PNetSCAN: uma abordagem de agrupamento em paralelo para resolução de problemas Big Data

Tales Lopes Silva

JUIZ DE FORA

NOVEMBRO, 2018

PNetSCAN: uma abordagem de agrupamento em paralelo para resolução de problemas Big Data

Tales Lopes Silva

Universidade Federal de Juiz de Fora

Instituto de Ciências Exatas

Departamento de Ciência da Computação

Bacharelado em Ciência da Computação

Orientador: Victor Ströele de Andrade Menezes

JUIZ DE FORA

NOVEMBRO, 2018

PNETSCAN: UMA ABORDAGEM DE AGRUPAMENTO EM PARALELO PARA RESOLUÇÃO DE PROBLEMAS BIG DATA

Tales Lopes Silva

MONOGRAFIA SUBMETIDA AO CORPO DOCENTE DO INSTITUTO DE CIÊNCIAS EXATAS DA UNIVERSIDADE FEDERAL DE JUIZ DE FORA, COMO PARTE INTEGRANTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE BACHAREL EM CIÊNCIA DA COMPUTAÇÃO.

Aprovada por:

Victor Ströele de Andrade Menezes
Doutor

Regina Maria Maciel Braga Villela
Doutora

Mario Antonio Ribeiro Dantas
Doutor

JUIZ DE FORA
DE NOVEMBRO, 2018

À Deus em primeiro lugar, sempre.

Aos meus pais.

À minha amada irmã.

À família.

Aos amigos de verdade.

Resumo

Big Data é um termo referente à crescente emergência de dados que ocorre, de maneira contínua e simbólica, na atualidade. Em decorrência da maior disponibilidade de dados, é de interesse que existam pesquisas voltadas para o estudos de técnicas de mineração e análise de dados, responsáveis por extrair informação por dentre esse mundo de conteúdo, a respeito do domínio ao qual pertencem. Agrupamento (Clustering) são técnicas responsáveis por identificar grupos de dados com características semelhantes e oferecer resultados com base na quantidade e qualidade dos grupos encontrados. Dentre os algoritmos existentes destaca-se o DBSCAN, um algoritmo baseado em densidade, com a particularidade de conseguir identificar conjuntos de dados com formas bem definidas. NetSCAN, um algoritmo baseado no DBSCAN, foi desenvolvido para atuar em redes sociais, como é a base de pesquisadores DBLP, considerando especificidades de implementação como o bidirecionamento e a sobreposição de nós em grafos. Esse trabalho propõe e estuda estratégias relacionadas à otimização de algoritmos de agrupamento de dados. Uma heurística, proposta, desenvolvida e implementada, relaciona o particionamento do conjunto de dados com a união (*merge*) dos resultados parciais, obtidos pelo processo de agrupamento em paralelo, originando o PNetSCAN, uma abordagem paralela do NetSCAN. Os resultados, referentes aos experimentos realizados, confirmam a investigação literária. Uma melhora de, pelo menos, 25% no tempo total de processamento, foi obtida na aplicação do método proposto nesse trabalho.

Palavras-chave: Big Data, Mineração de dados, Agrupamento, Clustering, Paralelismo, Particionamento, NetSCAN, PNetSCAN.

Abstract

Big Data is a term referring to the increasing emergence of data that occurs, in a growing and symbolic way, at the present time. As result of greater data availability, it is of interest that there are studies focused on the mining and data analysis, responsible for extracting information from this world of content, related the domain to which they belong. Clustering are techniques responsible for identifying data groups with similar characteristics and offering results based on the quantity and quality of the found groups. Among the existing algorithms, DBSCAN is a density-based algorithm, with the particularity of being able to identify well-defined data sets. NetSCAN, an DBSCAN based algorithm, was developed to act in social networks, as example the researchers basis: DBLP, considering implementation specificities such as bidirection and nodes overlaps in graphs. This work proposes and studies strategies related to the optimization of clustering algorithms. A heuristic, proposed, developed and implemented, relates data set partitioning with the partial results' union (*merge*), obtained by the parallel grouping processes, originating the PNetSCAN, a NetSCAN parallel approach. Results confirm literary researchs, an improvement of at least 25 % in time processing, was obtained in the method application proposed in this work.

Keywords: Big Data, Data Mining, Clustering, Parallelism, Partitioning, NetSCAN, PNetSCAN.

Agradecimentos

Agradeço a Deus pelo maior presente que me foi dado, a vida.

À minha mãe Rosania, por ser sempre um exemplo de resiliência e de amor incondicional, me guiando pelos caminhos místicos e ajudando a manter minha paz.

Ao meu pai Delci pela compreensão e incentivo, pela força e sensatez, sempre estando ao meu lado e ensinando que *"a vida é bela e maravilhosa"*.

À minha irmã Paula, *gudiguinha*, por ser minha maior fonte de amor e incentivo de ser, cada vez mais, um ser humano melhor.

À minha amada avó Laide, por ser o maior exemplo de pessoa, sempre irradiando bondade e espiritualidade.

Aos meus familiares pelo apoio e torcida, sempre desejando o melhor de mim.

Aos amigos de verdade, irmãos de outras vidas, que estão sempre ao meu lado me tornando uma pessoa mais honrada.

À querida *Lady*, que tanto me abriu os olhos e ensinou em tão pouco tempo, se tornando uma companheira nas lutas diárias da vida.

À Julia, por sempre me estender a mão e ajudar a me levantar, em todos os momentos.

Ao bichano Dudu, meu maior companheiro.

Agradeço ao professor Victor Ströele, não só pela ajuda e orientação nesse trabalho, mas pela extrema compreensão e empatia, se tornando não só uma inspiração, mas um amigo.

Agradeço aos integrantes, alunos e professores, do Grupo de Educação Tutorial da Computação (GETComp - UFJF), pelo crescimento pessoal e profissional, além da amizade e carinho, sempre.

Aos estimados professores do DCC, em especial o professor Stênio e o professor Marcelo Caniato pelo apoio em momentos cruciais durante minha graduação, atuando como verdadeiros exemplos a serem seguidos.

Aos professores de outras instituições, tanto do ICE, quanto fora, que me auxiliaram e fizeram parte dessa jornada.

Aos funcionários da UFJF que humildemente contribuem para o desenvolvimento e trabalho de todos no meio de ensino, duas coisas pouco valorizadas nesse país.

Finalmente, à todas as pessoas de bom coração que fazem/fizeram parte, direta ou indiretamente, da minha vida, dia após dia, onde sei que, o caminho é tortuoso e todos estamos travando nossas próprias batalhas.

"Não são as perdas nem as quedas que podem fazer fracassar nossas vidas, senão a falta de coragem para levantar e seguir em frente."

Samael Aun Weor

Conteúdo

Lista de Figuras

Lista de Tabelas

Lista de Abreviações

DCC	Departamento de Ciência da Computução
UFJF	Universidade Federal de Juiz de Fora
AGM	Árvore Geradora Mínima
SPMD	Single Program Multiple Data
CUDA	Compute Unified Device Architecture
GPU	Graphics Processing Unit
FPGA	Field Programmable Gate Array
ICFSKM	Incremental CFS Algorithm Based on K-mediods
PICFSKM	Parallel ICFSKM
DBLP	"The DBLP Computer Science Bibliography"
TCC	Trabalho de Conclusão de Curso

1 Introdução

Vivemos em uma era onde a constante evolução tecnológica e o crescimento digital estão presentes no dia-a-dia das pessoas, onde a capacidade e fontes para geração de dados crescem progressivamente e, formas de lidar com tamanha quantidade de informação são, cada vez mais, foco de pesquisa e desenvolvimento (LETOUZÉ, 2013). O volume total de dados digitais, acumulado durante anos, hoje facilmente corresponde ao volume que é gerado em algumas horas por alguma fonte ou aplicação específica (LOHR, 2012). Dados são utilizados em diversos tipos de serviços e podem, por exemplo, atuar como fonte de abastecimento para sistemas de recomendação, responsáveis por auxiliar na tomada de decisão em diversas áreas de desenvolvimento.

Big Data é um conceito diretamente relacionado ao crescente volume de dados que inclui desafios e oportunidades associados a esse tipo de recurso (HILBERT; LÓPEZ, 2011)(MARR, 2014), em outras palavras, também pode ser definido pela presença da grande quantidade de dados na atualidade e às formas de lidar com essa vazão de informação para uma maior compreensão de mundo. Pode-se dizer, ainda, que o termo é recente devido ao crescimento da necessidade, disponibilidade e acessibilidade de componentes eletrônicos, cujo o aumento influencia diretamente no crescimento da rede denominada "internet das coisas"(LOHR, 2012)(LETOUZÉ, 2013)(MARR, 2013). Para (LOHR, 2012) e (RAINIE; WELLMAN, 2012) o cenário de geração de grande volumes de dados[1] acarreta no surgimento de várias questões relacionadas a esse contexto, são essas: (i) capacidade de armazenamento, (ii) complexidade de algoritmos para manipulação de dados, (iii) tratamento e pré-processamento de conjuntos de dados, além de (iv) estratégias de extração de conhecimento, de forma eficiente, dentro de bancos de dados.

"Dados não estão apenas se tornando mais disponíveis, como também estão sendo melhor 'compreendidos' pelos computadores"(LOHR, 2012) e têm sido utilizados em diversas áreas da tecnologia como, processamento de linguagem natural, reconhecimento de

[1] "No início do ano 2003 havíamos criado apenas cinco exabytes de dados, atualmente criamos esse valor a cada dois dias. Prevê-se que esse valor aumente em cinquenta vezes até 2020.- Hal Varian, Economista Chefe da Google

padrões e aprendizado de máquina. A crescente disponibilidade de dados gera um número maior de informação a ser avaliada por usuários, acarretando em dificuldades na busca por conteúdo de relevância, como ocorre na pesquisa por material educacional na internet, onde diferentes repositórios abrigam diferentes tipos de dados de distintos assuntos e interesses.

O conhecimento implícito em bases de dados, muitas vezes não obtido de forma simples e direta, pode ser adquirido através de uma variedade de técnicas e algoritmos responsáveis por minerar dados e fornecer informações valiosas a respeito do meio de origem (WU et al., 2014). Mineração de Dados implica no processo de análise de grandes volumes de dados a fim de encontrar relações indiretas e colocá-las de maneira simples e resumida para que seja útil ao seu proprietário (HAND, 2007).

O modo de escolha, que antes era baseado em experiências obtidas, está sendo cada vez mais baseado em análise de dados (LOHR, 2012). Muitas empresas e instituições já investem e possuem grande interesse nos ganhos e benefícios que a análise de dados pode trazer, visto que, muito do que se refere à Big Data está diretamente relacionado à questões de mudança e transformação do meio. Isso implica nos negócios através de predições mais inteligentes trazendo vantagens em um cenário competitivo (MCAFEE et al., 2012). Mas não apenas empresas, que visam o lucro, são os agraciados dos benefícios da utilização de técnicas de mineração de dados, outras áreas como saúde, segurança e economia produtiva, consideradas áreas críticas e que visam o desenvolvimento na sociedade, também fazem uso dos ganhos e da evolução que a otimização do processo de tomada de decisão proporciona (KUMAR, 2016).

De acordo com Leskovec, Rajaraman e Ullman (2014), "O Maior desafio para aplicações Big Data é explorar grandes volumes de dados e extrair conhecimento para ações futuras". Wu et al. (2014) complementam afirmando que "o processo de extração de informação deve ser eficiente em tempo real, visto que, guardar toda informação observada seria inútil". O armazenamento de dados é um problema diante do cenário de crescimento ininterrupto de dados, guardar os dados observados a todo instante é algo com o custo extremamente alto e, consequentemente, inviável em vários sentidos. Porém, a inviabilidade não ocorre somente em questões de armazenamento, ela também se volta

para outros caminhos como o de processamento, constituindo um enorme desafio a busca por tratamento, análise e obtenção de resultados de maneira ágil e viável, em tempo real. Dessa forma, é de interesse científico e social, além de se fazer necessário, que existam pesquisas voltadas ao desenvolvimento e otimização de algoritmos de mineração e análise de dados, gerando o aperfeiçoamento de recursos e técnicas que auxiliam e impactam em diversas áreas do conhecimento.

Problemas que envolvem a necessidade do uso de técnicas de mineração de dados, mais especificamente técnicas de agrupamento (clustering)[2], requerem muito recurso computacional e possuem complexidade associada à questões de análise combinatória, onde existe a característica do tempo de execução crescer de acordo com o valor de entrada do algoritmo (SIPSER, 2006), ou seja, a quantidade de dados presentes no conjunto a ser analisado. Esse crescimento muitas vezes ocorre em uma curva representada por uma função polinomial onde, dependendo do tamanho da entrada, obter a solução ideal do problema pode ser inviável.

Todo o panorama apresentado até o momento aponta para a necessidade da implementação de algoritmos e métodos inteligentes capazes de processar, de maneira poderosa, essa gama de dados. Dessa forma, a proposta principal desse trabalho é abordar o estudo e utilização de processamento paralelo, ou paralelismo, em problemas de agrupamento de dados. A ideia surge da necessidade de viabilização e melhoria do tempo de execução de algoritmos de agrupamento de dados em bases de grande porte (ZHANG et al., 2017). A abordagem proposta envolve o particionamento do conjunto de dados e a aplicação do agrupamento em partes menores e, consequentemente, menos complexas. Para que a abordagem seja eficaz, as partes do conjunto devem ser resolvidas separadamente e seus resultados unidos a fim de otimizar o tempo total de execução. O esperado é que, com a base dividida em porções menores, a aplicação do algoritmo nesses grupos reduzidos permita a obtenção de um tempo de processamento melhor do que o tempo quando o algoritmo é aplicado sobre base completa. Com isso, a questão de pesquisa que norteia esse trabalho foi definida como: -"*É possível obter uma melhora no tempo de processamento, tendo em vista a existência de uma heurística de paralelismo válida, para o problema de*

[2]Agrupamento (Clustering) é um método de aprendizado, não-supervisionado, relacionado ao reconhecimento de padrões em mineração de dados (JAIN; MURTY; FLYNN, 1999).

agrupamento de dados?"

A hipótese é que, aplicando a ideia de paralelismo, seja observada uma melhora significativa no tempo total do processo de agrupamento de dados, embora uma etapa crucial seja a união das partições levando a um resultado válido. A execução do procedimento envolve, (i) particionamento da base, representada através de um grafo, (ii) aplicação do algoritmo de agrupamento nas partes da base, (iii) união das partições e, (iv) validação do resultado final. O fluxo que representa as etapas enumeradas, assim como o trabalho desenvolvido nesse TCC, pode ser visto na figura 1.1. A validação da heurística é realizada através da análise de ambos os resultados, tanto o obtido agrupando a base particionada, de forma paralela, quanto agrupando a base inteira e, caso o resultado final seja idêntico para ambos, o procedimento é considerado válido.

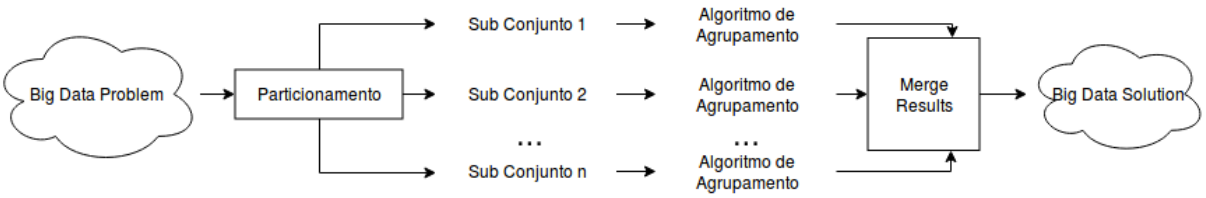

Figura 1.1: Fluxo de trabalho ($Workflow$) relacionado ao problema a ser resolvido.

Uma outra proposta é que, realizada a verificação e confirmada a corretude do método proposto, seja feito um estudo para relacionar a quantidade de partições com o tempo de processamento. Espera-se identificar, através desse estudo, o ponto onde o aumento das partições faça com que o tempo de processamento pare de melhorar, ou seja, é esperado que exista um valor ótimo para o número de partições.

Para o desenvolvimento da pesquisa foi necessário o estudo e entendimento do algoritmo NetSCAN (HORTA, 2017)(HORTA et al., 2018), um algoritmo baseado no DBSCAN (ESTER et al., 1996) que utiliza, da mesma maneira, a ideia de densidade dos vértices como estratégia para o cálculo dos agrupamentos. Embora o algoritmo possua eficácia no agrupamento de dados, os resultados obtidos, principalmente em relação à base de dados DBLP(LEY, 2002), demonstram sua inviabilidade em relação ao tempo de processamento para encontrar soluções em bases de grande porte, resultando, assim, na ideia e iniciativa do desenvolvimento de uma aplicação paralela do NetSCAN; PNetSCAN (Parallel NetSCAN). Neste contexto, esse trabalho propõe uma forma eficiente para

agrupar grandes conjuntos de dados, permitindo a extensão da estratégia, inclusive, para outros algoritmos.

Em relação às principais contribuições desse trabalho, destancam-se a (i) proposta de uma estratégia de paralelismo para técnicas de agrupamentos baseadas em densidade, (ii) desenvolvimento de um algoritmo para o particionamento de conjuntos de dados em grafos, (iii) estudo da atuação do algoritmo NetSCAN, de forma paralela, e, (iv) análise do número de partições em relação ao tempo de processamento do algoritmo, incluindo a busca pelo valor ótimo para o número de partições para um determinado conjunto de entrada. Além das contribuições citadas, o trabalho possui o objetivo de agregar tanto no contexto acadêmico, servindo como fonte de referência e pesquisa para trabalhos futuros, quanto em questões de desenvolvimento, fornecendo novos algoritmos e estratégias para lidar com problemas relacionados à análise de dados.

Este trabalho se baseia e busca agregar em pesquisas e trabalhos já avançados do NEnC (Grupo de Pesquisa em Engenharia de Conhecimento) através da otimização de processos e algoritmos relacionados à análise de redes sociais científicas (MENEZES et al., 2015)(STRÖDE et al., 2016)(STRÖELE; ZIMBRÃO; SOUZA, 2013)(ALMEIDA et al., 2016)(HORTA, 2017)(HORTA et al., 2018).

O restante do trabalho está estruturado da seguinte maneira, o capítulo 2 apresenta o referencial teórico relacionado aos principais temas desenvolvidos nesse trabalho: agrupamento, em especial o NetSCAN, paralelismo e particionamento. O capitulo 3 descreve a abordagem de agrupamento proposta, PNetSCAN. No capítulo 4 são apresentados os experimentos e resultados obtidos na realização desse trabalho, enquanto que o Capítulo 5 expõe as considerações finais e trabalhos futuros.

2 Paralelismo em Algoritmos de Agrupamento

A análise de dados é uma área que envolve um conjunto de técnicas e está presente em diversas aplicações computacionais. Dentre esse conjunto de aplicações existem aquelas responsáveis pela descoberta de conhecimento, chamadas de técnicas de mineração de dados, que podem ser definidas como uma forma semi-automatizada de analisar grandes volumes de dados (TALIA, 2002). Técnicas de agrupamento, ou *clutering*, são técnicas de mineração de dados responsáveis por arranjar grupos, levando em consideração a similaridade ou dissimilaridade existente entre objetos, onde objetos mais semelhantes tendem a pertencer ao mesmo conjunto.

Uma propriedade importante é que agrupamento é um método de classificação não supervisionada, ou seja, não existe uma classificação prévia dos dados, sendo obtida, em primeira instância, através da aplicação do algoritmo (TALIA, 2002)(HAND, 2007). Por conseguinte, algoritmos de agrupamento são técnicas de análise não supervisionada de dados onde algumas das principais estratégias são: (i) agrupamento hierárquico, (ii) agrupamento utilizando particionamento, (iii) agrupamento do vizinho mais próximo, (iv) agrupamento difuso, (v) agrupamento utilizando redes neurais artificiais, (vi) agrupamento por densidade, e (vii) abordagens evolucionárias para agrupamento (JAIN; MURTY; FLYNN, 1999).

Formalmente, algoritmos de agrupamento têm como objetivo definir conjuntos $S = \{s_1, s_2, ..., s_n\}$ onde, vértices de um mesmo grupo são mais semelhantes entre si do que com vértices de outros grupos. Um modelo para o problema de agrupamento pode ser definido por (2.1). Alguns algoritmos aceitam a existência de nós que não pertencem a nenhum grupo, esses nós são chamados de *noise*.

$$\forall v \in V, v \in S_i \tag{2.1}$$

DBSCAN, proposto por Ester et al. (1996), é um algoritmo de agrupamento

baseado em densidade que agrupa pontos próximos no espaço dimensional, utilizando dois parâmetros (ϵ,$minPts$), onde ϵ representa o raio mínimo para que dois vértices possam pertencer ao mesmo grupo k e, $minPts$ é o número mínimo de pontos necessários, dentro de um raio ϵ em torno de um vértice v, para que v seja considerado *core*[3] e possa agrupar os vértices dentro de ϵ, chamados *borderpoints*.

2.1 NetSCAN

NetSCAN é um algoritmo que utiliza a estratégia de agrupamento por densidade e foi apresentado em (HORTA, 2017), com o intuito de identificar grupos de pesquisadores em redes sociais científicas. O NetSCAN é baseado no DBSCAN, porém, apresenta algumas diferenças em relação a esse algoritmo: (i) considera o bi-direcionamento em grafos, ou seja, o peso do relacionamento entre dois elementos depende da direção da arco entre eles, (ii) permite a existência de nós sobrepostos, que são nós pertencentes simultaneamente a diferentes grupos e, (iii) possui um terceiro parâmetro opcional (*raio*) que permite a expansão da busca por elementos, influenciados pelo *core*, em uma profundidade maior. De acordo com (HORTA, 2017), a ideia do terceiro parâmetro é conseguir identificar pesquisadores, influenciados por intermédio de outros pesquisadores, e associá-los ao grupo do influenciador central.

O funcionamento do algoritmo consiste em selecionar, de forma aleatória, um vértice inicial v_x para servir como ponto de partida e retornar todos os vizinhos influenciados por v_x em uma distância máxima ϵ. Se a quantidade de nós influenciados por v_x for maior ou igual a $MinPts$, um grupo k é formado e v_x é considerado um *core*, podendo ser expandido. A expansão consiste no agrupamento do conjunto de nós vizinhos $W = \{w_1, w_2, ..., w_n\}$ de v_x, caso exista algum nó w em W definido como *core*, seus vizinhos também são agrupados em k. Quando o número de nós influenciados por v_x é menor do que $MinPts$, v_x é marcado como *noise* e, caso seja agrupado por algum nó *core*, passa a ser considerado um *border point*. Todos os rótulos dos vértices são estabelecidos na execução do programa e irão, posteriormente, auxiliar na execução da abordagem

[3]Nó centralizador que atende ao requisito de número mínimo de nós ($minPts$) em seu raio de vizinhança ϵ.

paralela na identificação de nós que já foram percorridos.

NetSCAN é o algoritmo alvo da abordagem de paralelismo proposto nesse trabalho, principalmente, pela motivação do algoritmo ser inviável em questões de tempo de processamento para bases de grande porte. O algoritmo possui suas funções implementadas através de *queries* executáveis no banco de dados orientado a grafo, Neo4j. Para que o algoritmo seja executado, é necessário chamar a seguinte *query* no console do Neo4j, ou em alguma implementação integrada ao banco.

- *Query: "CALL netscan.find_communities("Pessoa","Publicou","idpessoa","total", MinPts, ϵ, raio)"*

Os parâmetros passados, em ordem, na chamada da *query* são, rótulo dos vértices, rótulo das arestas, atributo identificador de vértices, peso das arestas, número mínimo de pontos, *epsilon* (distância mínima) e o parâmetro opcional, raio, definido por padrão com valor 1, para que a busca possa se aprofundar em diferentes níveis no grafo. As principais funções e procedimentos são mostradas nos algoritmo 1, algoritmo 2 e algoritmo 3.

Algorithm 1 NetSCAN (HORTA, 2017)

Input: grafo, eps, MinPts, raio

```
clusterId = 0
while n = buscaNoSemRotulo(grafo) do
    clusterId = clusterId + 1                          ▷ incrementa cluster
    vizinhos = buscaPorRegiao(n, grafo, eps, raio)
    expandirNo(n, vizinhos, grafo, eps, minPts, clusterId, raio)
end while
```

Algorithm 2 expandirNo (HORTA, 2017)

Input: n,vizinhos,grafo,eps,minPts,clusterId,raio

```
quantidadeVizinhos = tamanho(vizinhos)
if numVizinhos < minPts then
    defineNoise(n)
    retornafalso
else if NosPercorridos ≥ lim_sup then
    defineCore(n)
    agrupaVizinhos(N, vizinhos, grafo, eps, minPts, clusterId, raio)
end if
```

Algorithm 3 agrupaVizinhos (HORTA, 2017)

Input: n,vizinhos,grafo,eps,minPts,clusterId,raio

15 $agrupar(n, vizinhos, clusterId)$ ▷ agrupa vértices vizinhos à n
16 $i = 0$
17 **while** $i < tamanho(vizinhos)$ **do**
18 $\quad n = vizinhos[i]$
19 $\quad novosVizinhos = buscaPorRegiao(n, grafo, eps, raio)$
20 $\quad expandirNo(n, novosVizinhos, grafo, eps, minPts, cluster, Id, raio)$
21 $\quad i = i + 1$
22 **end while**

2.2 Paralelismo

Embora cada algoritmo seja eficiente em seu respectivo cenário, quando utilizados para classificar grandes conjuntos de dados, algoritmos de agrupamento tendem a ter um custo computacional relativamente alto. Impulsionado pela constante evolução de questões relacionadas à *big data* e, proporcional ao crescimento da quantidade de informação gerada e armazenada, vários estudos envolvendo paralelismo, aplicados à algoritmos de agrupamento, existem e vêm sendo desenvolvidos atualmente. Na busca por autores que levantam essas questões vários trabalhos puderam ser encontrados e contribuíram para o embasamento teórico e o desenvolvimento desse trabalho.

Em relação às estratégias de paralelismo, Talia (2002) identifica as três principais utilizadas em algoritmos de agrupamento de dados, são elas: (i) *paralelismo independente*, onde os processadores não comunicam entre si e utilizam todo o conjunto de dados para a operação, (ii) *paralelismo de tarefas*, onde cada processador utiliza diferentes procedimentos e funções no conjunto de dados, completo ou parcial, e (iii) *paralelismo SPMD* (*Single Program Multiple Data*), onde vários processadores utilizam o mesmo algoritmo em diferentes partições da base de dados e realizam a troca de resultados parciais entre si.

De acordo com Kim (2009), a maioria das técnicas de agrupamento em paralelo utilizam uma forma de combinação entre paralelismo de tarefas e paralelismo SPMD, juntamente com uma arquitetura cliente/servidor, ou master/slave, [4] para a comunicação. Embora existam muitas formas de se abordar problemas de agrupamento utilizando para-

[4]Programa (master) designa tarefas para outros programas (slaves) e a comunicação entre computadores é feita, através da troca de mensagens.

lelismo, a melhor forma não é facilmente analisada e descobrir o procedimento ideal é uma tarefa complexa e custosa (SKILLICORN, 1999). De acordo com as estratégias citadas, é possível observar que a abordagem proposta nesse trabalho se aproxima do paralelismo SPMD, exceto pela troca de mensagens entre computadores durante o processo de agrupamento, onde apenas resultados parciais, gerados a cada iteração, serão analisados em conjunto.

É interessante notar que, assim como o paralelismo é utilizado para otimizar algoritmos de agrupamento e outras formas de análise de dados, observamos que algoritmos de agrupamento também auxiliam no processo de paralelizar algumas tarefas (KIM, 2009).

PDBSCAN, apresentado por Xu, Jäger e Kriegel (1999), é o esquema paralelo relacionado ao algoritmo de agrupamento DBSCAN, que se associa à três fatores, i) requer conhecimento mínimo do dominio, ii) consegue descobrir grupos com formatos (disposição dos nós) arbitrários, e iii) possui eficiência em grandes conjuntos de dados. Seus resultados demonstram uma aceleração linear no tempo de processamento em relação ao número de computadores utilizados.

Tian et al. (2005) ressaltam a ineficiência do algoritmo k-means para trabalhar com grandes conjuntos de dados e estudam sua aplicação em paralelo, suas análises demonstram um ganho em eficiência e espaço. Em contrapartida, Yang et al. (2014) não implementam um k-means paralelo, mas o aplicam em arquiteturas que proveem paralelismo, entre as implementações estão, CUDA para GPU, Mitrion C para FPGA, MPI para beowulf e OpenMP para máquinas de memória compartilhada. Seus resultados demonstram escalabilidade em cada uma das implementações e afirmam o menor custo para GPU e FPGA em relação às outras implementações.

Finalmente, em um trabalho mais recente, Zhang et al. (2017) focam na ideia de que muitos algoritmos de agrupamento lidam apenas com dados estáticos enquanto que existe a necessidade, industrial por exemplo, da análise dinâmica de quantidades significativas de dados. Para isso, são definidas duas operações, *cluster creating* e *cluster merging*, em conjunto com o k-mediods, para o desenvolvimento de uma abordagem de agrupamento incremental: ICFSKM. PICFSKM, sua abordagem em paralelo, é implementada e seus resultados em uma base real indicam um ganho de quase 3 horas em tempo de

processamento.

2.2.1 Particionamento

A complexidade e o volume de dados a ser processado pelos algoritmos de agrupamento tornam o processamento completo do conjunto de dados inviável, sendo necessário o particionamento do mesmo para um processamento em etapas. Neste contexto, cada parte do conjunto de dados é processada separadamente e, após a execução de todas as etapas, os resultados parciais são unidos para obtenção do resultado geral.

No contexto deste trabalho, os conjuntos de dados, obtidos a partir de fontes específicas, constituem estruturas de natureza complexa que podem ser representadas através de redes de relacionamentos: grafos (STROGATZ, 2001). Essas estruturas representam conexões entre objetos (nós ou vértices) que possuem características específicas (atributos) e relações (arestas) uns com outros. Padrões contidos nesses dados podem ser analisados a fim de obter informações presentes, direta ou indiretamente, no conjunto.

Um grafo é uma estrutura que pode ser definida por $G = \{V, E\}$, onde $V = \{v_1, v_2, ..., v_n\}$ corresponde a um conjunto, não vazio, contendo n vértices e $E = \{e_i j, e_j i\}$ corresponde ao conjunto de arestas que conectam pares não ordenados de vértices, V_i e V_j, presentes em V. As arestas podem ser direcionadas, ou não, e ter um peso associado.

O particionamento em grafos é um problema de alta complexidade que foi provado ser um problema da classe NP-completo (JOHNSON et al., 1989). Esse problema também pode ser considerado um problema de agrupamento, visto que, a ideia é dividir o grafo em partições ou grupos menores onde o objetivo é encontrar grupos de vértices, de forma que as arestas entre vértices do mesmo grupo possuam um peso elevado e o peso das arestas entre diferentes grupos seja menor.

Para modelar o problema como um problema de otimização combinatória, deve-se ter uma função de avaliação cujo objetivo é encontrar partições que maximizem a densidade intracluster (2.2) e minimizem a esparsidade intercluster (2.3) (NEWMAN, 2006). O problema pode ser definido como uma dupla (k, v) onde o objetivo é particionar G, que contém n vértices, em k componentes com o limite de tamanho definido pela

equação (2.4), com v a ser estabelecido.

$$maximizar \sum peso(e_{ij}) \mid v_i, v_j \in k_z \tag{2.2}$$

$$minimizar \sum peso(e_{ij}) \mid k(v_i) \neq k(v_j) \tag{2.3}$$

$$v * (n/k) \tag{2.4}$$

Um caso clássico do problema de particionamento é o problema da bisseção, cujos valores de k e v são 2 e 1, respectivamente. Para o problema de particionamento balanceado, o objetivo é encontrar o custo mínimo de particionar G em k componentes, com cada uma contendo um número máximo de nós definido por (2.5), esse custo é comparado com o custo do corte (k,1) (ANDREEV; RACKE, 2006).

$$(1 + e) * (n/k) \tag{2.5}$$

O teorema de separação planar diz que qualquer grafo planar pode ser particionado em partes quase iguais removendo-se exatamente $\sqrt{n}$ vértices (LIPTON; TARJAN, 1979).

2.3 Considerações Finais do Capítulo

Neste capítulo foram apresentados conceitos fundamentais para o entendimento deste TCC, bem como trabalhos relacionados à pesquisa desenvolvida. Os três temas principais abordados no desenvolvimento da proposta foram discutidos, a saber: Agrupamento, Paralelismo e Particionamento.

Os trabalhos elencados neste capítulo serviram como referencial teórico e como motivação para o problema abordado nesse trabalho. A composição desse projeto pode ser tratada como um estudo exploratório, relacionado à paralelismo, onde uma abordagem quantitativa, com resultados mensuráveis, deve prover uma definição qualitativa da estratégia utilizada. Este trabalho se diferencia dos demais pela proposta e descrição de métodos relacionados a diversas etapas que permitem uma abordagem paralela, princi-

palmente sobre um contexto focado em um algoritmo que considera o bidirecionamento e a sobreposição de nós, voltado para análise de redes sociais científicas. Todo o estudo realizado aborda temas já bem conhecidos e propõe uma heurística para o particionamento balanceado de um grafo e uma heurística para a união dos resultados parciais, obtidos pelos agrupamentos em paralelo.

3 PNetSCAN: *Parallelized NetSCAN*

NetSCAN foi desenvolvido para atuar em redes sociais de grande porte, e possui resultados sobre a rede científica da DBLP (HORTA, 2017). Todavia o tempo de execução se torna um fator inviável devido à alta complexidade da rede, relacionada ao número de relações presentes na base.

Através da motivação definida acima, uma abordagem paralela do NetSCAN, denominada PNetSCAN, foi desenvolvida em busca de alcançar os mesmos resultados, obtidos através do processamento sequencial, porém, em um tempo viável de processamento. A figura 3.1 apresenta o funcionamento do PNetSCAN, os passos considerados no desenvolvimento da abordagem paralela são enumerados a seguir:

1. Particionamento do conjunto de dados em t grupos.
2. Execução do algoritmo NetSCAN, de forma paralela, sobre as t partições. Armazenar os resultados parciais obtidos em cada porção.
3. Realização do processo de união (*merge*) dos resultados parciais para obtenção do resultado final.

3.1 Estratégia de Particionamento

Sabe-se que particionar um conjunto de dados é uma tarefa extremamente difícil, principalmente se for levado em consideração algum tipo de critério a ser atendido. Para o presente trabalho, a tarefa de particionamento é necessária apenas como fornecimento de subconjuntos de dados sobre os quais o algoritmo de agrupamento será aplicado concorrentemente. Sendo assim, só é necessário que o conjunto seja particionado de forma balanceada em relação ao número de nós; outras restrições não foram consideradas nesse momento.

Podemos considerar que o particionamento ideal consiste na divisão de um grafo em subconjuntos iguais, porém, por uma série de fatores, se torna impraticável. Para

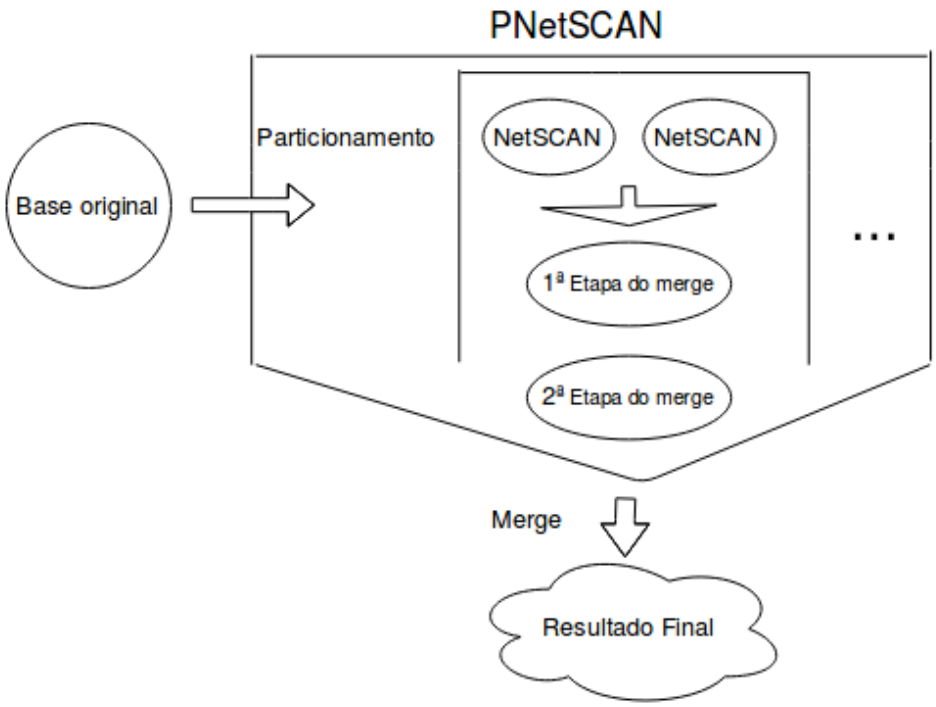

Figura 3.1: Abordagem paralela do NetSCAN: PNetSCAN.

essa tarefa, foi considerado um limite inferior e um limite superior para a quantidade de nós que cada partição pode ter. Seja n o número de vértices presentes no grafo, os limites utilizados equivalem a +10% e -10% de n, (3.1) e (3.2). Embora, para esse trabalho, esses tenham sido os limites escolhidos, o algoritmo demonstrou funcionar e encontrar partições com limites menores, como, 5%, 2% ou 1%, caso a estrutura do grafo permita essa configuração. A figura 3.2 ilustra uma situação onde o particionamento mais balanceado possível é de, respectivamente, 6 e 11, para o número de vértices em cada partição. Nesse exemplo, o nó central pertence à ambas as partições onde, +18,75% e -22,5%, são os limites encontrados.

$$LimiteSuperior : n/2 + n/10 \tag{3.1}$$

$$LimiteInferior : n/2 - n/10 \tag{3.2}$$

A estratégia utilizada para resolver o problema de particionamento consiste na ideia de escalonar, diferentes sub-árvores de G, em dois grupos, de acordo com sua quantidade de vértices. O método consiste em selecionar um vértice, ou um dos vértices, do grafo, que contenha o maior grau para servir como ponto de partida da construção da

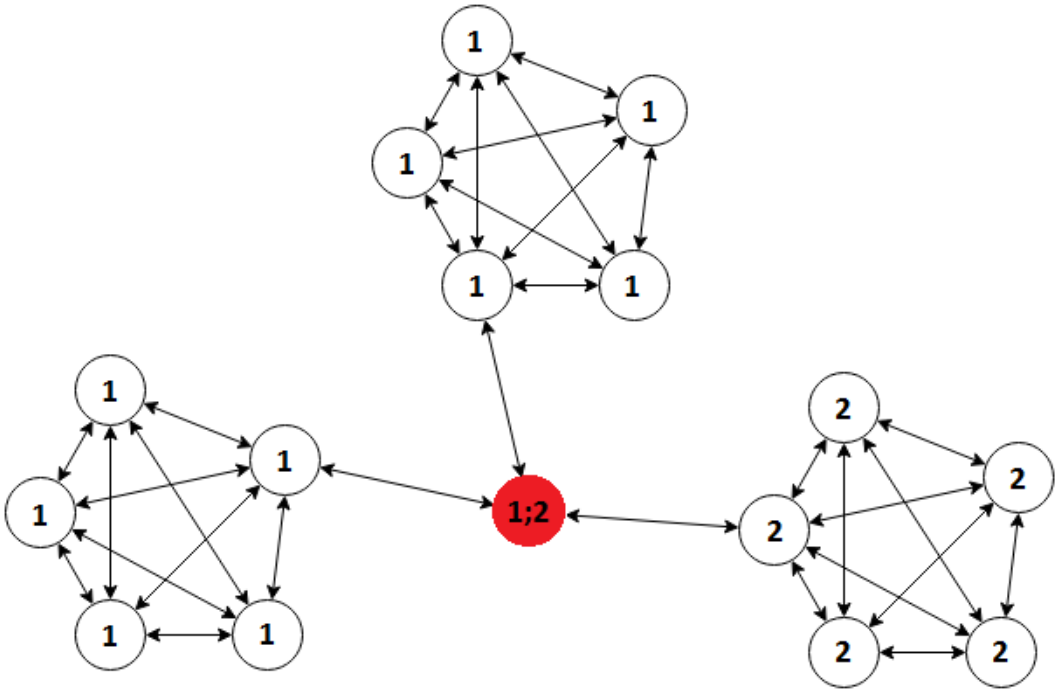

Figura 3.2: Exemplo do particionamento de um grafo, onde os limites ultrapassam 10%.

solução. Faremos menção ao vértice de maior grau como *maior_grau*. Em seguida os nós filhos e, suas respectivas sub-árvores, são visitados e designados a uma das partições. De acordo com o estudo realizado, foram constatados três possíveis casos, esses são descritos a seguir e ilustrados na figura 3.3.

1. Caso seja encontrada uma sub-árvore k que esteja dentro dos limites calculados, o grafo é dividido de forma que k pertença a uma partição e $\bar{k}$ a outra partição.

2. Caso uma sub-árvore de *maior_grau* possua uma quantidade de nós que exceda o limite superior, *maior_grau* passa a ser a raíz dessa sub-árvore e a busca pelo particionamento continua a partir desse ponto.

3. Caso o número de nós de todas as sub-árvores de *maior_grau* estejam abaixo do limite inferior estabelecido, cada uma das sub-árvores é colocada em uma das 2 partições seguindo a seguinte regra: se o número de nós da sub-árvore, acrescido do valor total em alguma das duas partições, estiver dentro do intervalo dos limites, a sub-árvore é então alocada nessa partição e a divisão é feita, caso contrário a sub-árvore será direcionada à partição com maior número de nós.

Se o terceiro caso ocorrer, é possível que um mesmo nó pertença a mais de uma

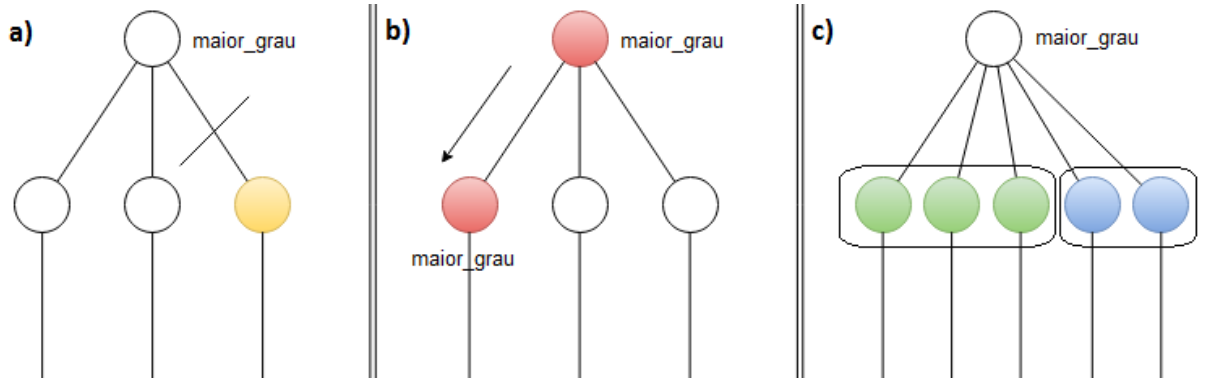

Figura 3.3: Exemplo dos 3 casos possíveis. a) caso 1, b) caso 2, c) caso 3.

partição. Temos que o vértice *maior_grau* é nó pai de todas as sub-árvores que foram alocadas para uma das partições, fazemos então com que ele pertença, simultaneamente, a ambas partições. Isso não é um problema tendo em vista que o algoritmo NetSCAN foi desenvolvido levando em consideração a ideia de multidisciplinaridade entre os pesquisadores, levando à existência de nós presentes em várias comunidades ou grupos, de forma concomitante.

A estratégia utilizada na implementação do algoritmo possibilita que o particionamento funcione apenas em divisões por potências de dois. Caso seja necessário um maior número de partições, o algoritmo deve ser reaplicado sucessivamente sobre as partições encontradas. Por exemplo, uma vez que temos duas partições, encontradas na primeira execução do particionamento, o algoritmo é aplicado novamente sobre cada uma das partições gerando quatro novas partições, e assim sucessivamente enquanto não atingir o número de partições desejado. Dessarte existe a possibilidade de trabalhar com valores na ordem de 2^n partições balanceadas.

3.1.1 Algoritmo de Particionamento

O algoritmo de particionamento foi implementado na linguagem C, utilizando a IDE CodeBlocks, e suas principais funções e procedimentos podem ser vistos no Algoritmo 4 e Algoritmo 5.

Algorithm 4 Principal

$grafo = PreProcessamento(dadosDeEntrada)$	▷ Leitura e análise dos dados
$AGM = Kruskal(grafo)$	▷ Encontra a AGM do grafo
$Particionar(AGM)$	▷ Particionamento da AGM

Algorithm 5 Particionar

Input: AGM, lim_sup, lim_inf
Output: partição1, partição2

```
23 Inicializa(vet1)                          ▷ Vetor que irá conter primeira partição
24 Inicializa(vet2)                          ▷ Vetor que irá conter segunda partição
25 particionado = false
26 i = 0
27 while !particionado do
28 |   NosPercorridos = BuscaProfundidade(NoDeMaiorGrau.filho[i])
29 |   if lim_inf <= NosPercorridos <= lim_sup then
30 |   |   corte(vet1, vet2)                  ▷ Particiona grafo em vet1 e vet2
31 |   else if NosPercorridos < lim_inf then
32 |   |   maiorQnt(vet1, vet2) = NosPercorridos   ▷ Aloca na maior partição
33 |   |   i = i + 1                          ▷ Atualiza índice da próxima sub-árvore
34 |   else if NosPercorridos > lim_sup then
35 |   |   inicializa(vet1, vet2)
36 |   |   NoDeMaiorGrau = NoDeMaiorGrau.filho[i]   ▷ Desce na sub-árvore
37 |   end if
38 end while
```

Explicando os procedimentos e os principais concentos envolvidos na implementação do algoritmo de particionamento, temos:

-*PreProcessamento*: Consiste na obtenção de informações importantes a respeito dos dados de entrada, que serão utilizadas ao longo de toda execução. Consiste, também, na modelagem dos dados através da utilização de uma biblioteca, implementada, de funções para manipulação de grafos.

A leitura dos dados inclui a realização da união de arestas que conectam os mesmos pares de nós. Como o grafo representa uma estrutura bidirecional, em muitos casos temos arestas entre os nós A e B, e, entre os nós B e A. Para resolver essa questão as duas arestas direcionadas são tranformadas em uma arestas não direcionada onde, o atributo peso da nova aresta é obtido através do cálculo da média entre as arestas eliminadas (3.3). A figura 3.4 exclarece como é realizada a operação.

Após a união de todos pares de arestas, entre os vértices presentes na base DBLP, o número total de arestas passou de 9915146 para 4957573.

$$\forall v_i, v_j \in V \Rightarrow e_{ij} = (e_{ij} + e_{ji})/2 \tag{3.3}$$

Uma árvore geradora mínima (AGM), figura 3.5, é qualquer árvore geradora de

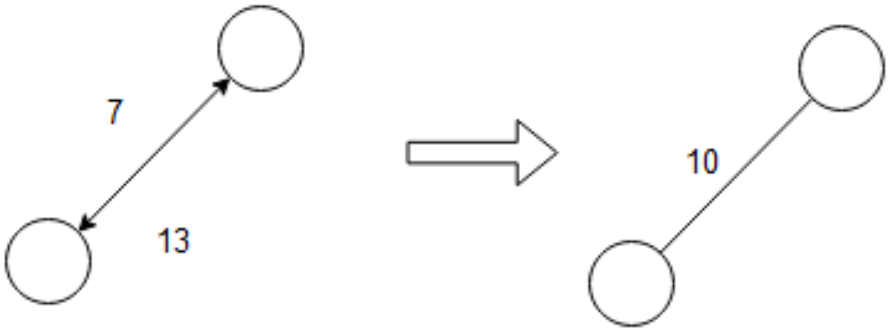

Figura 3.4: Processo da união de 2 arestas direcionadas.

um grafo, que possua a soma, de menor custo, do peso entre suas arestas.

Para o problema de particionamento, realizar o corte em um grafo, principalmente um grafo denso, seria uma tarefa de extremo custo e alta complexidade pertinente à necessidade do corte ter que ser feito em várias arestas ao longo de todo o grafo e, garantir que, todas as arestas entre as partições resultantes tenham sido retiradas. Uma estratégia, nesse caso, seria distribuir os vértices para diferentes partições de forma alternada e, no final, realizar a união dos vértices que eram conectados e estão na mesma partição. Nesse panorama, o custo computacional seria imensurável devido à necessidade de realizar a busca na estrutura do grafo e saber se, para cada par de vértices v_i e v_j, eles possuíam conexão antes do corte. A partir dessa e outras considerações foi decidido que o melhor caminho seria a utilização de uma AGM para o particionamento do conjunto, uma vez que é necessário apenas que um corte seja realizado para que o grafo seja dividido em duas partições.

Existem dois algoritmos clássicos na literatura que foram considerados para ser utilizado no cálculo da AGM, Kruskal (KRUSKAL, 1956) e Prim (PRIM, 1957). Prim executa na ordem de O(E + V log V) e Kruskal em O(E log V). Prim é mais eficiente em grafos de alta densidade, onde a ordem do número de arestas é muito maior em relação ao número de nós. Kruskal é considerado melhor para grafos esparços. Considerando a base de dados, após a união das arestas, temos que o número de arestas é quase da mesma ordem que o número de nós, sendo quatro vezes maior. Logo, o algoritmo escolhido a ser utilizado no cálculo da AGM foi o algoritmo de Kruskal. Além disso, como se trata de uma AGM, ao particionarmos um conjunto garantimos que o corte ocorra sobre uma aresta de peso baixo, existindo uma menor chance de ocorrer a separação de relações de

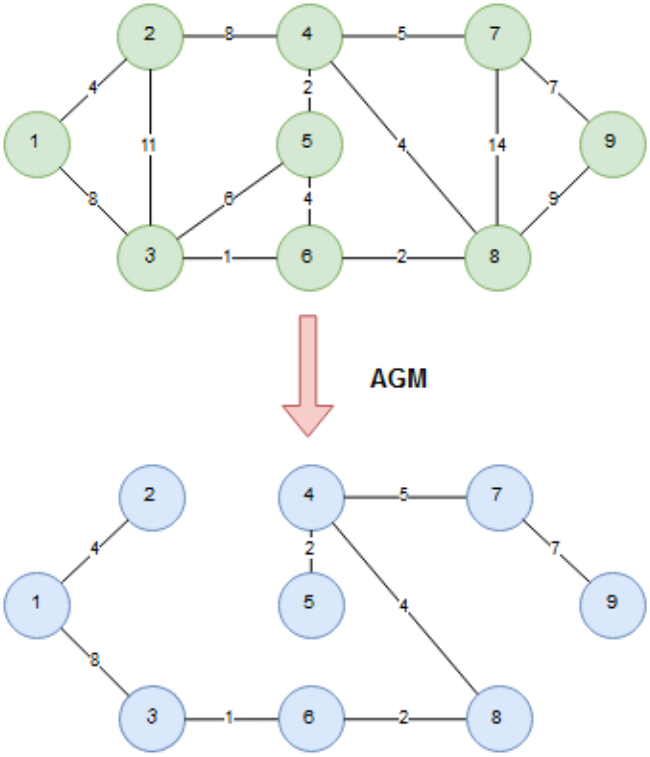

Figura 3.5: Exemplo de um grafo e sua respectiva AGM.

grande influência entre pesquisadores da rede.

-*Kruskal*: O algoritmo de Kruskal possui uma abordagem gulosa e foi implementado de acordo com (KRUSKAL, 1956), onde os passos são:

1. Ordenar as arestas do grafo pelo seu peso.

2. Selecionar a aresta de menor peso, verificando se a inserção da mesma no conjunto solução causa a formação de ciclo, caso forme a mesma será descartada, senão ela é incluida no conjunto solução.

3. Repetir o passo 2 até que o conjunto solução possua $(n-1)$ arestas, garantindo a existência de uma árvore geradora mínima.

Através da aplicação do algoritmo Kruskal, foi constatada a existência de diversas componentes conexas no banco. Foi então desenvolvido uma função para encontrar a maior componente conexa, para que apenas esse subconjunto fosse utilizado no estudo. Após a execução do algoritmo, a AGM encontrada, referente à maior componente conexa do grafo, possui 1100505 vértices e 1100504 arestas.

Particionamento: A função busca dividir o conjunto de dados de forma balanceada considerando dois limites pré estabelecidos e calculados. As sub-árvores do grafo são escalonadas em dois vetores considerando a quantidade de nós que possuem. Se a sub-árvore em questão possuir um número de nós maior do que o limite superior, descemos o nível na árvore, caso a sub-árvore esteja dentro do limite, o corte é realizado. No caso onde as sub-árvores possuem uma quantidade de nós inferior ao limite inferior, elas são divididas em um dos dois grupos, seguindo a estratégia já descrita no início dessa seção. Uma propriedade importante é que, nesse último caso, o nó pai das sub-árvores será inserido em ambas partições.

3.2 PNetSCAN

O contexto ideal para aplicação do NetSCAN paralelizado consiste em um cenário de integração onde, após a execução do algoritmo de particionamento, o NetSCAN seja chamado, em processos distintos, para cada partição encontrada. A execução do agrupamento nas partições pode ser interpretado como uma fila de processos onde, a cada dois resultados obtidos, é chamada a operação de *merge* para os grupos formados por esses resultados. Novamente, os resultados obtidos pelas operações de *merge* são colocados em uma outra fila e o *merge* é aplicado, de forma concorrente, nos resultados obtidos. As operações continuam até que não existam mais resultados parciais e tenha-se o resultado final do agrupamento, proveniente das operações de *merge*. Um exemplo do modelo pode ser visto na figura 3.6 abaixo.

A estratégia utilizada no desenvolvimento da operação de *merge* consiste em reaplicar o agrupamento sobre os elementos que sofreram o corte da aresta durante a etapa de particionamento. A concepção da ideia considera a baixa influência que, vértices distantes dos vértices de fronteira[5], possuem nas alterações geradas pela divisão do grafo, além do fato de que a eliminação de várias relações durante o corte interfere diretamente na computação dos grupos formados.

Foi observado que, após a execução do processo de agrupamento sobre cada partição, algumas dessas partições mantiveram grupos originalmente formados enquanto

[5]Vértices que tiveram a aresta removida durante a etapa de particionamento.

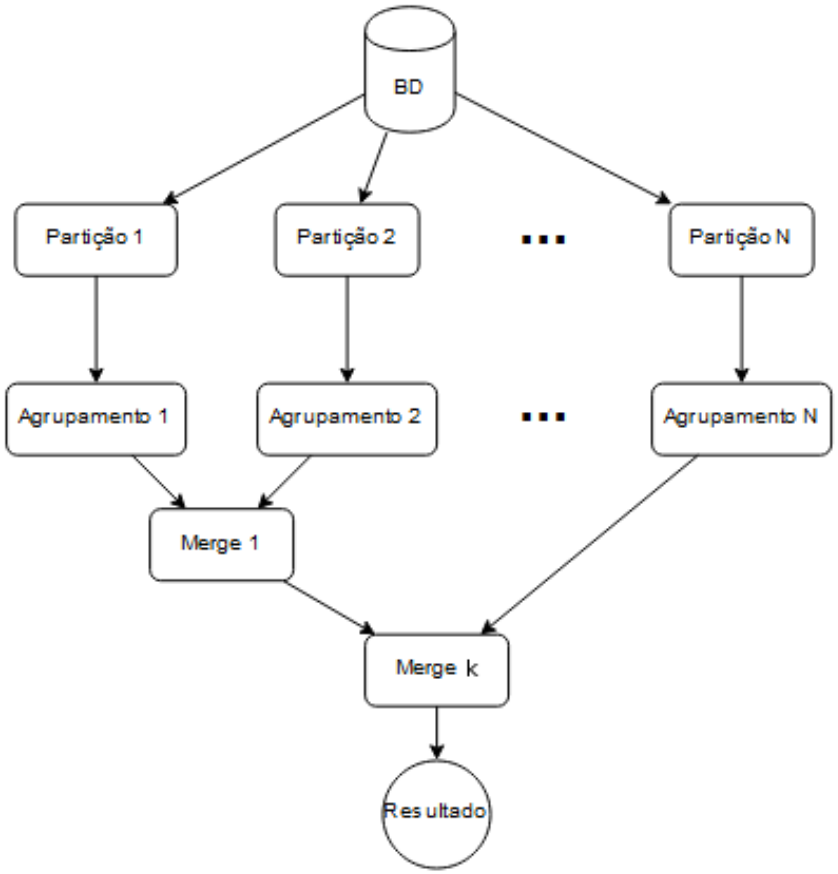

Figura 3.6: Exemplo do método de paralelismo proposto.

que outros grupos desapareceram, gerando novos nós *noise* e/ou outros grupos menores. O método foi testado e a validação foi realizada através de testes em um conjuntos de instância menores, onde foi observado todo o processo de agrupamento das partes, para que pudesse ser analisado o comportamento dos nós durante a execução do procedimento.

As figuras 3.7 e 3.8, abaixo, demonstram a diferença entre os processos de agrupamento sobre uma base inteira e o agrupamento sobre cada uma das partições dessa base, após o particionamento. Na figura 3.7 temos grupo1 = {1,2,3,4,5,6,7,8}, grupo2 = {8,9,10,11,12,13,14} e grupo3 = {9,15,16,17,18,19,20,21}. Os vértices 8 e 9 pertencem a mais de um grupo. Ao realizar o particionamento sobre o conjunto de vértices, na primeira partição, figura 3.8, foi observado a criação de um novo grupo4 = {1,3,6,9,15,16,17} e os nós 18, 19 e 20 foram marcados como *noise*. Na segunda partição foi encontrado um grupo5 = {2,4,5,7,8,9,10} e os nós 11, 12, 13, 14 e 21 foram marcados como *noise*. Esse é um exemplo de como os nós se comportam diante o procedimento de particionamento, seguido pelo agrupamento, onde é possível observar a criação de novos grupos, distintos dos grupos originais e, principalmente, a existência de vértices que não foram classificados

em nenhum grupo. A aplicação da operação de (*merge*) pode ser vista, também, na figura 3.7, onde os nós em vermelhos são os vértices que foram desmarcados, em decorrência do fato de que foram os vértices que tiveram relações eliminadas durante a etapa de particionamento. Esse vértices foram submetidos, novamente, à aplicação do agrupamento após serem desmarcados. Por fim, concluído todas as etapas do processo, foi observado que o algoritmo conseguiu encontrar o mesmo resultado obtido no agrupamento da base inteira. É válido observar, para esse exemplo, que os nós desmarcados compõem um grande percentual do total de vértices presentes, isso se deve ao baixo número de vértices na instância, utilizada como exemplo explicativo. A quantidade de nós desmarcados diminui significativamente para bases maiores. As relações, indicadas pelas arestas, não foram representadas nas figuras para uma melhor visualização desse exemplo.

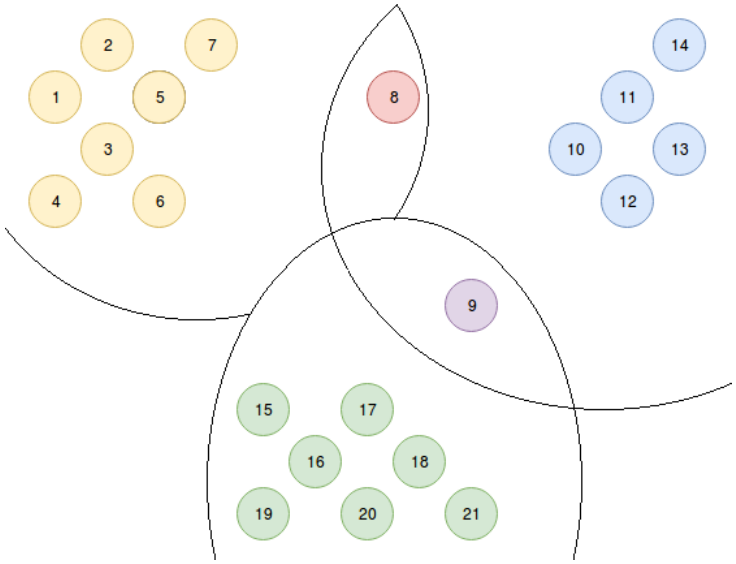

Figura 3.7: Agrupamento sobre uma base inteira.

Durante a etapa de agrupamento, o algoritmo NetSCAN realiza marcações nos nós que são agrupados, para que os mesmos não sejam percorridos novamente durante o processo de agrupamento. Em cada etapa de particionamento um arquivo contendo a identificação das arestas removidas é criado a fim de ser utilizado na etapa de *merge*. Através da leitura desse arquivo é possível identificar os nós que tiveram suas arestas removidas durante o particionamento; eles são desmarcados[6] e o NetSCAN é aplicado novamente sobre esse conjunto, com a diferença que o algoritmo irá considerar apenas os nós desmarcados nessa etapa.

[6] A desmarcação é feita removendo os atributos *noise*, *expanded* e *core* do vértice.

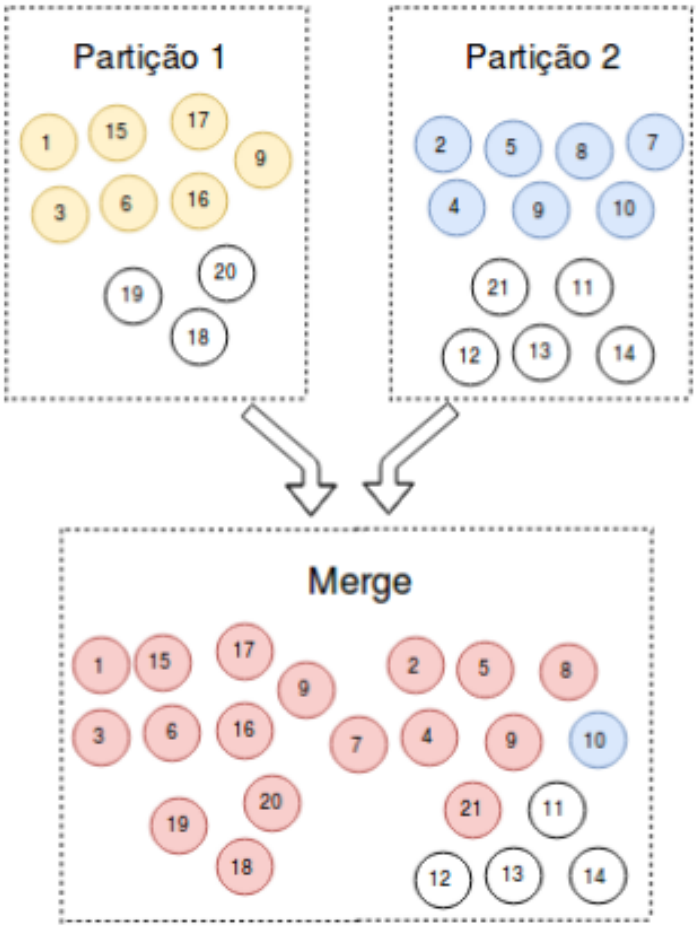

Figura 3.8: Agrupamento sobre as duas partições da base e a realização do *merge*.

O algoritmo 6 demonstra a implementação da função *merge* sobre duas partições.

Algorithm 6 Merge

Input: grafo,particao1,particao2,arestasRemovidas
Output: resultadoParcial

39 $nos = buscaNos(arestasRemovidas)$ ▷ Busca nós que tiveram suas arestas removidas
40 $Desmarca(nos, grafo)$ ▷ Elimina atributos classificadores
41 $Conecta(nos, arestasRemovidas)$ ▷ Conecta os nós que haviam sido desconectados
42 $NetSCAN(grafo)$ ▷ O algoritmo é aplicado novamente

O algoritmo foi desenvolvido em python e incorporado ao Neo4j através da biblioteca py2neo e da instalação do respectivo neo4j-driver, responsável pela integração. O algoritmo consiste na implementação de uma fila de partições, chamada através de threads, e sua execução realizada de forma paralela. As queries, em cypher[7], são responsáveis pelas operações no Neo4j e, são chamadas ao longo do código para realizar os diversos tipos manipulações no banco. Todo o algoritmo, associado, desenvolvido nesse trabalho,

[7]Linguagem Query do NEO4j, inspirada em SQL, para descrever padrões em grafos.

pode ser visto no algoritmo 7.

Algorithm 7 Algoritmo Final

Input: numeroParticoes, dados

```
*particao = particiona(dados, numeroParticoes)        ▷ Chama a função de particionamento
while i < numeroParticoes do
    NETSCAN(particao[i++])   ▷ Chama o agrupamento para cada uma da partições
end while
while NETSCAN(*results) == executando do
                                        ▷ Enquanto existir algum processo de agrupamento
    if NETSCAN(result[i]) == pronto then
        adicionaListaMerge(lista, agrupado[i])
    end if
    if tamanhoLista() ≥ 2 then
        MERGE(desempilha(2, lista))      ▷ Desempilha 2 resultados da pilha e realiza a união
    end if
end while
```

4 Experimentos e Resultados

Todos os os testes foram realizados em uma máquina com processador i5 - 4th geração, 4mb de memória ram e sistema operacional Ubuntu 14.04. Para os testes, a execução da etapa de agrupamento foi realizada de forma sequencial e, como forma de análise, os valores dos tempos, para cada partição, foram salvos e comparados para que pudesse ser contabilizado apenas o maior tempo, a fim de considerar um cenário paralelo.

4.1 Análise de tempo

Os valores utilizados para comparação e análise do tempo total de processamento, foram:

1. Tempo total de particionamento.
2. Tempo de aplicação do NetSCAN sobre as partições.
3. Tempo de *merge*.

Durante a etapa de agrupamento em paralelo é observado o tempo de aplicação do NetSCAN e, para cada partição, o maior tempo é levado em consideração na hora da soma dos valores. Seja T o conjunto contendo o tempo de execução do agrupamento em todas partições e, t_i o tempo de execução do agrupamento na partição i, apenas t_k será contabilizado no tempo total da aplicação do algoritmo em paralelo, de acordo com (4.1).

$$\forall t_i \in T, t_k \geq t_i \tag{4.1}$$

O método de análise de tempo é exemplificado pela figura 4.1, onde os maiores valores de tempo, para cada processo de particionamento das etapas 1, 2 e 3, são escolhidos e somados aos maiores valores, para cada processo de *merge* das etapas 4, 5 e 6, a fim de obter o valor total para o tempo de execução.

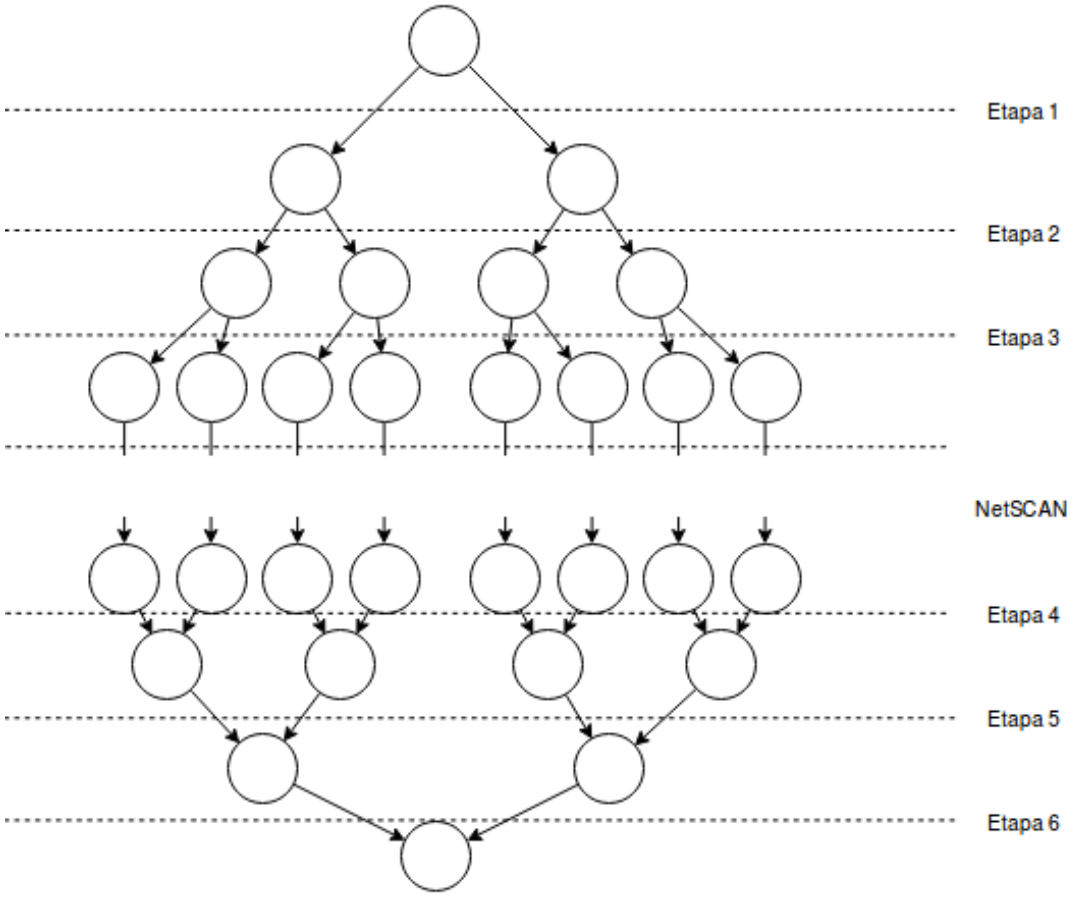

Figura 4.1: Etapas do processo de paralelismo para oito partições.

4.2 Validação

Bases com poucos vértices foram utizadas com o objetivo principal de validação do método proposto, em especial a operação *merge*, e, estudo de resultados. Quatro bases foram escolhidas nessa etapa. Essas são:

- *200-data*, uma instância gerada em (SEMAAN, 2013) que representa pontos em uma distribuição normal em R^2. Também foi utilizada em (HORTA, 2017), como experimento avaliativo em R^2, para verificar a capacidade do algoritmo na detecção de grupos. A rede possui um total de 200 vértices e 40000 relações. A instância pode ser visualizada na figura 4.2.

- *Artificial*, uma rede sintética gerada em (HORTA, 2017) para possuir três comunidades com a sobreposição de um nós central, que deve pertencer às três comunidades simultaneamente. A rede possui 16 vértices e 84 relações. A instância pode ser visualizada na figura 4.3.

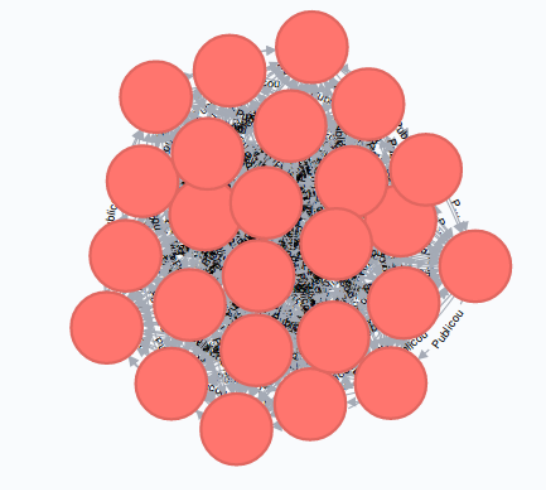

Figura 4.2: Instância 200-data.

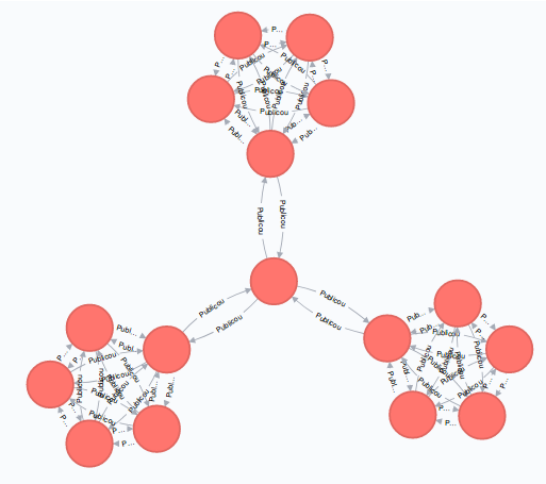

Figura 4.3: Instância Artificial.

- *Karatê*, Zacharys Karate Club Network (ZACHARY, 1977), uma rede real conhecida da literatura que representa as relações de amizade entre 34 membros de um clube de karatê. A rede possui 34 vértices e 347 relações. A instância pode ser visualizada na figura 4.4.

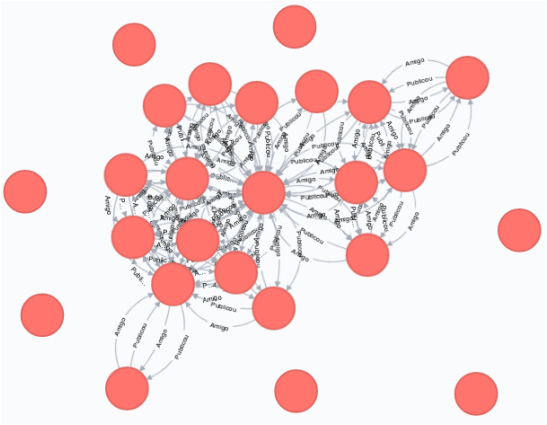

Figura 4.4: Instância Karatê.

- *Proteins*, uma rede real de proteínas (MEENA; DEVI, 2015) que possui três grupos distintos e dois nós sobrepostos. A rede possui 21 vértices e 143 relações. A instância pode ser visualizada na figura 4.5.

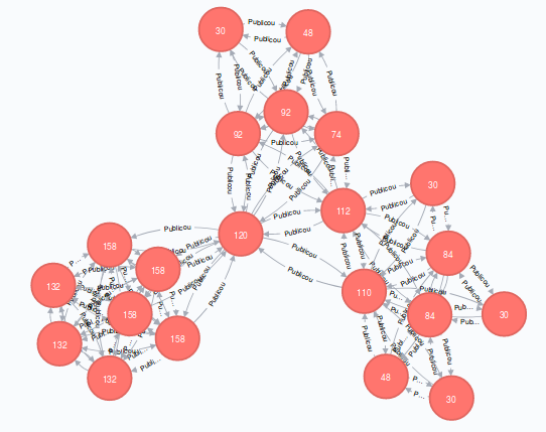

Figura 4.5: Instância Proteins.

Os resultados obtidos pelo PNetSCAN foram os mesmos obtidos pelo agrupamento sobre a base inteira. Em vista disso, o método de união (*merge*) proposto, se mostrou eficaz e, através dos testes preliminares realizados, comprovou ser válido. Nas tabelas 4.1 e 4.2 estão os resultados de tempo obtidos, a respeito das instâncias de validação, como primeira análise de desempenho do método de paralelismo proposto.

Embora instâncias menores possuam utilidade na validação da proposta, é possível observar que não houve ganho na aplicação do paralelismo para essas instâncias. Pelo contrário, houve um aumento no tempo de execução para todas as quatro instâncias utilizadas. Esses resultados demonstram que, para instâncias de pequeno porte, a abordagem de paralelismo não apresenta nenhum ganho no que se refere ao tempo de processamento.

Tempos de agrupamento sem paralelismo	
Instância	Agrupamento
Karatê	2,672 s
Proteins	2,332 s
Artificial	2,099 s
200data	59,541 s

Tabela 4.1: Análise do tempo de agrupamento para instâncias de validação, sobre a base inteira (sem particionar).

Tempos de agrupamento com paralelismo			
Instância	Particionamento	Agrupamento Paralelo	Total
Karatê	5,376 s	2,893 s	8,269 s
Proteins	5,612 s	2,54 s	8,152 s
Artificial	3,783 s	2,07 s	5,853 s
200data	17,156 s	68,961 s	86,117 s

Tabela 4.2: Análise do tempo de agrupamento para instâncias de validação, utilizando o particionamento da base.

4.3 Rede Científica

Após a conclusão da validação do método proposto, em especial a etapa de união dos resultados parciais (*merge*), experimentações com a base DBLP foram realizadas com o propósito de verificação da viabilidade da aplicação do NetSCAN para bases de grande porte, aplicando-se uma abordagem paralela.

A principal instância utilizada no desenvolvimento desse trabalho é a rede social científica de pesquisadores da DBLP (LEY, 2002), previamente utilizada e armazenada em um modelo orientado a grafo por (HORTA, 2017). O conjunto de dados representa uma rede de relacionamento onde é possível, através de técnicas de agrupamento, extrair informações a respeito de comunidades existentes nos diversos nichos científicos presentes na base. O arquivo da rede pôde ser modelado como um grafo bidirecionado $G = \{V, E\}$ onde os nós $V = \{v_1, v_2, ..., v_n\}$ representam os pesquisadores e o conjunto de arestas $E_{ij} = \{V_i, V_j\}$ representa as relações presentes entre os nós V_i e V_j, com $0 \leq i \leq n$ e $0 \leq j \leq n$.

É importante observar que, como o grafo é bidirecionado, temos que o peso não é necessariamente o mesmo em arestas com diferentes sentidos, sendo $peso(e_{ij}) \neq peso(e_{ji})$. O peso indica a relação de influência entre um pesquisador e outro e, pode ser utilizado para definir o grau de influência entre os pesquisadores (HORTA, 2017). Essa relação pode ser utilizada para identificar pesquisadores influentes e influenciados, além de outras características como, àqueles que possuem muita citação e, como resultado, apresentam várias arestas de saída, ou inclusive àqueles que publicam em várias áreas e, consequentemente, são influenciados por diversas fontes, levando à existência de várias arestas de incidência. A direção da aresta indica a influência que um pesquisador exerce sobre outro.

DBLP (LEY, 2002) é uma base de grande porte onde, através de seu estudo, foi possível identificar 1243022 vértices de Ids distintos, sendo 1306562 o maior Id existente. A base também inclui 9915146 conexões; relações científicas entre pesquisadores. Essa mesma base foi utilizada em (HORTA, 2017) onde foi modelada e, NetSCAN, uma técnica de agrupamento, foi desenvolvida e utilizada a fim de ser possível identificar comunidades científicas, subgrupos de pesquisa e pesquisadores multidisciplinares, incluindo seu nível de influência, dentro da rede. Nesse trabalho o estudo não foi realizado sobre a base completa, apenas partições retiradas da maior componente conexa da base foram utilizadas nessa etapa do andamento.

A base DBLP possui uma quantidade muito grande de vértices e relações, por esse motivo, conjuntos de dados menores foram retirados da maior componente conexa da base, a fim de serem utilizados para os testes conclusivos desse trabalho. O número de vértices, assim como os tempos obtidos, para cada uma das instâncias utilizadas, são apresentados nas tabelas 4.3, 4.4, 4.5 e 4.6.

Tempos de agrupamento das instâncias DBLP	
Instância	Agrupamento
DBLP-236	8,96218204498 s
DBLP-947	120,193242073 s
DBLP-1911	453,327996969 s (aprox.7,55 min)
DBLP-4617	2069,53085208 s (aprox.34,4 min)
DBLP-9131	6844,29151416 s (aprox.1,90 hrs)
DBLP-18098	29393,1737521 s (aprox.8,16 hrs)

Tabela 4.3: Análise do tempo de agrupamento para instâncias retiradas da base DBLP, sobre a base inteira (sem particionar).

Tempos de agrupamento das instâncias DBLP				
Instância	Particionamento	Agrupamento Paralelo	Total	melhora
DBLP-236	2,723 s	12,985758066 s	15,708758066 s	-75%
DBLP-947	3,871 s	110,28222394 s	114,15322394 s	+5%
DBLP-1911	5,516 s	371,729707003 s	377,245707003 s	+16%
DBLP-4617	12,335 s	1141,002952099 s	1153,337952099 s	+44%
DBLP-9131	24,884 s	5177,30083489 s	5202,18483489 s	+23%
DBLP-18098	68,321 s	25007,11244992 s	25075,43344992 s	+14%

Tabela 4.4: Análise de tempo das instâncias geradas, sobre a base particionada em dois.

Tempos de agrupamento das instâncias DBLP				
Instância	Particionamento	Agrupamento Paralelo	Total	Melhora
DBLP-236	7,899 s	14,577332973 s	22,476332973 s	-150%
DBLP-947	8,18 s	81,795166016 s	89,975166016 s	+25%
DBLP-1911	12,17 s	208,376779079 s	220,546779079 s	+51%
DBLP-4617	20,268 s	1167,686871052 s	1187,954871052 s	+42%
DBLP-9131	38,033 s	4608,371167892 s	4646,404167892 s	+32%
DBLP-18098	100,315 s	21217,9403048 s	21318,2553048 s	+27%

Tabela 4.5: Análise de tempo das instâncias geradas, sobre a base particionada em quatro.

Tempos de agrupamento das instâncias DBLP				
Instância	Particionamento	Agrupamento Paralelo	Total	Melhora
DBLP-236	11,008 s	21,882627964 s	32,890627964 s	-266%
DBLP-947	11,531 s	108,234939337 s	119,765939337 s	+3%
DBLP-1911	12,17 s	207,943732977 s	220,113732977 s	+51%
DBLP-4617	20,268 s	1317,747081757 s	1338,015081757 s	+35%
DBLP-9131	43,558 s	4383,762000081 s	4427,320000081 s	+35%
DBLP-18098	109,465 s	15615,039506894 s	15724,504506894 s	+46%

Tabela 4.6: Análise de tempo das instâncias geradas, sobre a base particionada em oito.

O algoritmo de particionamento desenvolvido foi, também, utilizado no particionamento da base DBLP para geração de instâncias menores no que tange à sua utilização nos experimentos desse trabalho. O tamanho de cada instância corresponde a 236, 947, 1911, 4617, 9131 e 18098, nessa ordem.

A tabela 4.3 demonstra os resultados obtidos na aplicação do algoritmo NetSCAN sobre cada uma das bases, sem a realização do particionamento. Os resultados obtidos são comparados com os resultados referentes às bases particionadas, sendo realizada a comparação de tempo entre o NetSCAN e o PNetSCAN. O percentual de melhora, apresentado na última coluna das tabelas 4.4, 4.5 e 4.6, indica a melhora obtida em relação à aplicação do agrupamento sobre a base inteira.

4.3.1 Análise dos resultados

É possível observar que cada instância representa um grafo com suas particularidades onde, de acordo com sua configuração, existe um número ótimo de partições cujo o tempo de execução obtém sua melhora máxima. Caso o número de partições aumente, o valor de tempo retoma o crescimento. Observando os gráficos de "tempo X partições", A.1, A.2,

A.3, A.4, A.5 e A.6, é possível observar o comportamento do valor do tempo de processamento diante da alteração do número de partições para cada uma das instâncias. Esses gráficos foram construído a fim de proporcionar uma melhor visualização dos resultados no que se refere à evolução do tempo conforme o aumento do número de partições. Possibilitando, também, indicar um ponto onde o valor do tempo não obtém mais melhora e começa a crescer.

O gráfico A.1 (figura 4.6) indica um comportamento de crescimento onde, o menor valor de tempo de processamento obtido provém da aplicação do algoritmo sobre a base inteira. Conforme a base é dividida, é possível observar um aumento no custo de processamento. A quantidade de vértices presentes na instância faz com que o esforço de paralelizar o processo ultrapasse o tempo agrupamento na base inteira.

Os gráfico A.2 (figura 4.7) apresenta uma situação de melhoria do tempo conforme número de partições cresce, seguido de uma ligeira retomada no crescimento no tempo, para mais de quatro partições.

O gráfico A.3 (figura 4.8) apresenta uma queda no tempo de processamento, até a divisão em quatro partições, e esse valor se mantém conforme o número de partições cresce.

O comportamento do gráfico A.4 (figura 4.9) se assemelha ao gráfico A.2 (figura 4.6), onde existe a situação de melhoria do tempo conforme número de partições cresce, seguido da retomada no crescimento no tempo, para mais de duas partições.

As instâncias com a maior quantidade de vértices, gráficos A.5 e A.6 (figuras 4.10 e 4.11), possuem melhora no tempo de processamento conforme o número de partições é incrementado. Esse resultado indica que um maior número de vértices pode representar um cenário onde, o tempo de execução continua a melhorar de acordo com o número de partições.

A tabela 4.7 condensa todos os resultados e apresenta o melhor valor para o número de partições, obtidos em cada instância utilizada, e o percentual máximo de melhora no tempo de processamento, comparado com o tempo de execução do agrupamento sem utilizar o particionamento.

Com base nos resultados obtidos é possível responder à questão de pesquisa deste

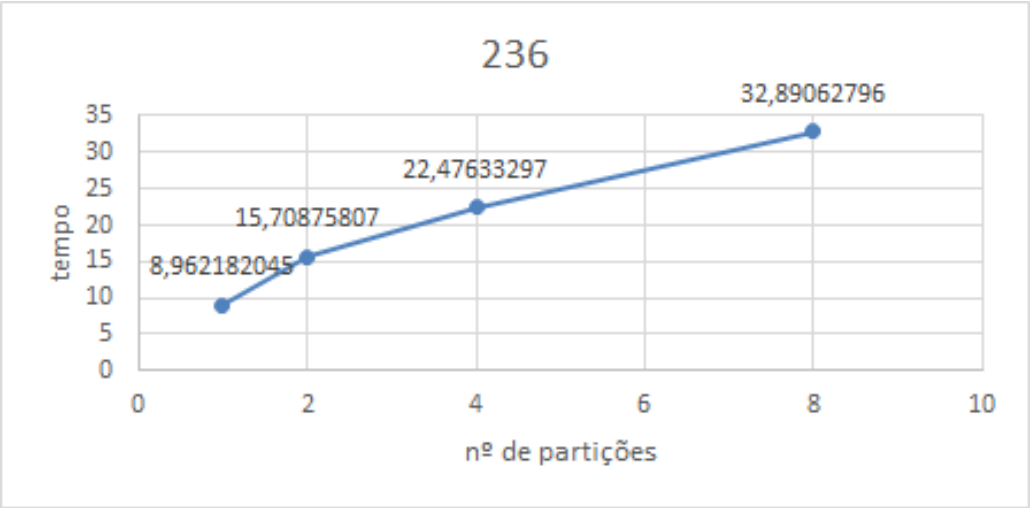

Figura 4.6: Gráfico "tempo X partições"da instância 236.

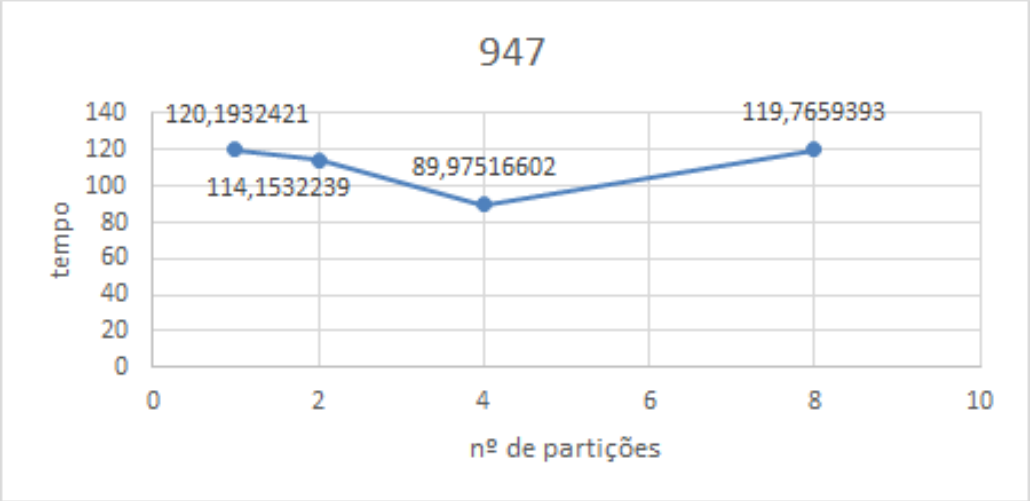

Figura 4.7: Gráfico "tempo X partições"da instância 947.

trabalho: "*É possível obter uma melhora no tempo de processamento, tendo em vista a existência de uma heurística de paralelismo válida, para o problema de agrupamento de dados?*". É possível concluir que existe, sim, uma melhora considerável no tempo de processamento do algoritmo NetSCAN. Exceto pelas instâncias de validação e pela primeira instância gerada, contendo 236 nós, as outras instâncias obtiveram uma melhora de, pelo menos, 25% em relação ao tempo de processamento original, obtido pela execução do algoritmo na base inteira. Se comparado com os resultados obtidos pela aplicação do NetSCAN na base inteira, o PNetSCAN, aplicado ao maior subconjunto utilizado, DBLP-18098, particionado em oito, obteve uma redução de 8,16 para 4,36 horas, uma melhora de aproximadamente 3,8 horas no tempo de execução.

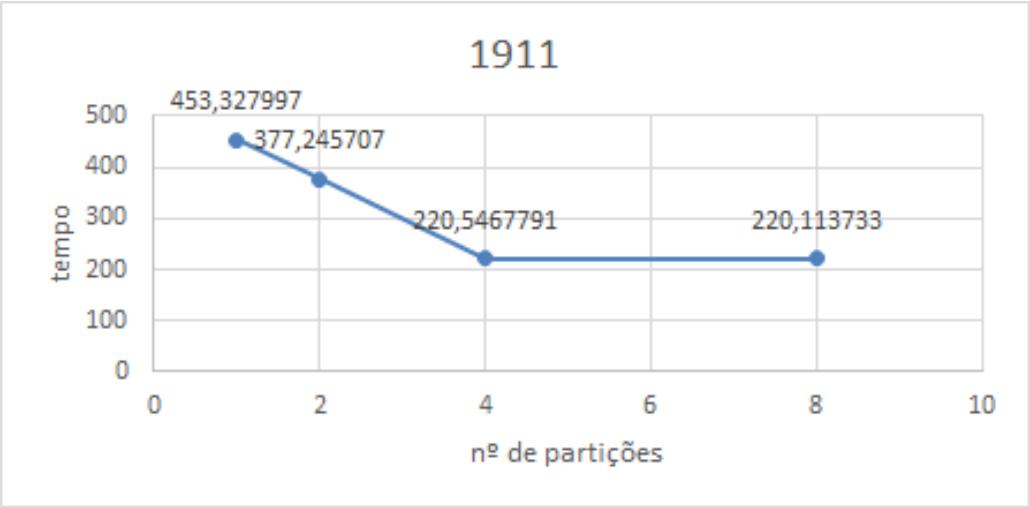

Figura 4.8: Gráfico "tempo X partições"da instância 1911.

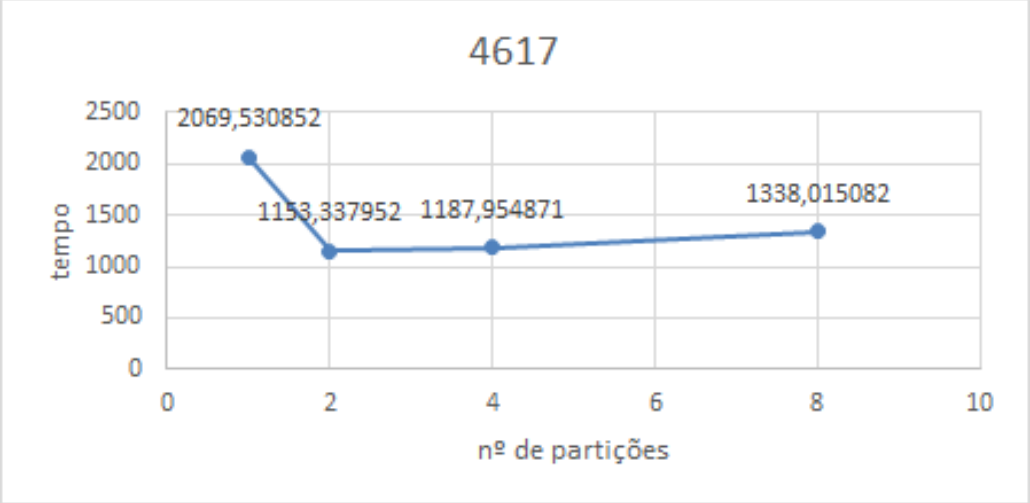

Figura 4.9: Gráfico "tempo X partições"da instância 4617.

Ainda, uma pesquisa foi realizada afim de entender o motivo pelo qual o tempo de processamento obtém melhora, apenas, até o particionamento por dois, na instância DBLP-4617. Como os valores considerados para descrever cada instância estão relacionados à quantidade de vértices, a hipótese é que o número de arestas pode estar relacionado ao esforço do algoritmo na realização dos cálculos necessários. Foi, então, levado em consideração o número de arestas presentes em cada instância e realizado o cálculo de densidade em cada uma das instâncias para entender seu efeito.

A densidade de um grafo pode ser definida pelas equações (4.2) e (4.3) (KOWALIK, 2006), onde V é o número de vértices e E o número de arestas no grafo. A equação (4.2) é referente ao cálculo da densidade, relacionado a um grafo não direcionado, en-

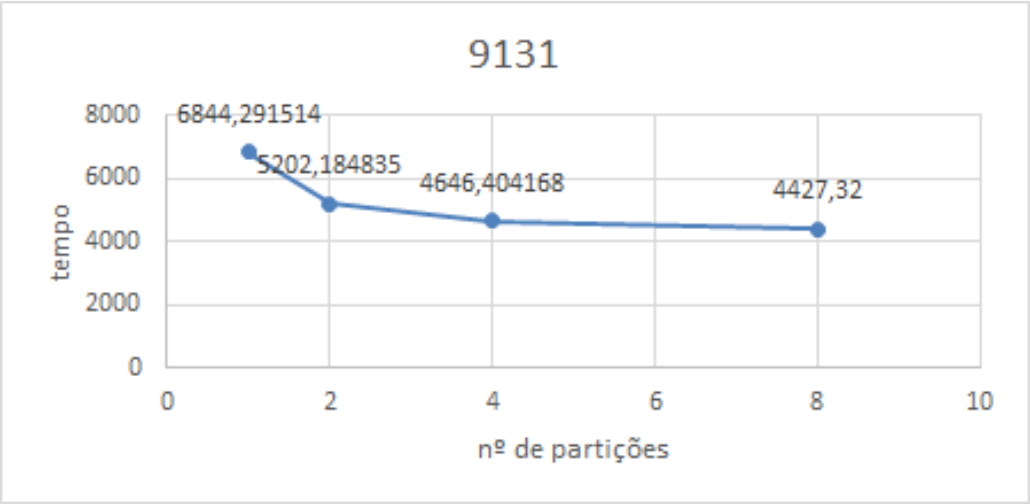

Figura 4.10: Gráfico "tempo X partições"da instância 9131.

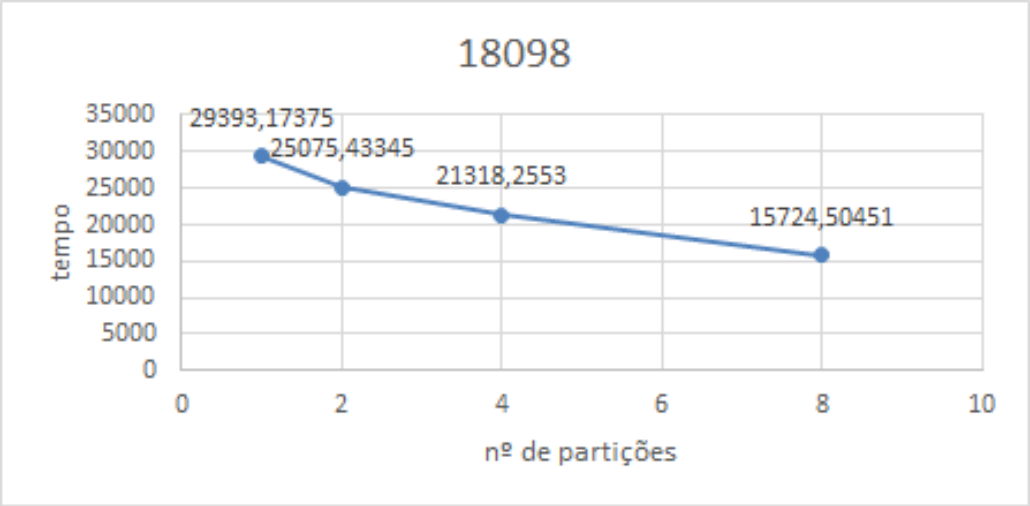

Figura 4.11: Gráfico "tempo X partições"da instância 18098.

quanto que equação (4.3), utilizada, é empregada para o cálculo de densidade em grafos direcionados. O valor de densidade indica a razão do número de arestas presente no grafo pelo número máximo de arestas que o grafo pode ter. A densidade em um grafo pode variar entre os valores 0 e 1.

$$D = \frac{2 \mid E \mid}{\mid V \mid (\mid V \mid -1)} \tag{4.2}$$

$$D = \frac{\mid E \mid}{\mid V \mid (\mid V \mid -1)} \tag{4.3}$$

Número ótimo de partições		
Instância	Partições	% de melhora
DBLP-236	1	-
DBLP-947	4	25%
DBLP-1911	8	51%
DBLP-4617	2	44%
DBLP-9131	8	35%
DBLP-18098	8	46%

Tabela 4.7: Melhor valor do número de partições de cada intância.

A tabela 4.8 apresenta as densidades obtidas para cada uma das instâncias. Pode-se observar que os valores são baixos para todas as instâncias e, diminuem conforme aumenta o tamanho da instância. Com esses resultados é possível concluir que a instância DBLP-4617 atingiu sua melhor configuração, relacionada ao desempenho da aplicação do PNetSCAN, com o número de partições igual a dois. Além do mais, é possível complementar que, para as instâncias adotadas nesse trabalho, a densidade não influencia na aplicação do método proposto.

Densidade das instâncias	
Instância	Densidade
DBLP-236	0,031193653
DBLP-947	0,009856429
DBLP-1911	0,004786836
DBLP-4617	0,0016112
DBLP-9131	0,000759638
DBLP-18098	0,000409075

Tabela 4.8: Valor de densidade para cada instância.

5 Considerações Finais

Nesse trabalho foi proposto o PNetSCAN, um processo de particionamento e paralelismo para execução do algoritmo NetSCAN, a fim de viabilizar o tempo de processamento para bases de grande ponte. No algoritmo apresentado, operações de particionamento e *merge* foram o foco do desenvolvimento e obtiveram êxito quanto ao seu propósito.

Uma estratégia de paralelismo foi definida e planejada. Com base na estratégia proposta, foi desenvolvido um algoritmo para o particionamento balanceado em grafos, que apresentou bom funcionamentos nas avaliações realizadas. A aplicação do método teve como alvo o algoritmo de agrupamento NetSCAN, testes foram conduzidos a fim de validar o procedimento, em seguida, análises comparativa entre resultados, tanto obtidos através da aplicação do NetSCAN sobre a base inteira, quanto na base particionada em diferentes número de partições, foram realizadas a fim de testar a eficiência da estratégia. Foi possível, através dos dados obtidos, realizar um estudo do comportamento do tempo de processamento em relação ao número de partições selecionadas. Um valor ótimo para o número de partições pôde ser identificado observando o comportamento da curva de "tempo X partições", apresentada nos gráficos dos resultados.

Foram realizados extensivos testes a fim de validar todas as implementações, e, testes de desempenho foram realizados utilizando componentes menores da base de dados dos pesquisadores da DBLP. Os experimentos comprovam o que muitos artigos já indicavam, uma melhora promissora no tempo de execução do algoritmo, se empregado a ideia de paralelismo.

5.1 Trabalhos Futuros

Na atual etapa do desenvolvimento foram utilizadas as primeiras instâncias geradas, com uma quantidade de vértices, que ainda não representam a base em sua totalidade. Todavia, para trabalhos futuros, a proposta é que seja utilizada a base DBLP de forma completa, bem como outras bases de grande porte, a fim de continuar a observação de

como se comporta a relação entre o número de partições e o tempo de processamento.

Como o NetSCAN está disponível como uma rotina do Neo4j, pretendemos estender a proposta deste trabalho para que o PNetSCAN seja executado considerando várias instâncias do Neo4j executadas em paralelo, seja em máquinas diferentes ou na mesma máquina utilizando Threads diferentes.

Além disso, pretende-se aprimorar o algoritmo de particionamento e de *merge* para melhorar a performance do processo como um todo, reduzindo ainda mais o tempo de processamento em bases de dados de grande porte.

Bibliografia

ALMEIDA, R. et al. Recomendação de recursos educacionais para grupos: buscando soluções em redes sociais. In: *Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação-SBIE)*. [S.l.: s.n.], 2016. v. 27, n. 1, p. 996.

ANDREEV, K.; RACKE, H. Balanced graph partitioning. *Theory of Computing Systems*, Springer, v. 39, n. 6, p. 929–939, 2006.

ESTER, M. et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In: *Kdd*. [S.l.: s.n.], 1996. v. 96, n. 34, p. 226–231.

HAND, D. J. Principles of data mining. *Drug safety*, Springer, v. 30, n. 7, p. 621–622, 2007.

HILBERT, M.; LÓPEZ, P. The world's technological capacity to store, communicate, and compute information. *science*, American Association for the Advancement of Science, p. 1200970, 2011.

HORTA, V. et al. Analyzing scientific context of researchers and communities by using complex network and semantic technologies. *Future Generation Computer Systems*, Elsevier, v. 89, p. 584–605, 2018.

HORTA, V. A. C. *Detecção de comunidades sobrepostas em Redes Sociais Científicas*. 46 f. Monografia (Graduação) — Universidade Federal de Juiz de Fora, Juiz de Fora, 2017.

JAIN, A.; MURTY, M.; FLYNN, P. Data clustering: A review acm computing surveys, vol. 31. *Google Scholar*, p. 264–318, 1999.

JOHNSON, D. S. et al. Optimization by simulated annealing: An experimental evaluation; part i, graph partitioning. *Operations research*, INFORMS, v. 37, n. 6, p. 865–892, 1989.

KIM, W. Parallel clustering algorithms: survey. *Parallel Algorithms, Spring*, 2009.

KOWALIK, Ł. Approximation scheme for lowest outdegree orientation and graph density measures. In: SPRINGER. *International Symposium on Algorithms and Computation*. [S.l.], 2006. p. 557–566.

KRUSKAL, J. B. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical society*, JSTOR, v. 7, n. 1, p. 48–50, 1956.

KUMAR, R. A. Impact of big data analytics on healthcare and society.'. *J Biom Biostat*, v. 7, n. 300, p. 2, 2016.

LESKOVEC, J.; RAJARAMAN, A.; ULLMAN, J. D. *Mining of massive datasets*. [S.l.]: Cambridge university press, 2014.

LETOUZÉ, E. Big data for development. *Retrieved April*, v. 2, n. 2013, p. 14, 2013.

LEY, M. The dblp computer science bibliography: Evolution, research issues, perspectives. In: SPRINGER. *International symposium on string processing and information retrieval*. [S.l.], 2002. p. 1–10.

LIPTON, R. J.; TARJAN, R. E. A separator theorem for planar graphs. *SIAM Journal on Applied Mathematics*, SIAM, v. 36, n. 2, p. 177–189, 1979.

LOHR, S. The age of big data. *New York Times*, v. 11, n. 2012, 2012.

MARR, B. *Big Data: The Mega-Trend That Will Impact All Our Lives @ONLINE*. 2013. Disponível em: ⟨https://www.linkedin.com/pulse/20130827231108-64875646-big-data-the-mega-trend-that-will-impact-all-our-lives⟩.

MARR, B. Big data: The 5 vs everyone must know. *LinkedIn Pulse*, v. 6, 2014.

MCAFEE, A. et al. Big data: the management revolution. *Harvard business review*, v. 90, n. 10, p. 60–68, 2012.

MEENA, J.; DEVI, V. S. Overlapping community detection in social network using disjoint community detection. In: IEEE. *Computational Intelligence, 2015 IEEE Symposium Series on*. [S.l.], 2015. p. 764–771.

MENEZES, V. S. A. et al. Extração de informações para aprimorar a previsão de relacionamentos em rede social científica. In: *SBBD (Short Papers)*. [S.l.: s.n.], 2015. p. 75–80.

NEWMAN, M. E. Modularity and community structure in networks. *Proceedings of the national academy of sciences*, National Acad Sciences, v. 103, n. 23, p. 8577–8582, 2006.

PRIM, R. C. Shortest connection networks and some generalizations. *Bell system technical journal*, Wiley Online Library, v. 36, n. 6, p. 1389–1401, 1957.

RAINIE, L.; WELLMAN, B. *Networked: The new social operating system*. [S.l.]: Mit Press, 2012.

SEMAAN, G. S. *Algoritmos para o Problema de Agrupamento Automático*. Tese (Doutorado) — Tese de Doutorado, Instituto de Computação, Universidade Federal Fluminense, 2013.

SIPSER, M. *Introduction to the Theory of Computation*. [S.l.]: Thomson Course Technology Boston, 2006. v. 2.

SKILLICORN, D. Strategies for parallel data mining. *IEEE concurrency*, IEEE, v. 7, n. 4, p. 26–35, 1999.

STRÖDE, V. et al. Information extraction to improve link prediction in scientific social networks. In: IEEE. *Computer Supported Cooperative Work in Design (CSCWD), 2016 IEEE 20th International Conference on*. [S.l.], 2016. p. 515–520.

STRÖELE, V.; ZIMBRÃO, G.; SOUZA, J. M. Group and link analysis of multi-relational scientific social networks. *Journal of Systems and Software*, Elsevier, v. 86, n. 7, p. 1819–1830, 2013.

STROGATZ, S. H. Exploring complex networks. *nature*, Nature Publishing Group, v. 410, n. 6825, p. 268, 2001.

TALIA, D. Parallelism in knowledge discovery techniques. In: SPRINGER. *International Workshop on Applied Parallel Computing*. [S.l.], 2002. p. 127–136.

TIAN, J. et al. Improvement and parallelism of k-means clustering algorithm. *Tsinghua Science & Technology*, Elsevier, v. 10, n. 3, p. 277–281, 2005.

WU, X. et al. Data mining with big data. *IEEE transactions on knowledge and data engineering*, IEEE, v. 26, n. 1, p. 97–107, 2014.

XU, X.; JÄGER, J.; KRIEGEL, H.-P. A fast parallel clustering algorithm for large spatial databases. In: *High Performance Data Mining*. [S.l.]: Springer, 1999. p. 263–290.

YANG, L. et al. High performance data clustering: a comparative analysis of performance for gpu, rasc, mpi, and openmp implementations. *The Journal of supercomputing*, Springer, v. 70, n. 1, p. 284–300, 2014.

ZACHARY, W. W. An information flow model for conflict and fission in small groups. *Journal of anthropological research*, University of New Mexico, v. 33, n. 4, p. 452–473, 1977.

ZHANG, Q. et al. An incremental cfs algorithm for clustering large data in industrial internet of things. *IEEE Transactions on Industrial Informatics*, IEEE, v. 13, n. 3, p. 1193–1201, 2017.