluções ótimas para o problema de particionamento permanece um desafio. Este trabalho iniciou a investigação dessa abordagem por meio de uma formulação de programação inteira; porém, sua elevada complexidade inviabilizou sua aplicação em sistemas de grande porte. Como continuidade, pretende-se investigar novas formulações, técnicas de relaxação e métodos de decomposição que permitam obter soluções ótimas ou próximas do ótimo para redes maiores.
- **Novas estratégias de particionamento.** Outros métodos de particionamento de grafos podem ser investigados com o objetivo de produzir grupos mais equi-

librados e aumentar as taxas de identificação de criticalidades. Adicionalmente, técnicas de aprendizado de máquina podem ser empregadas para identificar padrões na distribuição espacial das criticalidades e orientar automaticamente o processo de particionamento.

- **Estratégias de sobreposição entre partições.** Os resultados obtidos mostraram que a sobreposição de medidas entre grupos vizinhos aumenta a capacidade de identificação de criticalidades localizadas nas regiões de fronteira. Assim, a investigação de novos critérios para seleção das medidas compartilhadas, bem como diferentes níveis de sobreposição, constitui uma direção promissora para elevar as taxas de identificação mantendo reduzido o custo computacional.
- **Avaliação em sistemas de maior porte.** A realização de ensaios em sistemas de medição ainda maiores e em redes com diferentes características topológicas permitirá avaliar os limites de escalabilidade da metodologia proposta, bem como identificar oportunidades adicionais de otimização.
- **Particionamento baseado na matriz de covariância.** A representação proposta utiliza a topologia da rede e a distribuição das medidas para construir o grafo do sistema. Como continuidade, podem ser investigadas estratégias de particionamento que utilizem diretamente informações contidas na matriz de covariância Ω , explorando o grau de dependência entre medidas para produzir grupos mais adequados à AC.
- **Particionamento adaptativo.** Neste trabalho, o número de partições e o grau de sobreposição entre os grupos são definidos previamente. Como continuidade, podem ser desenvolvidas estratégias adaptativas capazes de ajustar automaticamente esses parâmetros em função das características topológicas do sistema de medição, do número de medidas, buscando maximizar simultaneamente a taxa de identificação e o desempenho computacional.
- **Explorar outros tipo de UMs:** A metodologia proposta foi avaliada considerando apenas UMs associadas a medições de fluxo e injeção. Como trabalho futuro, pretende-se investigar a aplicação da metodologia a planos de medição que incorporem medições fasoriais provenientes de PMUs (*Phasor Measurement Units*), incluindo fasores de tensão e corrente. Além disso, pode-se avaliar a utilização de novas tecnologias de medição, como WMUs (*Waveform Measurement Units*) e XMUs (*Extended Measurement Units*), investigando como as características dessas medições

podem influenciar o particionamento do sistema e a identificação de criticalidades.

- **Explorar métodos alternativos para a AC:** A metodologia proposta baseia-se na enumeração combinatória e no método *Branch and Bound* para a identificação de criticalidades. Como trabalho futuro, pretende-se investigar abordagens alternativas para a realização da AC, incluindo técnicas de otimização, algoritmos genéticos e métodos de aprendizado de máquina, avaliando sua eficácia em termos de desempenho computacional.

REFERÊNCIAS

AHMED, Muhammad *et al.* A Critical Review of State-of-the-Art Optimal PMU Placement Techniques. **Energies**, v. 15, mar. 2022. DOI: [10.3390/en15062125](https://doi.org/10.3390/en15062125).

ALI ABUR, Antonio Gómez Expósito. **Power System State Estimation: Theory and Implementation**. New York, N.J.: Marcel Decker, 2004.

ALMEIDA, M. C. de; ASADA, E. N.; GARCIA, A. V. On the Use of Gram Matrix in Observability Analysis. **IEEE Transactions on Power Systems**, v. 23, n. 1, p. 249–251, 2008.

AUGUSTO, A. A.; FILHO., Do Coutto; STACCHINI DE SOUZA, Julio C.. de. Low-cardinality critical k-tuples in measurement sets for state estimation. *In*: 2013 IEEE Grenoble Conference. [S. l.: s. n.], jun. 2013. p. 1–6.

AUGUSTO, Andre Abel. **Avaliação da Capacidade de Observação do Estado Operativo de Redes Elétricas**. Fev. 2016. Tese (Doutorado) – Instituto de Computação, Universidade Federal Fluminense, Niterói, RJ, Brasil.

AUGUSTO, Andre Abel; DO COUTTO FILHO, Milton Brown *et al.* Branch-and-Bound Guided Search for Critical Elements in State Estimation. *en*. **IEEE Transactions on Power Systems**, v. 34, n. 3, p. 2292–2301, mai. 2019. ISSN 0885-8950, 1558-0679. DOI: [10.1109/TPWRS.2018.2881421](https://doi.org/10.1109/TPWRS.2018.2881421). Disponível em: <https://ieeexplore.ieee.org/document/8536444/>. Acesso em: 17 jul. 2020.

AYALL, Tewodros Alemu *et al.* Graph Computing Systems and Partitioning Techniques: A Survey. **IEEE Access**, v. 10, p. 118523–118550, 2022. DOI: [10.1109/ACCESS.2022.3219422](https://doi.org/10.1109/ACCESS.2022.3219422).

AYRES, M.; HALEY, P. H. Bad Data Groups in Power System State Estimation. **IEEE Transactions on Power Systems**, v. 1, n. 3, p. 1–7, 1986.

BRAGA FLOR, Vinícius *et al.* Strategic observation of power grids for reliable monitoring. **International Journal of Electrical Power & Energy Systems**, v. 138, p. 107959, jun. 2022. DOI: [10.1016/j.ijepes.2022.107959](https://doi.org/10.1016/j.ijepes.2022.107959).

BRAGA FLOR, Vinicius B. *et al.* Critical Data Visualization to Enhance Protection Schemes for State Estimation. **IEEE Transactions on Smart Grid**, v. 14, n. 2, p. 1249–1261, mar. 2023. Conference Name: IEEE Transactions on Smart Grid. ISSN 1949-3061. DOI: [10.1109/TSG.2022.3203404](https://doi.org/10.1109/TSG.2022.3203404).

BRAGA FLOR, Vinicius B. *et al.* Tactics for improving computational performance of criticality analysis in state estimation. **IET Generation, Transmission & Distribution**, v. 15, n. 7, p. 1214–1226, 2021. DOI: <https://doi.org/10.1049/gtd2.12097>. eprint: <https://ietresearch.onlinelibrary.wiley.com/doi/pdf/10.1049/gtd2.12097>. Disponível em: <https://ietresearch.onlinelibrary.wiley.com/doi/abs/10.1049/gtd2.12097>.

CASTILLO, E. *et al.* State estimation observability based on the null space of the measurement Jacobian matrix. **IEEE Transactions on Power Systems**, v. 20, n. 3, p. 1656–1658, 2005.

CHEN, R.-. A fast integer algorithm for observability analysis using network topology. **IEEE Transactions on Power Systems**, v. 5, n. 3, p. 1001–1009, 1990.

CLAUSET, Aaron; NEWMAN, M. E. J.; MOORE, Cristopher. Finding community structure in very large networks. **Physical Review E**, American Physical Society (APS), v. 70, n. 6, dez. 2004. ISSN 1550-2376. DOI: [10.1103/physreve.70.066111](https://doi.org/10.1103/physreve.70.066111). Disponível em: <http://dx.doi.org/10.1103/PhysRevE.70.066111>.

CLEMENTS, K.; KRUMPHOLZ, G.; DAVIS, P. Power System State Estimation Residual Analysis: An Algorithm Using Network Topology. **IEEE Transactions on Power Apparatus and Systems**, PAS-100, n. 4, p. 1779–1787, 1981.

CLEMENTS, K. A.; DAVIS, P. W. Multiple Bad Data Detectability and Identifiability: A Geometric Approach. **IEEE Transactions on Power Delivery**, v. 1, n. 3, p. 355–360, 1986.

CLEMENTS, K. A.; KRUMPHOLZ, G. R.; DAVIS, P. W. State Estimator Measurement System Reliability Evaluation – An Efficient Algorithm Based on Topological Observability Theory. **IEEE Transactions on Power Apparatus and Systems**, PAS-101, n. 4, p. 997–1004, 1982.

CLEMENTS, K. A.; KRUMPHOLZ, G. R.; DAVIS, P. W. Power System State Estimation with Measurement Deficiency: an Observability/Measurement Placement Algorithm. **IEEE Transactions on Power Apparatus and Systems**, PAS-102, n. 7, p. 2012–2020, 1983.

DING, H.; EL-KEIB, A.A.; SMITH, R. Optimal clustering of power networks using genetic algorithms. **Electric Power Systems Research**, v. 30, n. 3, p. 209–214, 1994.

ISSN 0378-7796. DOI: [https://doi.org/10.1016/0378-7796\(94\)00857-4](https://doi.org/10.1016/0378-7796(94)00857-4). Disponível em: <https://www.sciencedirect.com/science/article/pii/0378779694008574>.

DO COUTTO FILHO, Milton B.; STACCHINI DE SOUZA, Julio C.; AUGUSTO, Andre A. Critical measuring units for state estimation. en. *In: 2014 Power Systems Computation Conference*. Wrocław, Poland: IEEE, ago. 2014. p. 1–7. ISBN 978-83-935801-3-2. DOI: [10.1109/PSCC.2014.7038472](https://doi.org/10.1109/PSCC.2014.7038472). Disponível em: <http://ieeexplore.ieee.org/document/7038472/>. Acesso em: 24 jul. 2020.

DO COUTTO FILHO, Milton B.; STACCHINI DE SOUZA, Julio C.; TAFUR, J. E. Villavicencio. Quantifying Observability in State Estimation. **IEEE Transactions on Power Systems**, v. 28, n. 3, p. 2897–2906, ago. 2013.

DO COUTTO FILHO, Milton B. *et al.* Educational Tool for Power System State Estimation Teaching and Learning. **IEEE Transactions on Power Systems**, v. 38, n. 6, p. 5885–5895, nov. 2023. ISSN 1558-0679. DOI: [10.1109/TPWRS.2022.3229908](https://doi.org/10.1109/TPWRS.2022.3229908). Disponível em: <https://ieeexplore.ieee.org/abstract/document/9991052>. Acesso em: 8 mar. 2026.

DO COUTTO FILHO, Milton B.; STACCHINI DE SOUZA, Julio C.; SCHILLING, M. Th. Handling Critical Data and Observability. **Electric Power Components and Systems**, v. 35, n. 5, p. 553–573, 2007. Publisher: Taylor & Francis _eprint: <https://doi.org/10.1080/15325000601078187>. DOI: [10.1080/15325000601078187](https://doi.org/10.1080/15325000601078187). Disponível em: <https://doi.org/10.1080/15325000601078187>.

DO COUTTO FILHO, Milton B.. *et al.* Identifying critical measurements & sets for power system state estimation. *In: 2001 IEEE Porto Power Tech Proceedings (Cat. No.01EX502)*. Porto, Pt: [s. n.], set. 2001. v. 3, 6 pp.

ER, M. C. A Parallel Algorithm for Cost-Optimal Generation of Permutations of r out of n Items. **Journal of Information and Optimization Sciences**, v. 9, n. 1, p. 53–56, 1988. DOI: [10.1080/02522667.1988.10698905](https://doi.org/10.1080/02522667.1988.10698905).

EXPOSITO, A. G.; ABUR, A. Generalized observability analysis and measurement classification. *In: PROCEEDINGS of the 20th International Conference on Power Industry Computer Applications*. [S. l.: s. n.], mai. 1997. p. 97–103.

GOU, B. Jacobian matrix-based observability analysis for state estimation. **IEEE Transactions on Power Systems**, v. 21, n. 1, p. 348–356, 2006.

GOU, B.; ABUR, A. A direct numerical method for observability analysis. **IEEE Transactions on Power Systems**, v. 15, n. 2, p. 625–630, 2000.

GUGGARI, Shankru; V, Umadevi; KADAPPA, Vijayakumar. **Advances in Partitioning Techniques: A Prospective towards Artificial Intelligence**. New York: Chapman e Hall/CRC, 1 jun. 2025. 134 p. ISBN 978-1-003-47190-5. DOI: [10.1201/9781003471905](https://doi.org/10.1201/9781003471905).

HE, Haibo; YAN, Jun. Cyber-physical attacks and defences in the smart grid: a survey. en. **IET Cyber-Physical Systems: Theory & Applications**, v. 1, n. 1, p. 13–27, dez. 2016. ISSN 2398-3396. DOI: [10.1049/iet-cps.2016.0019](https://doi.org/10.1049/iet-cps.2016.0019). Disponível em: <https://digital-library.theiet.org/content/journals/10.1049/iet-cps.2016.0019>. Acesso em: 16 jul. 2020.

KORRES, G. N.; CONTAXIS, G. C. Identification and updating of minimally dependent sets of measurements in state estimation. **IEEE Transactions on Power Systems**, v. 6, n. 3, p. 999–1005, 1991.

KRUMPHOLZ, G. R.; CLEMENTS, K. A.; DAVIS, P. W. Power System Observability: A Practical Algorithm Using Network Topology. **IEEE Transactions on Power Apparatus and Systems**, PAS-99, n. 4, p. 1534–1542, 1980.

KURI, J.; GAGNAIRE, M.; PUECH, N. On the resource efficiency of virtual concatenation in SDH/SONET mesh transport networks bearing protected scheduled connections. **Journal of Lightwave Technology**, v. 23, n. 10, p. 3012–3023, 2005. DOI: [10.1109/JLT.2005.856246](https://doi.org/10.1109/JLT.2005.856246).

LUCCIO, F.; SAMI, M. On the Decomposition of Networks in Minimally Interconnected Subnetworks. **IEEE Transactions on Circuit Theory**, v. 16, n. 2, p. 184–188, 1969. DOI: [10.1109/TCT.1969.1082924](https://doi.org/10.1109/TCT.1969.1082924).

MASSAOUDI, Mohamed *et al.* Advancing Coherent Power Grid Partitioning: A Review Embracing Machine and Deep Learning. **IEEE Open Access Journal of Power and Energy**, v. 12, p. 59–75, 2025. ISSN 2687-7910. DOI: [10.1109/OAJPE.2025.3535709](https://doi.org/10.1109/OAJPE.2025.3535709). Disponível em: <https://ieeexplore.ieee.org/document/10855832>. Acesso em: 8 mar. 2026.

MONTENEGRO, Davis; RAMOS, Gustavo A.; BACHA, Seddik. Multilevel A-Diakoptics for the Dynamic Power-Flow Simulation of Hybrid Power Distribution Systems. **IEEE Transactions on Industrial Informatics**, v. 12, n. 1, p. 267–276, 2016. DOI: [10.1109/TII.2015.2506541](https://doi.org/10.1109/TII.2015.2506541).

MONTICELLI, A. **State Estimation in Electric Power Systems: A Generalized Approach**. [S. l.]: Springer US, 2012. (Power Electronics and Power Systems). ISBN 978-1-4615-4999-4. Disponível em: https://books.google.com.br/books?id=_MfkBwAAQBAJ.

MONTICELLI, A.; WU, F. F. Network Observability: Identification of Observable Islands and Measurement Placement. **IEEE Transactions on Power Apparatus and Systems**, PAS-104, n. 5, p. 1035–1041, 1985.

MONTICELLI, A.; WU, F. F. Observability Analysis for Orthogonal Transformation Based State Estimation. **IEEE Power Engineering Review**, PER-6, n. 2, p. 45–45, 1986.

MORI, H.; TSUZUKI, S. A fast method for topological observability analysis using a minimum spanning tree technique. **IEEE Transactions on Power Systems**, v. 6, n. 2, p. 491–500, 1991.

NEAPOLITAN, Richard E. **Foundations of Algorithms**. 5. ed. Burlington, MA: Jones & Bartlett Learning, 2015.

NISHIO, Ayres. **Uso de GPUs para a Análise Concorrente de Sistemas de Medição**. Fev. 2021. Tese (Doutorado) – Instituto de Computação, Universidade Federal Fluminense, Niterói, RJ, Brasil.

NISHIO, Ayres; CLUA, Esteban W. G. *et al.* GPU-Based Criticality Analysis Applied to Power System State Estimation. In: GERVASI, Osvaldo *et al.* (ed.). **Computational Science and Its Applications – ICCSA 2020**. Cham: Springer International Publishing, 2020. p. 121–133. ISBN 978-3-030-58808-3.

NISHIO, Ayres; DO COUTTO FILHO, Milton B.; STACCHINI DE SOUZA, Julio C. *et al.* State Estimation Extensive Criticality Analysis Performed on Measuring Units: A Comparative Study. **Journal of Control, Automation and Electrical Systems**, v. 35, n. 6, p. 1135–1146, dez. 2024. ISSN 2195-3880, 2195-3899. DOI: [10.1007/s40313-024-01130-9](https://doi.org/10.1007/s40313-024-01130-9). Disponível em: <https://link.springer.com/10.1007/s40313-024-01130-9>. Acesso em: 29 nov. 2025.

NISHIO, Ayres; DO COUTTO FILHO, Milton B.; STACHINNI DE SOUZA, Julio C. *et al.* GPU parallel processing to enable extensive criticality analysis in state estimation. **Concurrency and Computation: Practice and Experience**, v. 36, n. 20, e8200, 2024. _eprint: <https://onlinelibrary.wiley.com/doi/pdf/10.1002/cpe.8200>. ISSN 1532-0634. DOI: [10.1002/cpe.8200](https://doi.org/10.1002/cpe.8200). Disponível em: <https://onlinelibrary.wiley.com/doi/abs/10.1002/cpe.8200>. Acesso em: 8 mar. 2026.

NUCERA, R. R.; GILLES, M. L. Observability analysis: a new topological algorithm. **IEEE Transactions on Power Systems**, v. 6, n. 2, p. 466–475, 1991.

PEDREGOSA, Fabian *et al.* **Scikit-learn: Machine Learning in Python.**

[*S. l.: s. n.*], 2018. arXiv: [1201.0490](https://arxiv.org/abs/1201.0490) [[cs.LG](#)]. Disponível em:

<https://arxiv.org/abs/1201.0490>.

QUINTANA, V. H.; SIMOES-COSTA, A.; MANDEL, A. Power System Topological Observability Using a Direct Graph-Theoretic Approach. **IEEE Transactions on Power Apparatus and Systems**, PAS-101, n. 3, p. 617–626, 1982.

RUSKEY, Frank; WILLIAMS, Aaron. The coolest way to generate combinations.

Discrete Mathematics, v. 309, n. 17, p. 5305–5320, 2009. DOI:

[10.1016/j.disc.2007.11.048](https://doi.org/10.1016/j.disc.2007.11.048).

SCHWEPPE, F. C.; WILDES, J. Power System Static-State Estimation, Part I: Exact Model. **IEEE Transactions on Power Apparatus and Systems**, PAS-89, n. 1, p. 120–125, jan. 1970.

SHI, Jianbo; MALIK, J. Normalized cuts and image segmentation. **IEEE**

Transactions on Pattern Analysis and Machine Intelligence, v. 22, n. 8,

p. 888–905, 2000. DOI: [10.1109/34.868688](https://doi.org/10.1109/34.868688).

SIMOES COSTA, Antonio; PIAZZA, T.S.; MANDEL, Arnaldo. Qualitative methods to solve qualitative problems in power system state estimation. **Power Systems, IEEE Transactions on**, v. 5, p. 941–949, set. 1990. DOI: [10.1109/59.65924](https://doi.org/10.1109/59.65924).

SNODGRASS, Jonathan *et al.* Case Study of Enhancing the MATPOWER Polish Electric Grid. *In*: 2022 IEEE Texas Power and Energy Conference (TPEC). [*S. l.: s. n.*], 2022. p. 1–6. DOI: [10.1109/TPEC54980.2022.9750807](https://doi.org/10.1109/TPEC54980.2022.9750807).

SOARES LIMA, Rafael Carlos *et al.* Topological Approach for Identifying Critical Measurements and Sets in State Estimation. **IEEE Latin America Transactions**, v. 20, n. 1, p. 100–107, 2022. ISSN 1548-0992. DOI: [10.1109/TLA.2022.9662178](https://doi.org/10.1109/TLA.2022.9662178).

Disponível em: <https://ieeexplore.ieee.org/document/9662178/>. Acesso em: 11 jul. 2022.

SOLARES, C. *et al.* Binary-arithmetic approach to observability checking in state estimation. **IET Generation, Transmission & Distribution**, v. 3, n. 4, p. 336–345, 2009.

SOU, Kin Cheong; SANDBERG, Henrik; JOHANSSON, Karl Henrik. Computing Critical k -Tuples in Power Networks. **IEEE Transactions on Power Systems**, v. 27, n. 3, p. 1511–1520, ago. 2012. Conference Name: IEEE Transactions on Power Systems. ISSN 1558-0679. DOI: [10.1109/TPWRS.2012.2187685](https://doi.org/10.1109/TPWRS.2012.2187685).

WU, F. F.; MONTICELLI, A. Network Observability: Theory. **IEEE Transactions on Power Apparatus and Systems**, PAS-104, n. 5, p. 1042–1048, 1985.

ZETTER, K. **Inside the cunning, unprecedented hack of Ukraine's powergrid.** [*S. l.: s. n.*], 2016. Disponível em: <https://www.wired.com/2016/03/inside-cunning-unprecedented-hack-ukraines-power-grid>.

ZHANG, Miao; MIAO, Zhixin; FAN, Lingling. Power Grid Partitioning: Static and Dynamic Approaches. *In: 2018 NORTH AMERICAN POWER SYMPOSIUM (NAPS). 2018 North American Power Symposium (NAPS)*. Fargo, ND: IEEE, set. 2018. p. 1–6. ISBN 978-1-5386-7138-2. DOI: [10.1109/NAPS.2018.8600609](https://doi.org/10.1109/NAPS.2018.8600609). Disponível em: <https://ieeexplore.ieee.org/document/8600609/>. Acesso em: 15 abr. 2024.