Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Efeitos do *coarsening* na classificação de grafos k-partidos

Paulo Eduardo Althoff

Dissertação apresentada como requisito parcial para
conclusão do Mestrado em Informática

Orientador
Prof. Dr. Thiago de Paulo Faleiros

Brasília
2022

Ficha catalográfica elaborada automaticamente,
com os dados fornecidos pelo(a) autor(a)

AA467e

Althoff, Paulo Eduardo
Efeitos do coarsening na classificação de grafos k partidos / Paulo Eduardo Althoff; orientador Thiago de Paulo Faleiros. -- Brasília, 2022.
97 p.

Tese (Doutorado - Mestrado em Informática) -- Universidade de Brasília, 2022.

1. classificação de vértices. 2. contração de redes. 3. grafos heterogêneos. 4. grafos k-partidos. 5. coarsening. I. Faleiros, Thiago de Paulo, orient. II. Título.

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Efeitos do *coarsening* na classificação de grafos k-partidos

Paulo Eduardo Althoff

Dissertação apresentada como requisito parcial para
conclusão do Mestrado em Informática

Prof. Dr. Thiago de Paulo Faleiros (Orientador)
CIC/UnB

Prof. Dr. Díbio Leandro Borges
UnB

Prof. Dr. Alneu de Andrade Lopes
USP

Prof. Dr. Ricardo Pezzuol Jacobi
Coordenador do Programa de Pós-graduação em Informática

Brasília, 13 de junho de 2022

Dedicatória

Dedico esta dissertação a meu pai Carlos Henrique e à minha mãe Ilse (*in memoriam*) principais responsáveis por tudo que eu tenho e conquistei, por sempre incentivarem meus estudos e sempre estarem presentes.

Agradecimentos

A Deus, por tudo o que me proporciona na vida.

À minha esposa Lilian, por todo apoio e compreensão durante estes anos, por entender o quanto este mestrado era importante para mim e relevar minha ausência neste período.

À Ana Luiza, Teddy, e Martha, por revisarem meus textos e apresentações, se dispondo inclusive ao esforço de tentar entender o que são grafos k-partidos como parte deste empenho.

Ao meu orientador Prof. Thiago de Paulo Faleiros, por estar sempre disponível para me ajudar e me guiar durante este trabalho, e principalmente por me ajudar a manter o foco.

Ao Prof. Alan Valejo, por todo apoio, tendo atuado como um co-orientador, me indicando trabalhos a serem lidos e sempre estando disponível para tirar minhas dúvidas sobre os seus artigos.

Aos meus amigos Moreira e Castro, meus exemplos de pesquisadores, pelas dicas, materiais e revisões, além de todo incentivo que me foi dado.

Aos professores Díbio e Alneu, pelas considerações e discussões durante a defesa, que acrescentaram muito ao conteúdo e forma do trabalho.

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES), por meio do Acesso ao Portal de Periódicos.

“Roma ne fu pas faite toute en un jour.”

Resumo

Com a quantidade de dados gerados no mundo ultrapassando a capacidade humana de avaliação, métodos de classificação automática de conteúdo se tornam cada vez mais relevantes. Grafos são uma forma de representação genérica de representação de entidades e suas interações, e podem ser aplicados praticamente em todo tipo de problema, principalmente em sua versão heterogênea. Por esse motivo é um dos temas que mais tem crescido no número de pesquisas na área de *machine learning*. Porém, essa mencionada quantidade de dados, resulta em grafos com elevado número de vértices e arestas, também gerando problemas nas questões de armazenamento e escalabilidade dos algoritmos aplicados a eles. Neste trabalho, é proposta a utilização de técnicas de *coarsening*, para reduzir o tamanho desses grafos como forma de endereçar estes problemas. Essa abordagem é bem estabelecida nas áreas de visualização e de particionamento de grafos, e já foi provada recentemente válida para problemas de classificação em grafos homogêneos. As questões investigadas neste trabalho dizem respeito aos níveis de economia de armazenamento e tempo de classificação proporcionados pelo *coarsening*, em comparação com os impactos causados pela redução dos grafos, em métricas de qualidade de classificação (*Accuracy*, *Precision*, *Recall* e *F-score*). O procedimento proposto foi avaliado em centenas de milhares de grafos, variando o número de vértices de entrada no intervalo de 15 a 100 mil vértices, com diferentes esquemas. Foram investigados os impactos também da quantidade de arestas intra-comunitárias e da esparsidade dos grafos nas citadas métricas. Quanto à economia de recursos, as reduções nos grafos apresentaram resultados expressivos, onde reduções de 20% no número de vértices chegam em economias de armazenamento de mais de 1/3, além de tornar as classificações cerca de duas vezes mais rápidas. Em contraste a isso, quanto à perda na qualidade das classificações, destaca-se como fato positivo a baixa perda apresentada para reduções em torno de 20%, com variações médias de $1.72 \pm 0.54\%$, $0.55 \pm 0.17\%$, $1.78 \pm 0.55\%$, $2.36 \pm 0.77\%$, em relação ao grafo original, para as métricas de *Accuracy*, *Precision*, *Recall* e *F-score*, respectivamente. Com especial destaque para a perda da métrica de *Precision*, que, além de níveis médios baixos, apresentou uma relativa estabilidade com a variação dos tamanhos dos grafos, variando menos de 4% na comparação entre grafos de 2 mil e de 100 mil vértices. Esses resultados mostram que

o *coarsening* tem grande potencial para gerar versões reduzidas dos grafos, reduzindo a quantidade de memória necessária para seu armazenamento e o tempo necessário para se classificá-los, sem, no entanto, resultar em perdas expressivas de qualidade.

Palavras-chave: classificação de vértices, contração de redes, grafos heterogêneos, grafos k-partidos, coarsening

Abstract

With the amount of data generated in the world surpassing human capacity of evaluation, automatic content classification methods have become increasingly relevant. Graphs are a generic representation form to represent entities and their interactions and can be applied to practically every kind of problem, mostly in its heterogeneous version. For this reason, it is one of the topics that has grown the most in the machine learning area. However, this mentioned amount of data results in graphs with a high number of vertices and edges, leading to problems in terms of storage and scalability for algorithms applied to them. In this work, the use of coarsening techniques is proposed to reduce the size of these graphs as a way to address these problems. This approach is well established in the areas of graph visualization and partitioning, and has been recently proven valid for classification problems in homogeneous graphs. The research questions investigated in this work concern the levels of storage savings and classification time provided by coarsening, in comparison with the impacts caused by graph reduction, on classification quality metrics (Accuracy, Precision, Recall and F-score). The proposed procedure was evaluated in hundreds of thousands of graphs, varying the number of vertices on the input graphs in the ranges from 15 to 100 thousand vertices, with different schemes. The impacts of the amount of intra-community edges and the sparseness of the graphs on the aforementioned metrics were also investigated. In the issue of resource savings, the reductions in the graphs showed expressive results, where reductions of 20% in the number of vertices imply in storage savings of more than 1/3, in addition to making the classifications about twice as fast. In contrast to this, regarding the loss in the quality of the ratings, stands out as a positive fact the low loss presented for reductions around 20%, with average variations of $1.72 \pm 0.54\%$, $0.55 \pm 0.17\%$, $1.78 \pm 0.55\%$, $2.36 \pm 0.77\%$, relative to the original graph, for the metrics of Accuracy, Precision, Recall and F-score, respectively. With special emphasis on the loss of Precision metric, which, in addition to low average levels, presented relative stability with graph sizes variation, varying less than 4% in the comparison between graphs of 2 thousand and 100 thousand vertices. These results show that coarsening has great potential to generate reduced versions of graphs, reducing memory required for their storage and the time needed to classify them, without

resulting in significant quality losses.

Keywords: node classification, network coarsening, heterogeneous graphs, k-partite graphs, coarsening

Sumário

1 Introdução **1**
1.1 Contribuições 4
1.2 Metodologia e Estrutura do Documento 6

2 Classificação semissupervisionada usando grafos **7**
2.1 Grafos 8
2.1.1 Notações Matemáticas e Definições básicas 9
2.2 Classificação por propagação de rótulos 13
2.2.1 Abordagem probabilística 15
2.2.2 *Abordagem iterativa* 16
2.2.3 Algoritmos de Regularização 18
2.3 Tratamento da Heterogeneidade 19
2.4 Tratamento para Escalabilidade 24
2.4.1 *Sampling* 26
2.4.2 *Coarsening* 27
2.4.3 Redução de grafos em classificação 31
2.4.4 Uso de *coarsening* em grafos heterogêneos 32
2.5 Avaliação de Classificações 34
2.5.1 Avaliação para problemas multi-classes 35
2.5.2 Geração de grafos sintéticos 36

3 Coarsening de grafos k-partidos usando label propagation **39**
3.1 Notações 40
3.2 Método de coarsening 42
3.3 Determinação das partições para o coarsening 46
3.4 Algoritmo 48
3.5 Trabalhos Relacionados 52

4 Análise dos efeitos do coarsening em grafos k-partidos **53**
4.1 Análises em redes sintéticas 53

4.2 Dados reais 67

5 Conclusão **70**

5.1 Limitações do Trabalho e Ameaças a Validade 72

5.2 Trabalhos futuros 73

Referências **74**

Lista de Figuras

1.1 Exemplo da maior representatividade da representação heterogênea dos dados. A figura (b) representa um grafo com interações entre postagens em redes sociais, que possui o esquema uni-partido da figura (a). Já o esquema em (c), extrai das postagens informações das palavras e figuras contidas nela, os usuários que as fizeram, e, em um segundo nível, os grupos que esses usuários pertencem, formando um grafo heterogêneo k-partido em (d). 3

2.1 Modelagem do problema das 7 pontes de Königsberg. 9
2.2 Exemplo de grafo e sua matriz de adjacências (Equação 2.1). 11
2.3 Exemplos de grafos k-partidos e seus esquemas: (a)bi-partido; (b)k-partido em estrela; (c)k-partido em fila. 12
2.4 Exemplo de equivalência estrutural, os vértices v_2, v_3, v_4 possuem as mesmas vizinhanças. 12
2.5 O grafo inicia com um conjunto rotulado e um não rotulado (a). As informações sobre os rótulos começam a ser propagadas a cada iteração para os vizinhos dos nós rotulados (b). As iterações continuam até a convergência ou até um determinado numero de iterações (c). 13
2.6 Uma mesma rede de co-autoria em suas representações homogênea (a), e heterogênea (b). 19
2.7 Propagação das informações de rótulos entre diferentes tipo. 20
2.8 Propagação das informações de rótulos entre diferentes tipo. 21
2.9 Decomposição de um grafo k-partido em grafos bi-partidos (um para cada relação do grafo original). 22
2.10 Extensão do método da Figura 2.8 para grafos k-partidos, considerando cada par de partições como uma bi-partição. 22
2.11 Exemplo de um grafo bi-partido (a), e suas projeções homogêneas (b). (Fonte: Valejo et al. (2017)) 23
2.12 Exemplo de projeção realizada com o uso de meta caminhos. 24
2.13 Fases de um processo multinível (Adaptado de Valejo, 2019). 25

2.14 Exemplo da melhora na visualização de um grafo com muitos nós após uma redução. 26
2.15 Processo de *sampling* em um grafo. 27
2.16 Exemplo de do processo de *coarsening* do grafo, com a seleção dos nós ou *matching* (a), e a o agrupamento ou *contraction* (b). 28
2.17 Procedimento de *Random Matching* (RM), dado o grafo (a), um vértice é escolhido aleatoriamente (b), e dentre seus vizinho (c), um é selecionado para o *matching* (d). 29
2.18 Procedimento de *Heavy-edge Matching* (HEM), dado o grafo (a), um vértice é escolhido aleatoriamente (b), e dentre seus vizinhos e com os pesos das arestas entre eles (c), é selecionado para o *matching* aquele com a conexão mais forte (d). 29
2.19 O coarsening de funciona com uma atribuição de um rótulo distinto para cada vértice (a). Após isso é executado o procedimento de propagação de rótulos (b), (c) e (d). E executa o *matching*, marcando os vértices que terminam com um mesmo rótulo (e), para serem unidos no *contraction* (f). 30
2.20 Projeção de uma rede bi-partida em duas redes homogêneas. 32
2.21 Matriz de Confusão para o caso binário (Adaptado de Joydwip Mohajon (2020)). 34
2.22 Matriz de Confusão para o caso multiclasses (Adaptado de Joydwip Mohajon (2020)). 36

3.1 Exemplo de, dada uma partição que se deseje classificar (marcada de vermelho em (a)), efetua-se a redução das outras partiçÕes (b) 40
3.2 Exemplo de grafo antes e após o procedimento, reduzindo todas as partições, exceto a que se deseja classificar 42
3.3 Decomposição de grafo k-partido em bi-partições. 43
3.4 O procedimento para uma bi-partição se inicia com a atribuição dos rótulos conhecidos na partição propagadora (a), em seguida, os vértices da propagadora não rotulados recebem um rótulo diferente para cada um (b). A propagação segue de forma semi-síncrona, iniciando no sentido da propagadora para a receptora (c), e alternando com outro sentido (d), até a convergência (e), quando então os vértices da partição receptora, que possuírem rótulos iguais, são agrupados (f). 44

3.5 Ordem de execução do *coarsening* de forma radial em relação a partição alvo $S_a = S_1$. De b para c o *coarsening* ocorre nos vizinhos de S_1, que recebem influência apenas da partição alvo, e em um segundo momento, de d para e as partições a um nível a mais de distância de S_a recebem influência apenas de suas partições guia, que carregam em si informações estruturais de S_a. 47
3.6 Exemplo de identificação de esquema de um grafo para aplicação do algoritmo: (a) grafo G k-partido a ser tratado pelo problema, cada uma das partições tem seus nós representados por uma cor $V = \{V_1 \cup V_2 \cup V_3 \cup V_4 \cup V_5 \cup V_6\}$ onde deseja-se classificar os nós vermelhos; (b) esquema $S(G)$ do grafo, conde $S_a = S_1$ 48
3.7 Exemplo de identificação da partição alvo do grafo (a) e de seu nó correspondente no esquema (b) 48
3.8 Exemplo de identificação da partição guia da partição não-alvo do esquema em (a); Identificada a partição não-alvo S_5 (b), traça-se P_5 (c) o menor meta-caminho entre S_5 e S_1 (alvo); Por pertencer a P_5 e estar a um-hop de distância de S_5, a partição S_4 é identificada como sendo sua partição guia (d) 49
3.9 Exemplo de extração das partições guia: (a) selecionadas as partições S_2, S_3 e S_4; (b) encontram-se P_2, P_3 e P_4 que são o caso trivial de menor caminho para $S_a = S_1$; (c) identifica-se a partição guia das três como sendo a própria S_1; (d) selecionadas as partições S_5 e S_6; (e) encontra-se P_5 e P_6, menores meta-caminhos entre S_5 e S_1 e entre S_6 e S_1, respectivamente; (f) identifica-se a partição guia de S_5 como sendo S_4 e a guia de S_6 como sendo S_2. 50

4.1 Topologias distintas usadas nos experimentos representadas por exemplos de grafos (d, e, f) e seus respectivos esquemas (a, b, c). 55
4.2 Variações do tamanho dos grafos em MB (a) e do tempo necessário para classificação dos mesmos (b) com o aumento do número de vértices nos grafos. 57
4.3 Variações do tamanho dos grafos em MB (a) e do tempo necessário para classificação dos mesmos (b) com a mudança na densidade de arestas. . . . 58
4.4 Variações nas métricas de *Accuracy* (a), *Precision (macro)* (b), *Recall (macro)* (c) e *F-score (macro)* (d) conforme o grafo é reduzido, para níveis distintos de dispersão. 59
4.5 Métricas de *Accuracy* (a), *Precision (macro)* (b), *Recall (macro)* (c) e *F-score (macro)* (d) conforme o tamanho do grafo aumenta. 60

4.6 Comparativos entre experimentos 1 e 2, para mesmos tamanho de grafos, nas métricas de *Accuracy* (a, b), *Precision (macro)* (c, d), *Recall (macro)* (e, f), e *F-score (macro)* (g, h) conforme o tamanho do grafo aumenta. . . 62
4.7 Métricas de *Accuracy* (a), *Precision (macro)* (b), *Recall (macro)* (c) e *F-score (macro)* (d) com a variação do ruído nos grafos. 63
4.8 Métricas de *Accuracy* (b), *Precision (macro)* (b), *Recall (macro)* (c) e *F-score (macro)* (d) conforme o tamanho do grafo aumenta, com o número de vértices variando de 2 mil a 100 mil. 65
4.9 Métricas de *Accuracy* (b), *Precision (macro)* (b), *Recall (macro)* (c) e *F-score (macro)* (d) conforme o tamanho do grafo aumenta, no experimento com até 100 mil vértices. 66
4.10 Esquema da rede DBLP 68
4.11 Análise de economias de armazenamento e tempo e grafo sintético de dispersão 0.02. 68
4.12 Análise de economias de armazenamento e tempo e grafo sintético de dispersão 0.02. 69
4.13 Métricas de *Accuracy* (a), *Precision (macro)* (b), *Recall (macro)* (c) e *F-score (macro)* (d) conforme o grafo é reduzido. 69

Lista de Tabelas

2.1 Notações adotadas no presenta capítulo 10
2.2 Parâmetros da ferramenta HNOC (Valejo et al., 2020b) relevantes para o presente trabalho 38

3.1 Notações adotadas no presenta capítulo 41

4.1 Economia, em tamanho de armazenamento e tempo de classificação, das reduções dos grafos em comparação com suas formas originais 57
4.2 Comparação de economias com a densidade de arestas nos grafos 58
4.3 Aumento da perda com a redução dos grafos acentuada pelo maior tamanho dos grafos (comparativo entre 2000 e 15000 vértices) 59
4.4 Aumento da perda com a redução dos grafos acentuada pelo maior tamanho dos grafos 61
4.5 Perda relativa das métricas em comparação ao grafo sem reduções em seus valores máximos 64
4.6 Dados do experimento completo para o menor (2000) e maior (100000) número de vértices 64
4.7 Distribuição dos vértices, da base DBLP, usados no experimento. 67

Capítulo 1

Introdução

Há anos a geração de dados no mundo já ultrapassa a capacidade de processamento humana, e esse efeito foi consideravelmente acelerado no biênio 2020/2021 pois, devido a pandemia do COVID-19, grande parte das atividades presenciais humanas foram substituídas por interações *online* (Josh James, 2021). E, aliado a isso, está o crescimento mundial do acesso à internet, que já vinha crescendo antes mesmo da pandemia, quando a presença na internet em termos populacionais mais que dobrou em uma década, só durante a pandemia cresceu mais 20% (Szymański and Rathman, 2021).

Todo esse volume de dados cria uma oportunidade para a área de *machine learning*. Mais dados significam uma maior capacidade de se gerar melhores modelos de abstração do mundo, porém trazem consigo um problema no que diz respeito à correta classificação desses dados. Considerando que a quantidade de dados geradas é maior do que a capacidade humana de avaliação, a classificação automática tem recebido muita atenção das pesquisas mais recentes na área de *machine learning* (van Engelen and Hoos, 2020).

A forma mais comum de *machine learning* é o *aprendizado supervisionado*, mas esse tipo de técnica exige um número considerável de dados previamente rotulados para orientar as respostas de um supervisor. Ademais, obter um conjunto de treinamento de tamanho suficiente para se resultar em um modelo acurado nem sempre é uma tarefa fácil. Nesse contexto, o aprendizado semissupervisionado possibilita encontrar soluções para problemas com menor número de informações previamente conhecidas.

No aprendizado semissupervisionado, dados rotulados são combinados com dados não rotulados para realizar o aprendizado. Algoritmos desse tipo consideram relações entre os dados não rotulados e rotulados para compensar a falta de rótulos (Silva, 2008). Em particular, a representação de dados na formas de redes ou grafos traz vantagens para algumas dessas técnicas, em que aquelas relações são extraídas das características topológicas destas estruturas (Aggarwal, 2018).

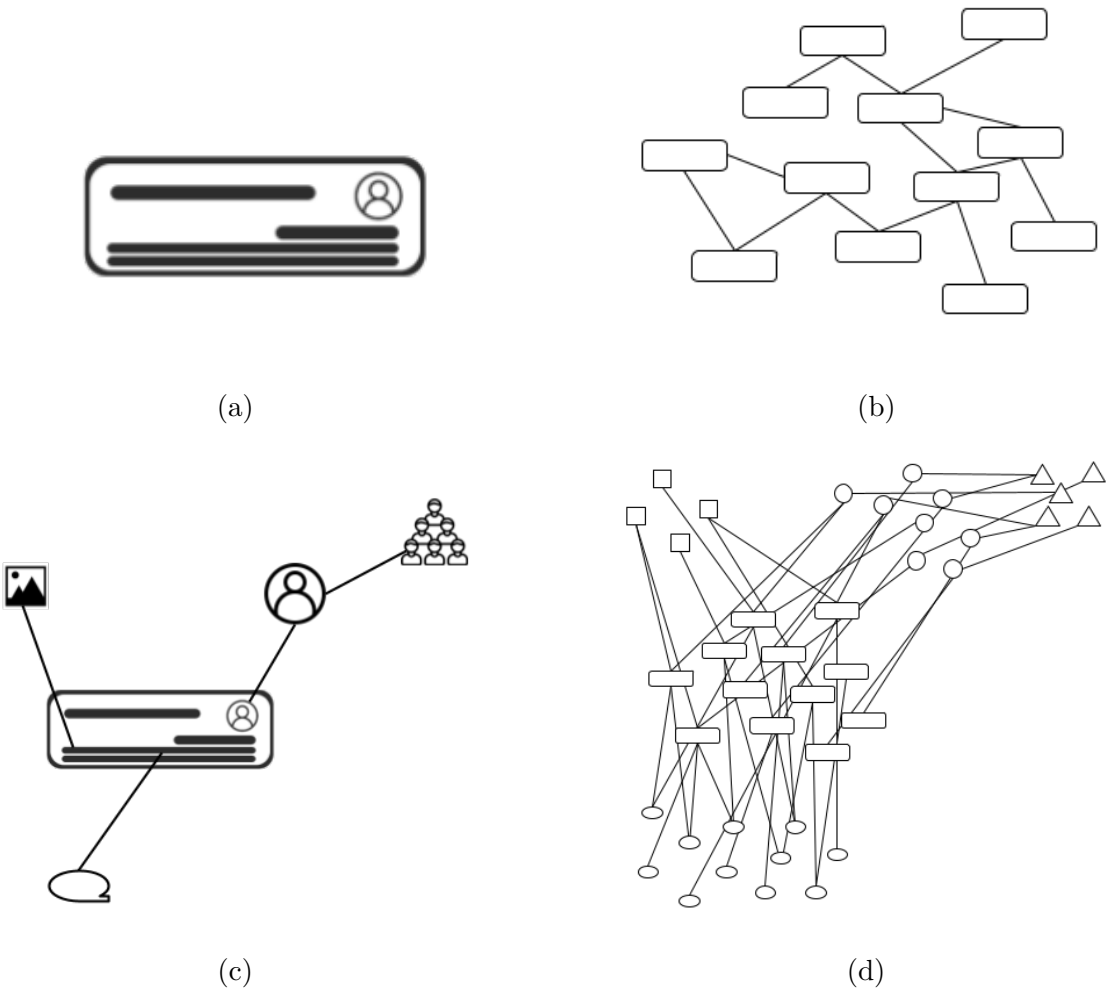
Um exemplo prático da necessidade desse tipo de técnica é a moderação de conteúdo

para se evitar propagação de notícias falsas ou discurso de ódio (Mansouri et al., 2020). Nos antigos fóruns de discussão na internet isso era feito somente por alguns usuários moderadores, porém em redes do tamanho do *Facebook* ou *Twitter*, essa abordagem não escala. Para este tipo de redes, existem algoritmos que tentam classificar os conteúdos não só analisando o texto, mas também os padrões de propagação da informação entre os usuários, e apontar alguns casos não tão claros para moderadores humanos avaliarem. Isso permite que a atuação do moderador seja escalável, por filtrar apenas os que realmente necessitam da intervenção de um especialista (Mansouri et al., 2020; Lyons, 2018; Campbell, 2021).

Apesar da abordagem semissupervisionada ter trazido uma solução para reduzir a necessidade de intervenção humana, um problema ainda presente é que, conforme as redes crescem, eleva-se a quantidade necessária de memória, tanto para armazenamento, quanto processamento em tempo de execução (Walshaw, 2004), além disso, a execução de algoritmos nesses grafos, como os de classificação listados mais acima, torna-se custosa.

Uma técnica bastante pesquisada nos últimos anos para tratar estas limitações é a de substituir os grafos por versões reduzidas dos mesmos, dessa forma reduzindo a necessidade da capacidade de armazenamento, e acelerando a execução de algoritmos nos dados em comparação com a rede em seu tamanho completo, dentre outras vantagens (Liu et al., 2018). Uma categoria dentro destas técnicas são os algoritmos de *coarsening*, que consistem em unir grupos de nós similares, reduzindo muitas vezes informações redundantes. Este tipo de técnica é bem estabelecido nas áreas de visualização e de particionamento de grafos (Valejo, 2019), e já foi provada recentemente válida para problemas de classificação em grafos homogêneos (Liang et al., 2020).

Porém, estes métodos, em sua maioria, costumam considerar apenas um tipo de vértice e arestas, conhecidas como redes uni-partidas, e as redes de informação do mundo real são mais comumente heterogêneas, contendo diferentes tipos de vértices, e arestas (Romanetto, 2020). Um modo de representação heterogênea amplamente utilizado em grafos, é o de dividir os diferentes tipos de dados em subconjuntos disjuntos, ou partições, usando arestas para representar relações entre os diferentes tipo, formando os chamados grafos k-partidos. Esse tipo de representação heterogênea tem uma maior capacidade representativa, por permitir modelar diferentes tipos de relacionamentos entre objetos distintos, como exemplificado na Figura 1.1.



(a) (b)

(c) (d)

Figura 1.1: Exemplo da maior representatividade da representação heterogênea dos dados. A figura (b) representa um grafo com interações entre postagens em redes sociais, que possui o esquema uni-partido da figura (a). Já o esquema em (c), extrai das postagens informações das palavras e figuras contidas nela, os usuários que as fizeram, e, em um segundo nível, os grupos que esses usuários pertencem, formando um grafo heterogêneo k-partido em (d).

Acompanhando o crescente interesse em explorar técnicas para grafos heterogêneos (Zhou et al., 2020; Liu et al., 2018; Wu et al., 2021), a pesquisa em métodos de *coarsening* segue um mesmo caminho, sendo amplamente explorada recentemente para os casos de grafos bi-partidos (Valejo et al., 2017, 2018a,b, 2020a, 2021), porém, métodos focando redes k-partidas ainda foram pouco explorados (Valejo, 2019).

Baseado nos dados presentes nos três últimos parágrafos, chega-se a seguinte hipótese:

Métodos de coarsening poderiam ser utilizados como forma de reduzir grafos heterogêneos, com o intuito de economia de espaço de armazenamento e escalabilidade de métodos aplicados a eles, se obtendo baixas perdas de qualidade em problemas de classificação.

Desta hipótese, surgem duas questões de pesquisa a serem respondidas:

1. Quais os níveis de economia de armazenamento e tempo de classificação proporcionados por esses procedimentos *coarsening* se aplicados em grafos k-partidos?
2. Quais os impactos da redução nos grafos em suas métricas de qualidade de classificação?

Para responder estas perguntas, os seguintes objetivos específicos são destacados:

- Identificar adaptações realizadas em algoritmos de aprendizado em grafos homogêneos para serem usados em grafos heterogêneos;
- Utilizar métodos de extensão identificados na última etapa para adaptar algoritmos de *coarsening* para grafos homogêneos e bi-partidos para seu uso em redes k-partidas; e
- Realizar análises da efetividade na redução das redes para fins de economia de espaço de armazenamento, redução no tempo de algoritmos de classificação, e perda relativa de informação da rede original, causada pela redução.

1.1 Contribuições

Apresentam-se como contribuições do presente trabalho:

- o desenvolvimento de um método de *coarsening* aplicável a grafos k-partidos (Capítulo 3), com destaques para:

  - o uso de *cross-propagation* como solução da oscilação de *label propagation* em grafos bi-partidos, da seção 3.2, que entrou como parte do trabalho "*Coarsening Algorithm via Semi-Synchronous Label Propagation for Bipartite Networks*" (Valejo et al., 2021), que recebeu o prêmio *Best Paper of Brazilian Conference on Intelligent Systems - BRACIS 2021*;
  - a seção 3.3, que propõe um método de ordenação das partições e de escolha de caminhos no esquema para conduzir o *coarsening*, se diferenciando de outros trabalhos que a fazem de forma aleatória (Zhu et al., 2016);

- uma análise dos impactos do *coarsening* em métricas de classificação de grafos k-partidos (seção 4.1), realizadas em centenas de milhares de grafos gerados sinteticamente. Que, além de apontarem o uso desse tipo de técnicas como opção promissora para resolver os problemas de armazenamento e processamento de grandes grafos, identificou a existência de um *threshold* que pode ser utilizado como auxílio na resolução de problemas em grafos reais 4.2. Essas análises podem ser usadas para identificação de níveis de redução de grafos apropriado a cada caso, dependendo das restrições de tempo/memória, e da aceitabilidade da introdução de perdas na qualidade da classificação de um problema em específico, como explicado no Capítulo 5.
- a base de código desenvolvida contendo o algoritmo proposto, os experimentos realizados e um guia de reprodução e extensão dos experimentos com o intuito de servir como base para futuros trabalho. Tanto a base quanto os guia estão disponíveis em *https://github.com/pealthoff/CoarseKlass*, sendo que a *branch* "dissertacao" irá manter a versão do código descrita neste trabalho.

1.2 Metodologia e Estrutura do Documento

Para realização do presente trabalho, a seguinte sequência de atividades foi realizada:

Primeiramente foi realizada uma revisão dos conceitos de aprendizado em grafos, o Capítulo 2 fornece os principais conceitos, iniciando com uma introdução a grafos na seção 2.1, e na área de aprendizado, focando nas métodos baseados em propagação de rótulos (2.2), que são utilizados posteriormente no trabalho.

Após isso, foram investigadas técnicas para tratar a já mencionada heterogeneidade dos grafos, a seção 2.3 apresenta alguns casos de algoritmos baseados em propagação de rótulos que foram estendidos para atender a grafos com mais tipos de vértices e arestas. Ainda no mesmo capítulo, são destacadas técnicas para tratar da escalabilidade em grandes grafos (2.4), focando nos métodos baseados em *coarsening*.

Para a questão de pesquisa definida na área da análise da aplicabilidade do *coarsening* na classificação de grafos k-partidos, um algoritmo foi proposto para realização dos testes empíricos para validação da hipótese. O Capítulo 3 explica o funcionamento desse algoritmo, e discute alguns trabalhos relacionados com o mesmo.

Com o algoritmo definido, extensivos testes foram realizados com centenas de milhares de grafos, os parâmetros destes testes são apresentados no Capítulo 4, assim como os resultados dos mesmos.

Por fim, o Capítulo 5 apresenta as conclusões do trabalho, apontando as ameaças a validade do mesmo e sugere potenciais trabalhos futuros.

Capítulo 2

Classificação semissupervisionada usando grafos

Aprendizado de máquina, ou *machine learning*, é um campo da Inteligência Artificial voltada para desenvolver algoritmos que possam aprender a realizar tarefas sem executar apenas passos previamente ensinados. Os algoritmos são genéricos o suficiente para inferir regras lógicas a partir dos dados inseridos, ou seja, adaptar as ações dadas circunstâncias extrapolando padrões.

Uma categorização muito utilizada para se definir algoritmos de *machine learning* é quanto ao tipo de *feedback* utilizado para o aprendizado, também conhecido como modo de supervisão. A forma mais comum de aprendizado de máquina é o *aprendizado supervisionado*, que se baseia na construção de um conjunto de dados denominado conjunto de treinamento, no qual amostras são classificadas por um supervisor. Dessa forma, um algoritmo, durante seu processo de aprendizado, pode verificar se suas respostas estão coerentes ou não com o esperado, assim, pode ajustar-se de forma a minimizar o erro para tal conjunto. As tarefas comumente tratadas com esta categoria de algoritmos são as de classificação e regressão (Silva, 2008), a diferença entre as duas está na saída esperada, onde a da classificação está dentro de um conjunto finito de resultados (por exemplo um previsor de resultados de um jogo que categoriza instâncias entre $[vitória, empate, derrota]$. Já a regressão busca um valor, como por exemplo um previsor de temperatura.

Já no aprendizado não-supervisionado, não há disponível nenhuma informação prévia sobre a categoria ou qualidade de uma ação. A tarefa mais comum a ser resolvida por este tipo de algoritmo é a de agrupamento, onde o algoritmo tenta detectar padrões nos dados de entrada e agrupá-los sem que os grupos estejam previamente definidos.

Para se obter respostas corretas a fim de se realizar os treinamentos no caso supervisionado, em muitos problemas reais o alto custo computacional, ou a necessidade da intervenção humana, torna o procedimento inviável. O aprendizado semissupervisionado

utiliza partes de técnicas não-supervisionadas para tentar preencher estas lacunas. Fazendo uso de relações entre os dados não rotulados e rotulados para compensar a falta de rótulos (Silva, 2008). Uma grande motivação da pesquisa deste tipo de algoritmo está no fato de que, exemplos não rotulados existem normalmente em maior abundância, enquanto os rotulados são escassos. Técnicas semissupervisionadas são utilizadas comumente em tarefas de *clustering* ou de classificação (Sanches, 2003).

Algoritmos de classificação, como o nome sugere, têm como por objetivo atribuir classes a exemplos fornecidos como entrada, se baseando em atributos destes exemplos ou a disposição topológica de suas relações. Matematicamente falando, é o processo de se descobrir uma função de mapeamento $f : X \to Y$, onde X é o conjunto dos elementos de entrada e Y é um conjunto discreto de classes.

Além da vantagem de não necessitar de uma grande quantidade de dados pré-rotulados, classificadores semisupervisionados oferecem uma melhor compreensão de as distribuições de classe de dados do que classificadores supervisionados, que usam apenas dados rotulados para o treinamento (Blum and Mitchell, 1998).

Uma das principais técnicas semissupervisionadas de aprendizado são os chamados métodos gráficos, que consistem na construção de grafos a partir do conjunto de dados do problema (Romanetto, 2020), e usar as características topológicas deste grafo para realizar a disseminação dos dados rotulados para os não rotulados.

2.1 Grafos

Grafos são representações matemáticas usadas para representar relações entre pares de objetos, onde os objetos são representados por "vértices" e suas relações por "arestas". Essas estruturas foram criadas para modelar problemas do mundo real, tendo suas origens no século XVIII, com o problema das sete pontes de Königsberg (Diestel, 2010). O enunciado do problema é: "Na cidade de Konigsberg, Alemanha, um rio passava pela cidade e a dividia em quatro partes. Para interligar estas partes, havia sete pontes. É possível, passando só uma vez por cada ponte, fazer um caminho que passe por todas elas?". A Figura 2.1 representa a modelagem desse problema utilizando grafos, que foi utilizada por Euler em 1735 para resolvê-la.

A representação por grafos permite a remoção de informações que não são relevantes para a solução de problemas que dependam apenas de vértices (neste caso as partes partes da cidade dividida pelo rio) e as arestas (neste problema as pontes que interligam aquelas partes).

O conceito de grafo é amplamente utilizado nas áreas da matemática e computação, mas sua alta flexibilidade representacional permite sua utilização em áreas como socio-

Figura 2.1: Modelagem do problema das 7 pontes de Königsberg (Fonte: Wikipedia contributors (2021)).

logia, biologia, dentre outras (Romanetto, 2020). Por ter um escopo de utilização tão extenso, muitas vezes diferentes terminologias são utilizadas. No escopo deste trabalho, podem ser considerados equivalentes os conceitos de:

- redes, grafos
- nós, vértices
- conexões, ligações, relações, arestas

2.1.1 Notações Matemáticas e Definições básicas

A Tabela 2.1 contém as notações que serão utilizadas ao longo deste capítulo de fundamentação teórica:

Para um conjunto V , considere por $V \times V$ o conjunto de todos os pares não-ordenados de elementos de V. Um **grafo** é um par $G(V, E)$ em que V é um conjunto arbitrário e E é um subconjunto de $V \times V$. Os elementos de V são chamados **vértices**, ou nós, e os de E são chamados **arestas**.

Uma aresta de E, $e_{i,j}$ representa a conexão do par (v_i, v_j), onde v_i e v_j são vértices de V, neste caso diz-se que os vértices v_i e v_j são **vizinhos** ou adjacentes. No caso de ser um grafo **dirigido**, ou seja, que a direção da aresta importa, lê-se a presença de (v_i, v_j) em E como v_i incide em v_j. Ainda, o conjunto de vértices vizinhos de v_i é representado por $\mathcal{N}(v_i)$, dita sua **vizinhança**.

A **matriz de adjacências** de um grafo G é a matriz $A(G)$ com linhas e colunas indexadas por V tal que $A_{i,j} = 1 \Leftrightarrow e_{i,j} = (v_i, v_j) \in E$, onde i e j são os índices dos vértices v_i e $v_j \in V$. Por exemplo, para o grafo da Figura 2.2, a matriz de adjacências seria dada pela Equação 2.1. Interessante notar que como não se trata de um grafo dirigido, essa matriz, por definição, se torna simétrica.

Tabela 2.1: Notações adotadas no presenta capítulo

Notação	**Descrição**
$G(V, E)$	grafo
$V = \{v_1, v_2, ..., v_{\|V\|}\}$	conjunto de vértices
E	conjunto de arestas
$\mathcal{N}(v_i)$	vizinhança, conjunto de vértices adjacentes ao vértice v_i
A	matriz de adjacências
$w(v_i, v_j)$, W	mapeamento dos pesos das arestas
$deg(v_i)$	grau do vértice v_i
$d_G(i, j)$	distância entre os vértices v_i e v_j
$V_i \subset V$	i-ésima partição do conjunto de vértices V
$V^L \subset V$	conjunto vértices rotulados
$V^U \subset V$	conjunto vértices não rotulados
P_{ij}	probabilidade de transição, em um *random walk* no grafo, entre o vértice v_i e o v_j
$C = c_1, c_2, ..., c_n$	conjunto de classes possíveis para rotulação dos vértices
$\hat{y}_i[c]$	probabilidade do vértice y_i assumir o rótulo c
Y	distribuição dos rótulos nos vértices
$D = diag\{D_{ii} = \sum_j W_{ij}\}$	matriz diagonal dos graus
TP, FN, FP, TN	True Positive, False Negative, False Positive, True Negative
d	*dispersion* ou dispersão do grafo
n	*noise* ou ruído do grafo

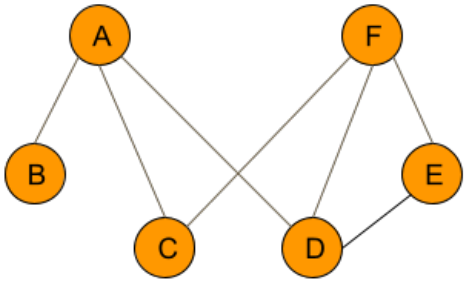


Figura 2.2: Exemplo de grafo e sua matriz de adjacências (Equação 2.1).

$$A_{6\times6} = \begin{bmatrix} 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 \end{bmatrix} \tag{2.1}$$

Um grafo é **ponderado**, se existem diferenças entre as arestas, ou seja, a cada uma delas pode ser atribuído um peso $w(v_i, v_j)$, neste caso o grafo passa a ser representado por uma tripla $G = (V, E, W)$. Se v_i e v_j não são adjacentes, definimos que $w(v_i, v_j) = 0$. Nesse caso, a matriz de adjacências passa a ter seus elementos definidos por $A_{ij} = w(v_i, v_j)$, ao invés de apenas valores unitários.

A partir disso, é definido o **grau** de um nó $deg(v_i) = \sum_{v_j \in V} w(v_i, v_j)$, particularmente, em um grafo não-ponderado o grau é igual ao número de vizinhos (pois todas as arestas tem peso unitário).

Um **caminho** de v_i para v_j é uma sequência de arestas que unidas ligam os dois nós. Um grafo é dito **conexo** se existem caminhos entre quaisquer dois nós. A **distância** $d_G(v_i, v_j)$ é dada pelo número de arestas presentes no menor caminho entre dois nós, e o **diâmetro** do grafo é dado pelo máxima distância entre um par de nós do grafo.

Um grafo G é dito **bi-partido** se existem conjuntos disjuntos V_1 e V_2 tais que $V_1 \cup V_2 = V$ e que toda aresta em E tem uma extremidade em V_1 e outra em V_2, neste caso V_1 e V_2 são ditas **partições**. Analogamente, o grafo k-partido possui k partições onde $V = \{V_1, V_2, ..., V_k\}$ e $\forall (u, v) \in E$ com $u \in V_i, v \in V_j$, $i \neq j$.

O **esquema** de um grafo k-partido pode ser definido como uma meta-representação de conexões existentes entre partições distintas. A Figura 2.3 exemplifica o conceito. Já um meta-caminho, é um caminho do esquema de um grafo, usualmente utilizado para representar uma relação entre múltiplas partições em uma determinada ordem.

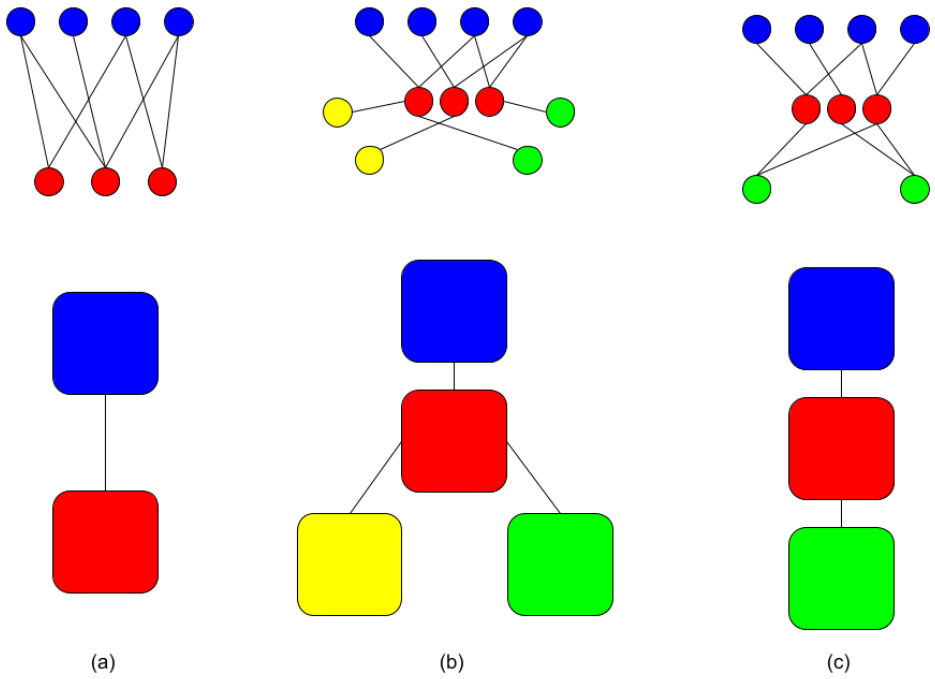


Figura 2.3: Exemplos de grafos k-partidos e seus esquemas: (a)bi-partido; (b)k-partido em estrela; (c)k-partido em fila.

Equivalência estrutural: Dois vértices são ditos estruturalmente equivalente quanto seus conjuntos de vizinhos são idênticos (Exemplo: Figura 2.4).

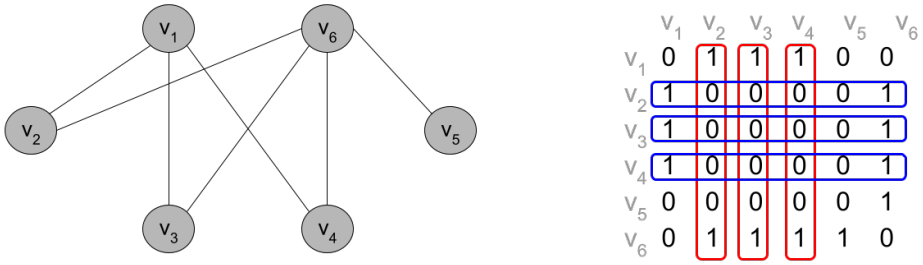


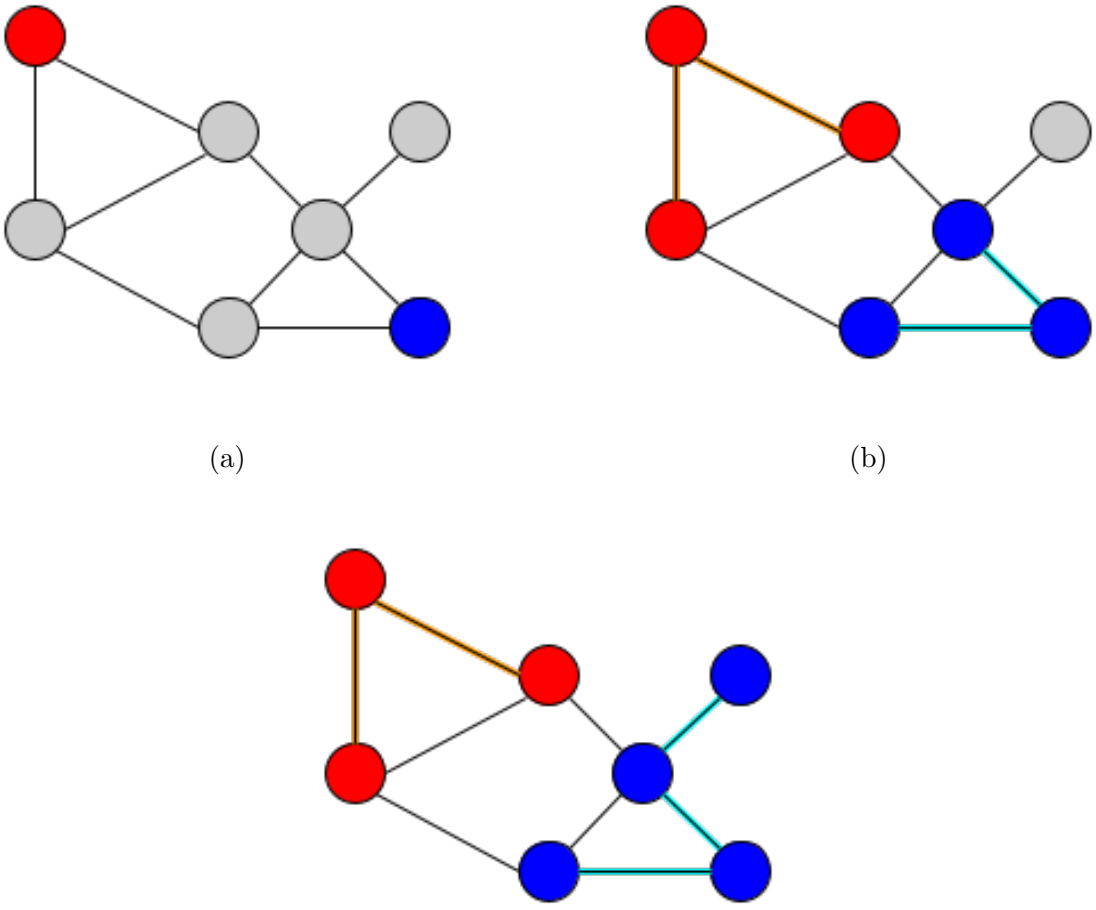
Figura 2.4: Exemplo de equivalência estrutural, os vértices v_2, v_3, v_4 possuem as mesmas vizinhanças.

Na matriz de adjacências, estes vértices possuem colunas e linhas idênticas. Com a ressalva que no caso de grafos direcionados, se dois nós apontam um para o outro mas não para si mesmos, terão esta diferença, mas ainda serão considerados equivalentes.

2.2 Classificação por propagação de rótulos

Dentro dos mencionados métodos gráficos de classificação semissupervisionada, neste trabalho serão explorados uma classe de métodos conhecida como propagação de rótulos, ou *label propagation*.

O objetivo da propagação é espalhar os rótulos de um pequeno conjunto de dados rotulados para um conjunto maior de dados não rotulados usando as relações de vizinhança do grafo 2.5. Com isso, é capaz de detectar comunidades usando apenas a estrutura de rede como guia e não requer uma função objetivo predefinida ou informações prévias sobre as comunidades (Zoidi et al., 2015).



(a) (b)

(c)

Figura 2.5: O grafo inicia com um conjunto rotulado e um não rotulado (a). As informações sobre os rótulos começam a ser propagadas a cada iteração para os vizinhos dos nós rotulados (b). As iterações continuam até a convergência ou até um determinado numero de iterações (c).

Considere um grafo $G(V, E)$ com seu conjunto de vértices $V = \{v_1, v_2, ..., v_{|V|}\}$. Suponha um conjunto de dados rotulados $V^L \subset V$, que são atribuídos rótulos do conjunto $C = c_1, c_2, ..., c_n$ e um conjunto de dados não-rotulados $V^U \subset V$, aos quais se deseja rotular. A intuição por trás deste algoritmo está na seguinte suposição: uma aresta em E conectando dois vértices distintos de V carrega uma noção de similaridade, ou seja, dois conectados têm mais elevada probabilidade de serem parecidos, do que nós não conectados (Mallawaarachchi, 2020). Assim, um rótulo pode se tornar dominante em um grupo de nós densamente conectados, mas terá problemas para cruzar uma região esparsamente conectada, definindo assim comunidades baseadas em similaridade. Dessa forma, os rótulos que seriam atribuídos a V^U, estariam condicionados a proximidade de seus vértices aos nós já rotulados de V^L. Dessa forma, é importante que a construção do grafo siga as citadas premissas.

Uma forma comum de se realizar essa construção, quando se possui representações vetoriais dos dados, é o uso da distância euclidiana destas representações, como por exemplo no algoritmo proposto por Zhu and Ghahramani (2002), onde o peso das arestas que conectam dois pontos é dado pela distância eucliadiana entre eles (Equação 2.2).

$$d_{ij} = \sqrt{(x_i - x_j)(x_i - x_j)} = \sqrt{\sum_{d=1}^{D}(x_i^d - x_j^d)^2} \tag{2.2}$$

Apesar de simples, essa métrica considera todas as dimensões da representação vetorial dos dados como iguais. Uma forma de se generalizar isso é usar a distância de Mahalanobis (Mahalanobis, 1936), que insere uma matriz A de transformações lineares e rotações nas dimensões da representação vetorial (Equação 2.3).

$$d_{ij}^M = \sqrt{(x_i - x_j)A(x_i - x_j)} \tag{2.3}$$

Diversas outras técnicas podem ser utilizadas para a montagem do grafo de acordo com características específicas do problema, como probabilidades de transição entre estados (Brin and Page, 1998), características relacionais entre os dados (Macskassy and Provost, 2003), transformações de atributos de um dado em vértices conectados (Faleiros et al., 2016), etc.

Após a definição do grafo, é realizado um método de inferência de rótulos para o conjunto V^U, a partir do conjunto V^L. Este passo pode ser derivado de diferentes abordagens, dentre os quais aqui serão destacadas: a abordagem probabilística, utilizando *random walks* (seção 2.2.1), a abordagem iterativa (seção 2.2.2), e a abordagem de regularização de grafos (seção 2.2.3) (Zoidi et al., 2015).

2.2.1 Abordagem probabilística

Uma *random walk* em G é definida como uma sequencia de vértices $v_0, ..., v_t, v_{t+1}, ...$ determinada aleatoriamente pelo processo onde dado um vértice v_t, o próximo vértice da sequência é escolhido dentre um dos vizinhos de v_t ($v_{t+1} \in \mathcal{N}(v_t)$), com uma probabilidade $P_{ij} = P(x_{t+1} = v_j | x_t = v_i)$, com $\sum_i P_{ij} = 1$. Essa distribuição de probabilidades define uma matriz de transição no grafo de forma que um agente realizando uma *random walk* no grafo, possui uma distribuição de probabilidades $p(t)$ de estar em cada um dos nós em um tempo t, a partir disso é possível se calcular a probabilidade deste agente estar nos outros nós do grafo no tempo $t + 1$ pela operação $p(t)P$. Assim, a partir de uma probabilidade inicial $p(0)$, teremos:

$$p(t) = p(0)P^t \tag{2.4}$$

No caso de grafos conectados e não bi-partidos, para *random walks* grandes o suficiente, este procedimento converge para uma distribuição limitante:

$$\lim_{t \to \infty} p(t) = \pi \tag{2.5}$$

Os métodos probabilísticos, utilizam esses *random walks* para a identificação de características topológicas dos grafos de maneira efetiva, podendo ser utilizada como técnica de amostragem. Isso pode ser feito realizando um número arbitrário de caminhadas, e considerando apenas o subgrafo composto pelos nós presentes nas *walks*. Isso pode ser usado no lugar do grafo por sua expressividade em incorporar tanto informações locais de vizinhança como de vizinhos de maior ordem (vizinho do vizinho, etc). A intuição desse método está no fato de que, dado um número suficientemente grande de caminhadas, nós que não foram acessados teriam uma importância baixa pela baixa probabilidade de se acessá-los.

Especificamente em algoritmos de *label propagation* usando *random walks* a decisão de classificação é feita baseada na probabilidade de se atingir nós rotulados partindo-se de nós não-rotulados (ou vice-versa). Por exemplo, Szummer and Jaakkola (2002) calculam as probabilidades de transição de um vértice v_i para o um vértice v_j a partir dos pesos das arestas do grafo:

$$p_{ij} = \frac{W_{ij}}{\sum\limits_k W_{ik}} \tag{2.6}$$

Então, a distribuição de probabilidade dos rótulos é calculada pela soma de inúmeras simulações de *random walk* iniciados em vértices rotulados:

$$\hat{y}_j[c_k] = \sum_{v_i \in V^L} p_{ij}^{\infty} y_i[c_k] \tag{2.7}$$

Onde $\hat{y}_i[c]$ representa a probabilidade do vértice y_i assumir o rótulo c, e $y[c]$ representa a distribuição inicial de rótulos.

A partir das probabilidades de transição, é possível simular inúmeros procedimentos de *random walks* considerando os vértices rotulados como *absorption states*, ou seja, um caminho ao atingir um nó rotulado não mais prossegue.

Seja P a matriz de probabilidades de transição entre os vértices formada pelos itens da Equação 2.6, ordenada, sem perda de generalidade, com os vértices de V^L posicionados primeiramente:

$$P = \begin{bmatrix} P_{ll} & P_{lu} \\ P_{ul} & P_{uu} \end{bmatrix} = \begin{bmatrix} \mathcal{I} & 0 \\ P_{ul} & P_{uu} \end{bmatrix} \tag{2.8}$$

Aplicando a Equação 2.5:

$$\pi = \begin{bmatrix} \mathcal{I} & 0 \\ (\sum_{n=0}^{\infty} P_{uu}^n)\, P_{ul} & P_{uu}^{\infty} \end{bmatrix} = \begin{bmatrix} \mathcal{I} & 0 \\ (\mathcal{I} - P_{uu})^{-1} P_{ul} & 0 \end{bmatrix} \tag{2.9}$$

Assim, aplicando esta matriz de transição no vetor de rótulos inicial $Y = \begin{bmatrix} Y_l \\ Y_u \end{bmatrix}$:

$$\begin{bmatrix} \hat{Y}_l \\ \hat{Y}_u \end{bmatrix} = \begin{bmatrix} \mathcal{I} & 0 \\ (\mathcal{I} - P_{uu})^{-1} P_{ul} & 0 \end{bmatrix} \begin{bmatrix} Y_l \\ Y_u \end{bmatrix} \tag{2.10}$$

$$\begin{bmatrix} \hat{Y}_l \\ \hat{Y}_u \end{bmatrix} = \begin{bmatrix} Y_l \\ (\mathcal{I} - P_{uu})^{-1} P_{ul} Y_l \end{bmatrix} \tag{2.11}$$

Logo os rótulos dados como verdadeiros no início se mantém, e, se $(I - P_{uu})$ for não-singular, a solução converge.

2.2.2 *Abordagem iterativa*

Apesar da convergência dada pelos algoritmos da seção anterior, tanto simular inúmeras *random walks*, quanto inverter matrizes na solução fechada, são procedimentos custosos. Por este motivo (Zhu and Ghahramani, 2002) propôs uma abordagem iterativa para solução do problema, que funciona partindo de dados dispostos em um espaço euclidiano, e faz a transformação desses dados para representá-los na forma de um grafo, onde o peso das arestas que conectam dois pontos é dado pela distância euclidiana (Equação 2.2) entre eles Equação 2.12,

$$w_{ij} = exp\left(-\frac{d_{ij}^2}{\sigma^2}\right) = exp\left(-\frac{\sum_{d=1}^{D}(x_i^d - x_j^d)^2}{\sigma^2}\right) \tag{2.12}$$

onde σ é um parâmetro que regula o quanto a distância entre os pontos influencia no peso das arestas.

Com essa representação é introduzida a noção de probabilidade de uma transição entre 2 nós, como sendo proporcional ao peso da aresta entre os mesmos (Equação 2.13),

$$T_{ij} = P(j \to i) = \frac{w_{ij}}{\sum_{k=1}^{l+u} w_{kj}} \tag{2.13}$$

O passo de propagação se dá da seguinte forma:

1. Propagar os rótulos entre os nós vizinhos.
2. Normalizar as linhas de Y.
3. Substituir os rótulos de vértices originalmente pertencentes ao conjunto rotulado V^L pelos rótulos originais.

Os passos 1 e 2 podem ser modelados como:

$$Y^{(t+1)} = D^{-1}WY^{(t)} \tag{2.14}$$

, onde $Y^{(t}$ seria a distribuição dos rótulos após a t-ésima iteração, e $D = diag\{D_{ii} = \sum_j W_{ij}\}$ a matriz diagonal dos graus.

A intuição por trás deste algoritmo é a que, ao invés de permitir que os rótulos originais se dissipem na propagação, a substituição efetuada no passo 3, faz com que eles atuem como fontes constantes da informação de rótulos original.

A convergência é provada de forma similar ao *random walk* com *absorption nodes*, podendo ser considerado como uma interpretação distinta da mesma solução.

Uma limitação do *label propagation* de Zhu and Ghahramani (2002), é de que os rótulos originais se mantém até o final de sua execução, não havendo a possibilidade de uma correção, pelo algoritmo, de um vértice rotulado erroneamente. O trabalho de Zhou et al. (2003) endereça este problema adicionando uma parcela na equação de atualização que com alguma probabilidade altera o rótulo de um elemento que faz parte do grupo rotulado inicialmente, ao invés de mantê-lo sempre igual ao original, o parâmetro α controla essa probabilidade:

$$Y^{(t+1)} = \alpha D^{-1/2}WD^{-1/2}Y^{(t)} + (1-\alpha)Y^{(0)} \tag{2.15}$$

2.2.3 Algoritmos de Regularização

No mesmo trabalho, Zhou et al. (2003) apresentam uma abordagem mais genérica, através de um *framework* de regularização (Equação 2.16) cuja minimização deriva não só o algoritmo em questão, mas diversas outras abordagens da literatura, inclusive o *label propagation* original.

$$Q(\hat{Y}) = \frac{1}{2}\sum_{i,j=1}^{n} W_{ij} \left\| \frac{\hat{Y}_i}{\sqrt{D_{ii}}} - \frac{\hat{Y}_j}{\sqrt{D_{jj}}} \right\| + \mu \sum_{i=1}^{n} \|\hat{Y}_i - Y_i\|^2 \tag{2.16}$$

Na Equação 2.16, o primeiro termo da soma ($\frac{1}{2}\sum_{i,j=1}^{n} W_{ij} \left\| \frac{\hat{Y}_i}{\sqrt{D_{ii}}} - \frac{\hat{Y}_j}{\sqrt{D_{jj}}} \right\|$) é chamado de *smoothness constraint*, ou consistência local, que controla a consistência de rótulos entre nós próximos. Já o segundo termo ($\mu \sum_{i=1}^{n} \|\hat{Y}_i - Y_i\|^2$), o *fitting constraint*, também chamado de consistência global, regula o quanto o resultado se aproxima dos dados conhecidos inicialmente, para os nós inicialmente rotulados. E representam uma função objetivo a ser minimizada.

Outros algoritmos desta categoria apenas diferem na escolha dessas *constraints*. Por exemplo Subramanya and Bilmes (2011) utilizam conceitos divergências de entropia entre funções de probabilidade para sua função objetivo:

$$Q(\hat{Y}) = \sum_{i=1}^{|V^L|} D_{KL}(r_i||p_i) + \mu \sum_{i=1}^{|V|} \sum_{j \in \mathcal{N}(\sqsubseteq_\rangle)} w_{ij} D_{KL}(p_i||p_j) - \sum_{i=1}^{|V|} H(p_i) \tag{2.17}$$

, onde $D_{KL}(p||q) = \sum_y \log \frac{p(y)}{q(y)}$ é a chamada divergência de Kullback-Leibler, que é uma distância estatística do quando duas funções de probabilidade são distintas (Kullback and Leibler, 1951) e $H(p) = -\sum_y p(y) \log p(y)$ é chamada de entropia de informação, que representa o grau de incerteza de uma variável aleatória (Shannon, 1948).

2.3 Tratamento da Heterogeneidade

As técnicas apresentadas na seção anterior, consideram inicialmente todos os nós e conexões dos grafos da mesma forma. Ou seja, consideram apenas um tipo de vértice e arestas, conhecidas como redes uni-partidas (Valejo, 2019). Porém, as redes de informação do mundo real são normalmente heterogêneas, contendo diferentes tipos de vértices, e arestas (Romanetto, 2020). Como é possível perceber na Figura 2.6, que apresenta uma rede de colaboração científica, a representação heterogênea tem uma maior capacidade representativa, por permitir modelar diferentes tipos de relacionamentos entre objetos distintos. Pois enquanto a versão homogênea consegue apenas apresentar as informações de co-autoria em si, a versão heterogênea consegue apresentar dados sobre quais os artigos em que os autores colaboraram, as conferências onde forma apresentados, e as palavras usadas.

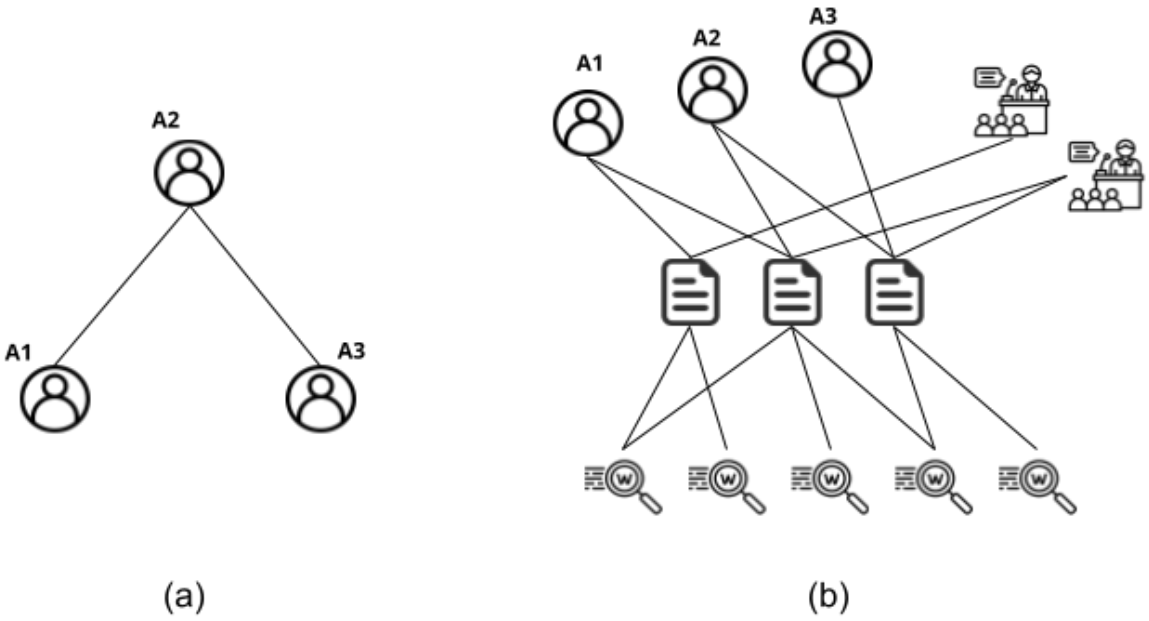


Figura 2.6: Uma mesma rede de co-autoria em suas representações homogênea (a), e heterogênea (b).

Ainda é possível utilizar os métodos para redes homogêneas nestes tipo de grafo, apenas desconsiderando as informações de tipo, porém, trabalhos que consideram essas informações de forma diferentes apresentam resultados consideravelmente superiores (Ji et al., 2010; Luo et al., 2014; Gupta et al., 2017; Romanetto, 2020).

Uma forma identificada na literatura de se fazer essa diferenciação é a de se estender algoritmos para grafos homogêneos. Seguindo o mesmo princípio de regularização de Zhou

et al. (2003), Ji et al. (2010) expandem o *framework* da Equação 2.16, com as seguintes adaptações:

Os nós do grafo são identificados como pertencendo a cada uma das diferentes partições V_i, que eles denominam "tipos", mas todos os tipos podem ser classificados dentro de um mesmo conjunto de classes.

São inseridos, no processo de propagação, dois parâmetros de regularização para lidar com a heterogeneidade das relações: $0 \leq \lambda_{ij} < 1$ representa o grau de influência que os rótulos de vértices em V_i devem ter sobre os rótulos dos vértices em V_j. Já o parâmetro $0 < \alpha_i < 1$, mede o nível de confiança que se tem nos rótulos dos vértices em V_i.

A intenção destes parâmetros é que os rótulos sejam propagados (Figura 2.7), assim como em Zhou et al. (2003), porém com a preocupação adicional de que diferentes tipos de vértices possuem diferentes tipos de influência nessa propagação. É importante ressaltar que esses parâmetros extras devem ser atribuídos manualmente por um especialista do domínio a qual o algoritmo está sendo aplicado.

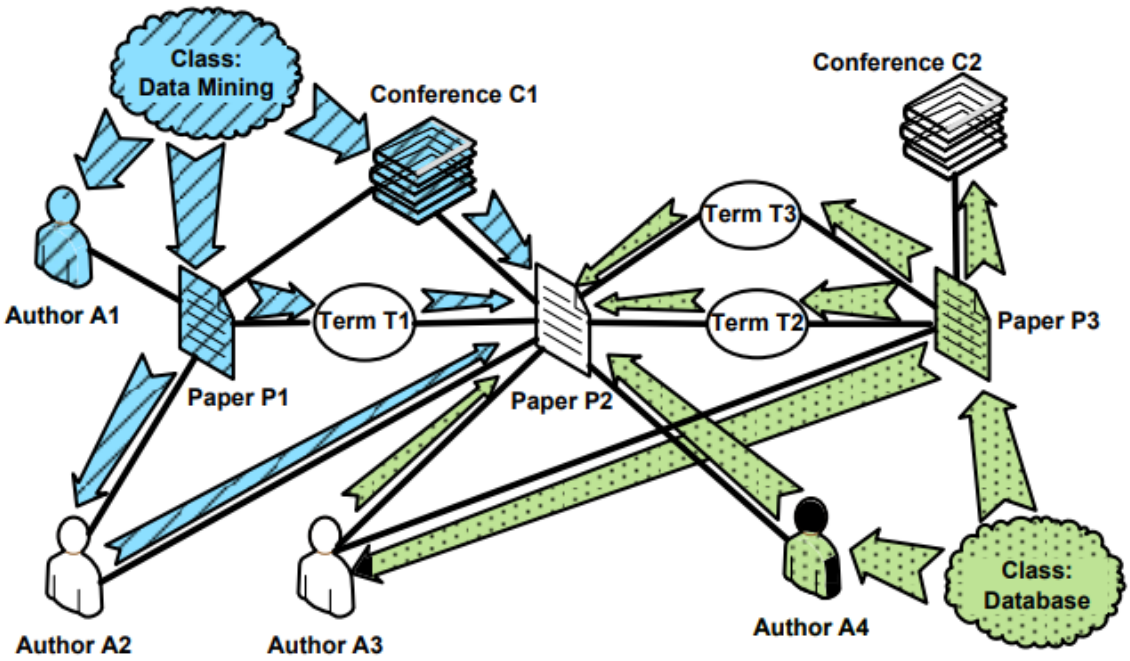


Figura 2.7: Propagação das informações de rótulos entre diferentes tipo (Fonte: Ji et al. (2010)).

Também na linha de expandir a propagação de mensagens para tipos diversos Faleiros et al. (2016) trata o problema de classificação de textos considerando o grafo bi-partido composto por "documentos" e os "termos" (palavras) presentes em seu conteúdo.

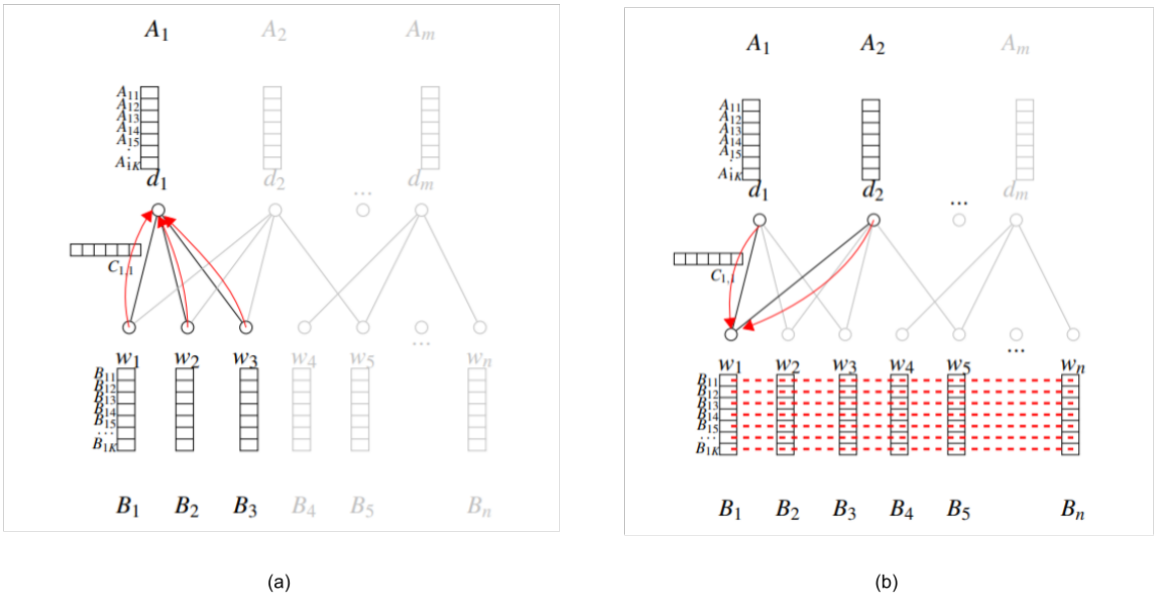


Figura 2.8: Propagação das informações de rótulos entre diferentes tipo (Fonte: Faleiros (2016)).

Neste trabalho, a função objetivo é adaptada para utilizar a divergência KL, assim como em Subramanya and Bilmes (2011). Porém para adaptar para o caso bi-partido, os rótulos utilizados são vetores de probabilidade de dimensão $|C|$, contendo a probabilidade de classificação de um vértice entre as possíveis classes do problema. Esses vetores são propagados entre as duas partições.

Um outro fato relevante a se citar do trabalho de Faleiros et al. (2016), é que ele não tenta realizar a classificação de todas as partições com um mesmo conjunto de classes (como o GNetMine faz), mas ele diferencia as partições como a de documentos sendo o alvo da classificação, e a partição de termos funcionando apenas como um atributo dos documentos que é o caminho da propagação. Esse tipo de divisão é aplicável no caso k-partido, como fazem Luo et al. (2014) e Gupta et al. (2017).

Ainda na mesma linha de algoritmos, Romanetto (2020), realiza uma extensão do trabalho de Faleiros et al. (2016) para o caso de grafos k-partidos. O método utilizado é o de considerar as relações entre partições par-a-par, como representado na Figura 2.9. Este tipo de decomposição de grafos k-partidos por pares também é encontrado em Zhu et al. (2016), Deng et al. (2017) e Valejo et al. (2020b).

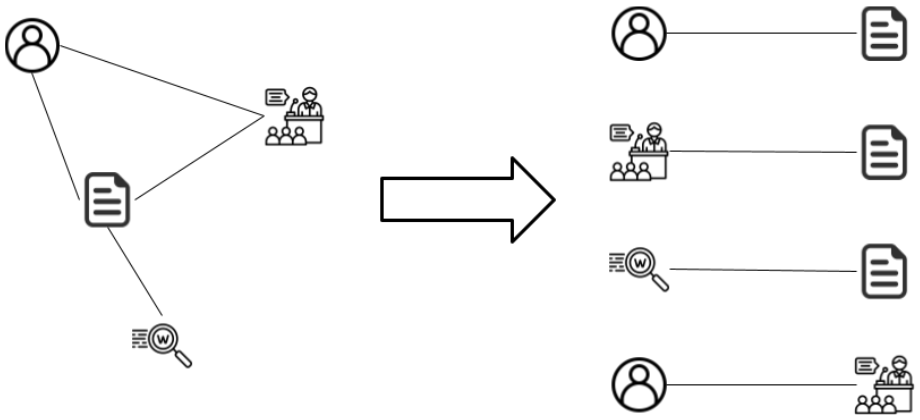


Figura 2.9: Decomposição de um grafo k-partido em grafos bi-partidos (um para cada relação do grafo original).

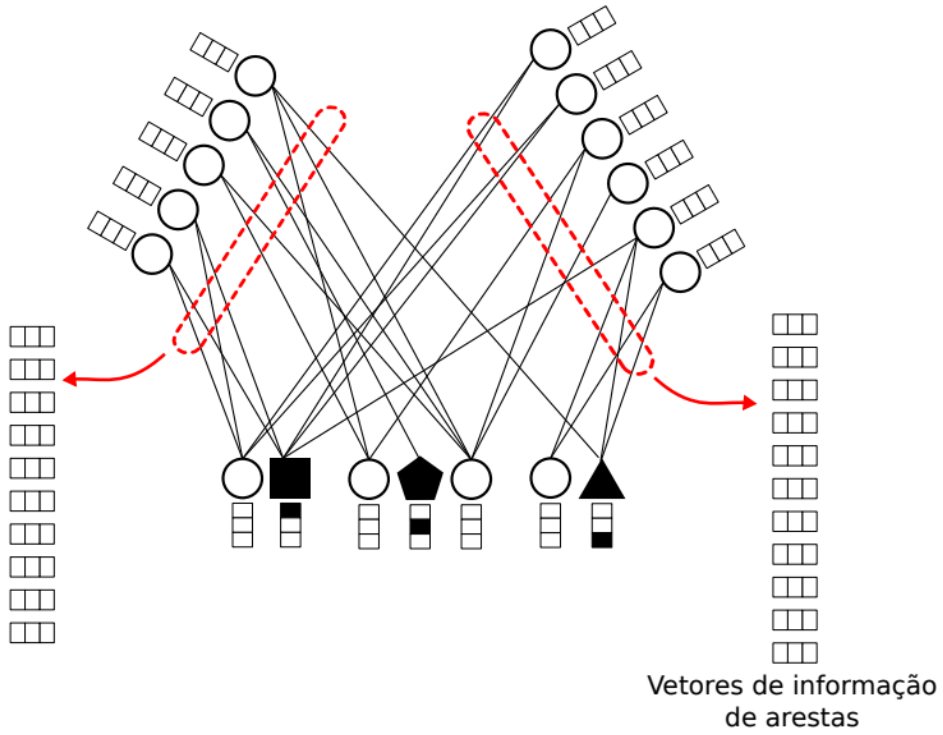
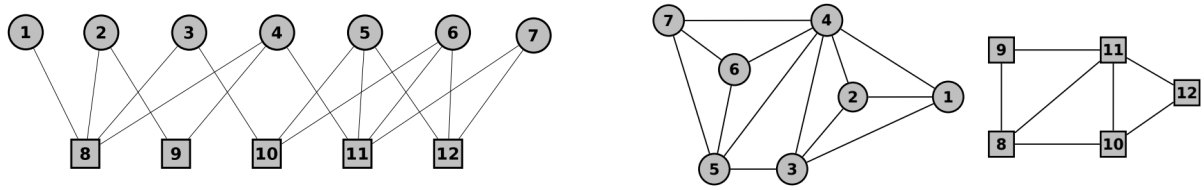


Figura 2.10: Extensão do método da Figura 2.8 para grafos k-partidos, considerando cada par de partições como uma bi-partição (Fonte: Romanetto (2020)).

Uma outra forma de se tratar a heterogeneidade de grafos é a de estender o conceito de vizinhança, criando relações homogêneas em cima da estrutura heterogênea do grafo, em um procedimento conhecido como "projeção". Exemplos deste procedimento podem

ser encontrados em (Opsahl, 2010), Padrón et al. (2011) e Valejo et al. (2017) para o caso de redes bi-partidas, onde representações homogêneas são obtidas para uma partição, considerando como vizinhos nessas novas rede homogênea, vértices que possuem vizinhos em comum na outra partição.



(a) (b)

Figura 2.11: Exemplo de um grafo bi-partido (a), e suas projeções homogêneas (b). (Fonte: Valejo et al. (2017))

De forma similar, Luo et al. (2014) e Gupta et al. (2017) também reduzem as relações a serem usadas no aprendizado para uma partição, usando meta-caminhos como forma de mapear, na estrutura heterogênea, relações relevantes para o problema tratado. Luo et al. (2014) fazem uma combinação linear de um grupo de meta caminhos pré determinado de forma a obter uma matriz de relação entre elementos de uma mesma partição alvo. Após isso expandem o *framework* de regularização da Equação 2.16 utilizando essa relação no lugar da matriz de pesos. A ideia por traz desta técnica é de que, se dois vértices são conectados por um meta-caminho considerado relevante para o problema, eles estariam altamente relacionados. Baseado nesta mesma ideia, porém fazendo uma transformação do grafo mais explícita, Gupta et al. (2017) usam os meta caminhos para gerar versões homogêneas do grafo com elementos da partição alvo que se conectam através destes meta-caminhos (Figura 2.12). As classificações dos diversos grafos homogêneos, um para cada meta-caminho definido, são combinadas linearmente com um vetor de pesos, que é treinado segundo as classificaões conhecidos.

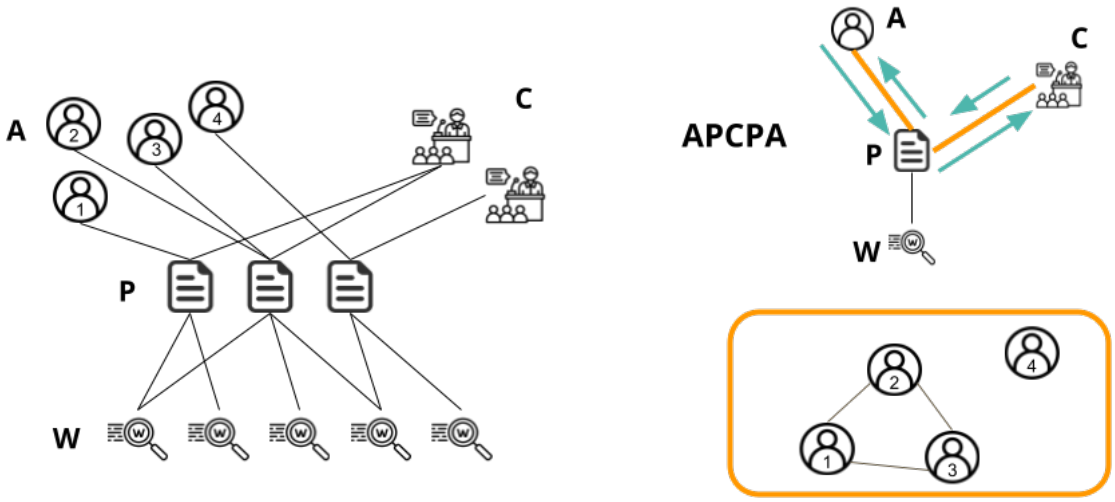


Figura 2.12: Exemplo de projeção realizada com o uso de meta caminhos.

2.4 Tratamento para Escalabilidade

Para grafos com um elevado número de nós, nem sempre é computacionalmente viável aplicar os métodos de classificação em grafos listados anteriormente, para isso, técnicas para reduzir custos foram criadas. Dentre estas técnicas estão as que realizam a redução do tamanho dos grafos (Valejo, 2014).

Os benefícios de técnicas de redução de grafos trazem diversas vantagens (Liu et al., 2018):

- **Redução do volume e armazenamento de dados**: Grafos do mundo real podem se tornar muito grandes, chegando a casa de bilhões de nós para redes sociais, por exemplo. Representações menores destes grafos levam a necessidade de menos espaço de armazenamento do que seus correspondentes originais. Além disso podem levar também a diminuição do número de operações de E/S, do número de troca de mensagens entre *clusters*, além de poder viabilizar o seu carregamento em memória;
- **Aceleração de algoritmos aplicados ao grafo**: Muitos métodos de análise em grafos não conseguem lidar com eficiência com o aumento elevado do número de vértices. Como a maioria dos algoritmos em grafos tem sua complexidade dependente do número de arestas/vértices, a redução do grafo, se conseguir manter certas características importantes para o problema tratado, resulta em uma aceleração da aplicação destes algoritmos. Esse fato viabiliza uma classe de métodos conhecidos como Multiníveis, que consistem na redução do grafo, aplicação do algoritmo alvo, e posterior projeção da solução encontrada para o grafo original (Figura 2.13).

- **Facilitação de análises visuais**: A redução do número de vértices pode facilitar uma avaliação visual realizada por um usuário, adequando a quantidade de informações à capacidade de percepção visual humana (Figura 2.14); e
- **Eliminação de ruído**: Algumas vezes a redução do tamanho dos grafos pode inclusive melhorar a qualidade dos dados pela remoção de dados errôneos. Nesses casos os padrões de disposição dos dados funciona como um filtro para eliminar esses ruídos.

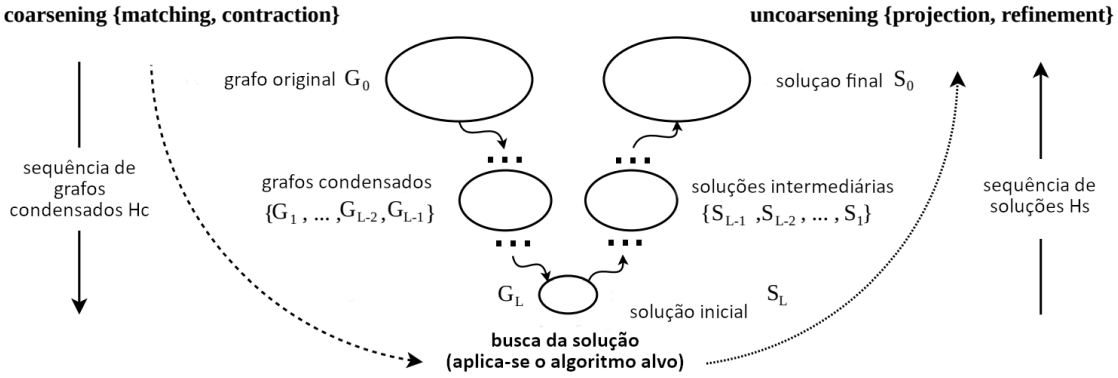


Figura 2.13: Fases de um processo multinível (Adaptado de Valejo, 2019).

É possível destacar duas principais categorias de algoritmos para realizar esta redução em grafos, as técnicas de *sampling* e as técnicas de *coarsening* (Lin et al., 2013).

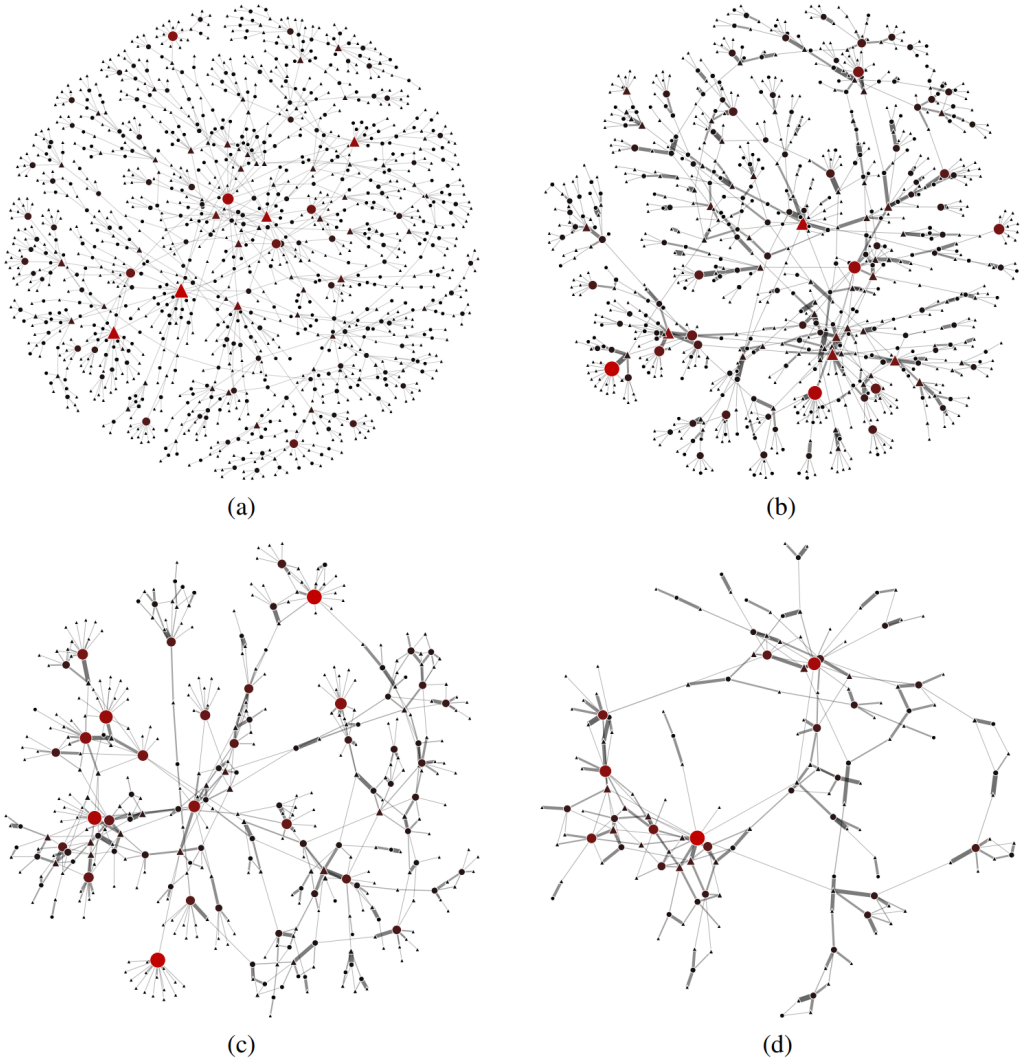


Figura 2.14: Exemplo da melhora na visualização de um grafo com muitos nós após uma redução (Fonte: Valejo (2019)).

2.4.1 *Sampling*

As técnicas de redução por *sampling* consistem em, a partir de um grafo completo, identificar gradualmente um pequeno conjunto de vértices ou arestas que representem suficientemente bem o grafo (Figura 2.15).

Existem três estratégias para realização do *sampling*, por seleção de vértices, seleção de arestas, ou por exploração (Leskovec and Faloutsos, 2006).

A opção mais simples de selecionar os vértices é pela simples escolha aleatória, selecionando uniformemente um conjunto de nós para fazerem parte do grafo reduzido. Após isso replicam-se as arestas as quais ambas as extremidades estão presentes na redução. Esta técnica é chamada de *Random Node* (RN) *sampling*.

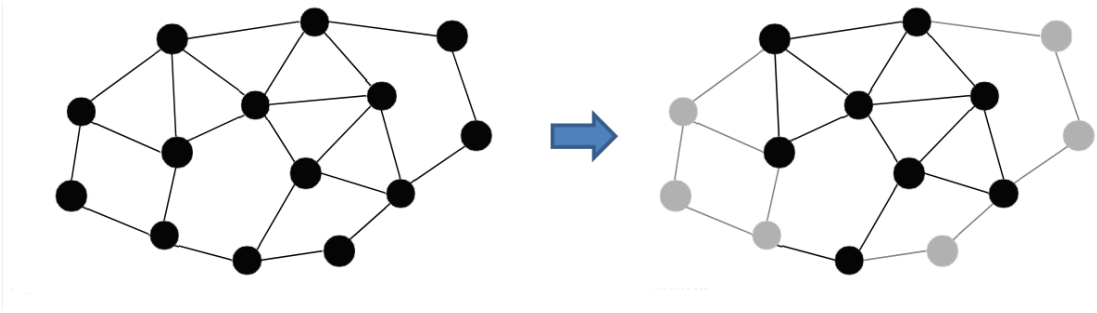


Figura 2.15: Processo de *sampling* em um grafo (Fonte: Lin et al. (2013)).

Além da distribuição uniforme, podem ser utilizadas probabilidades baseadas em características do grafo, como por exemplo a *Random Degree Node* (RDN), que prioriza nós *hubs*, ou seja, com elevado número de arestas. Ou métodos mais avançados, como o *Random PageRank Node* (RPN), priorizando de acordo com uma classificação dada pelo algoritmo *PageRank* (Brin and Page, 1998).

De forma similar, uma seleção simples de arestas pode ser dada também por uma distribuição uniforme, técnica conhecida como *Random Edge* (RE). Um problema desta técnica é que ela possivelmente eleva o diâmetro do grafo, alterando características importantes da interação entre os nós (Leskovec and Faloutsos, 2006). Já o método *Random Node-Edge* (RNE), soluciona este problema selecionando primeiro um vértice aleatoriamente, e então seleciona uma de suas arestas.

Já as estratégias de exploração consistem em percorrer o grafo utilizando *Random Walks*, e selecionar o caminho percorrido como o grafo reduzido.

Uma classe de algoritmos que também pode ser enquadrada como *sampling* são os chamados métodos de esparsificação. Que de maneira muito similar reduzem o grafo pela seleção de nós ou arestas, com a diferença apenas de que são selecionados aqueles que serão removidos do grafo, sob alguma abordagem de identificar itens de menor relevância para a estrutura geral do grafo.

2.4.2 *Coarsening*

Algumas abordagens empregam técnicas de, iterativamente, agrupar elementos dos grafos que atendam a alguma medida de similaridade. Estas técnicas são conhecidas como *coarsening*.

A partir de um grafo G_0, são gerados sucessivamente os grafos $G_1, G_2, ...G_L$ agrupando seus nós e arestas. Os vértices agrupados também são chamados de super-vértices, e, de forma análoga, as arestas agrupadas de super-arestas. A Figura 2.16 apresenta um exemplo do processo. Este procedimento pode ser subdividida em dois passos: no passo

de *matching*, os vértices que serão agrupados são selecionados, já no passo de *contraction* o novo grafo é formado, segundo a regras de contração dos pesos e valores dos elementos envolvidos.

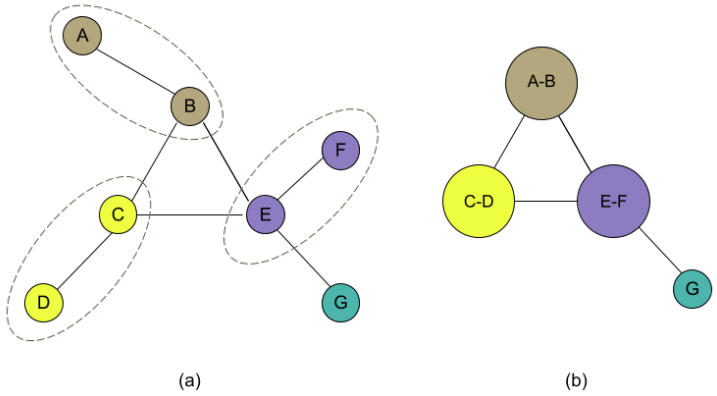


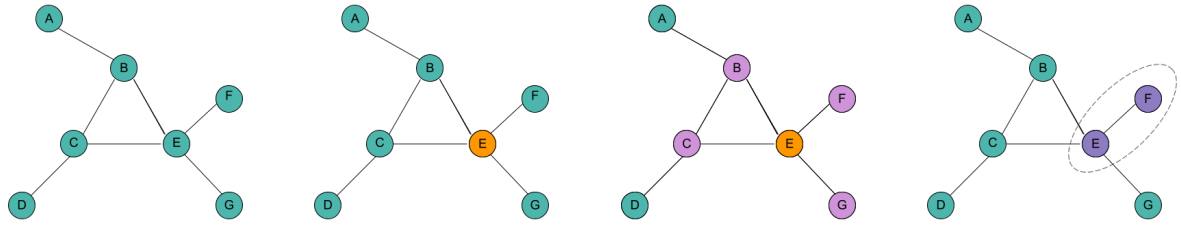
Figura 2.16: Exemplo de do processo de *coarsening* do grafo, com a seleção dos nós ou *matching* (a), e a o agrupamento ou *contraction* (b).

Matching

Uma escolha de uma estratégia de *matching* adequada é um componente chave para uma eficiente redução do grafo uma vez que ele é responsável por selecionar os vértices a serem colapsados. Consequentemente, ela interfere de forma determinante na qualidade e no viés da representação reduzida do grafo (Valejo, 2019).

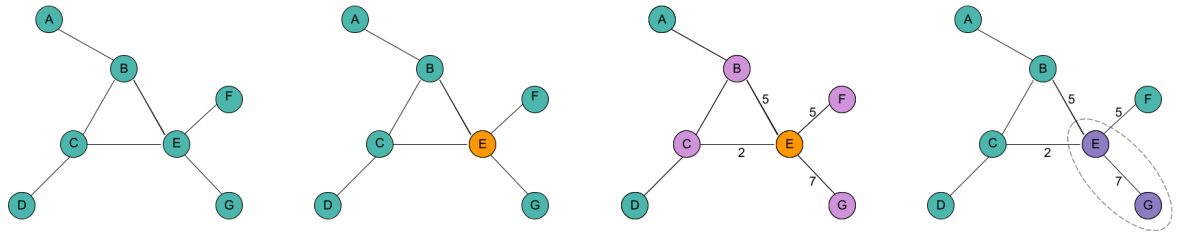
Trabalhos na área de particionamento de grafos utilizaram técnicas de seleção baseadas na escolha aleatória de nós (Bui and Jones, 1993; Hendrickson, 1994). Karypis and Kumar (1995) elencam e nomeiam estas técnicas, a mais básica, chamada de *Random Matching* (RM), onde um vértice do grafo é selecionado de maneira aleatória dentre os nós que ainda não passaram pelo *matching*. Dentro os vizinhos deste nó, escolhe-se um vértice que também ainda não tenha sido selecionado. Se não existir tal vizinho, marca o próprio vértice para fazer parte do grafo reduzido, se existir, marca ambos os nós para que uma união dos dois, chamada de super-nó, faça parte da redução. O procedimento é repetido até que todos os vértices tenham sido marcados (Figura 2.17).

Uma versão modificada desta seleção, chamada de *Heavy-edge Matching* (HEM), altera a escolha do vizinhos como sendo aquele que possui aresta de maior peso com o vértice sendo analisado (Figura 2.18). O objetivo desta técnica é minimizar o peso médio das arestas do grafo condensado.



(a) (b) (c) (d)

Figura 2.17: Procedimento de *Random Matching* (RM), dado o grafo (a), um vértice é escolhido aleatoriamente (b), e dentre seus vizinho (c), um é selecionado para o *matching* (d).



(a) (b) (c) (d)

Figura 2.18: Procedimento de *Heavy-edge Matching* (HEM), dado o grafo (a), um vértice é escolhido aleatoriamente (b), e dentre seus vizinhos e com os pesos das arestas entre eles (c), é selecionado para o *matching* aquele com a conexão mais forte (d).

Diversas outras estratégias similares surgiram, alterando a característica do grafo em que se deseja-se priorizar, como por exemplo, o *Modified Heavy-edge Matching* (HEM*) utiliza os graus do vértice para o desempate na seleção do HEM.

Zhu et al. (2012) introduzem variações baseadas em métricas de similaridade estrutural dos vértices, ou seja, com vizinhanças similares. Liang et al. (2020), em uma forma mais estrita, agrupam nós que possuem a mesma exata vizinhança, com o foco em reduzir redundância nas informações da rede.

Diferente dos métodos apresentados acima, onde os nós são combinados por pares, Meyerhenke et al. (2014) propõe um método onde é possível combinar conjuntos de vértices de uma vez, levando a uma convergência mais rápida. Para isso, atribuindo rótulos aos nós e fazendo um procedimento de *label propagation* proposto por Raghavan et al. (2007), marca-se para serem condensados todos os nós que convergirem para um mesmo rótulo (2.19).

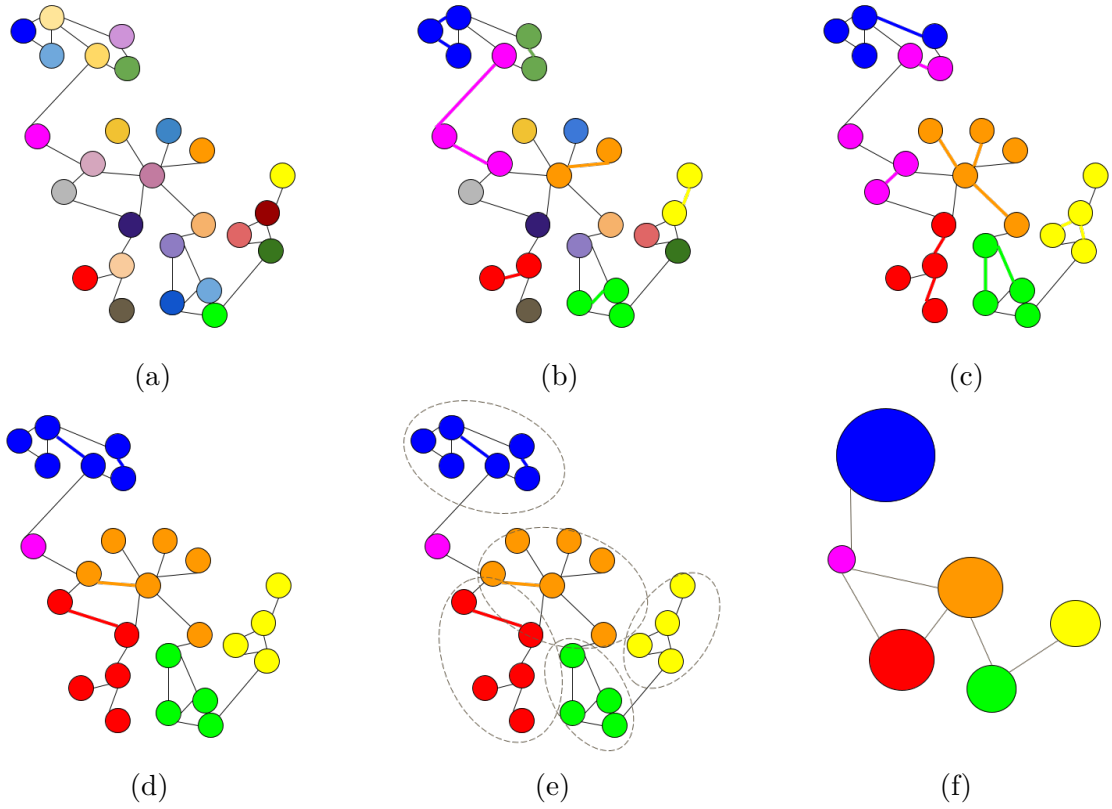


Figura 2.19: O coarsening de funciona com uma atribuição de um rótulo distinto para cada vértice (a). Após isso é executado o procedimento de propagação de rótulos (b), (c) e (d). E executa o *matching*, marcando os vértices que terminam com um mesmo rótulo (e), para serem unidos no *contraction* (f).

Contraction

A etapa de *contraction* é onde, a partir de um nível de *coarsening* do grafo G_i, o seu chamado grafo sucessor G_{i+1} é gerado. Esse grafo sucessor é composto de todos os super-vértices resultantes do processo de *matching* no grafo G_i, e ainda, herda vértices que não tenham sido marcados para algum agrupamento. Caso os nós possuam algum valor ou peso, estes poderão ser agrupados no super-vértice resultante, o mesmo vale para as arestas, uma aresta entre 2 vértices u_i e v_i em um grafo G_i, será formada pela combinação de todas as arestas em G_{i-1} que possuam como uma de suas extremidades um vértice que foi agrupado em v_i e outra em um vértice que foi agrupado em u_i.

Um super-vértice do grafo sucessor G_{i+1}, é chamado de vértice sucessor dos nós de G_i que foram unidos na fase de *contraction*. Da mesma forma, arestas unidas terão uma aresta sucessora.

A forma de agregar pesos de arestas ou dados associados aos vértices vai depender de cada implementação, podendo ser adaptada conforme o problema, ou otimizada segundo alguma função objetivo. A forma mais simples é a de simplesmente ignorar estes dados, caso não sejam relevantes para o problema. Já a forma mais comum, que mostra ser uma boa aproximação, se dá pela soma dos pesos associados às estruturas predecessoras (Valejo, 2019).

Alguns trabalhos introduzem algum tipo de normalização como por exemplo, Liang et al. (2020) reduzem os valores de pesos das super-arestas de acordo com os graus dos vértices envolvidos na contração. O objetivo desse procedimento é o de diminuir a influência de nós *hubs* (com muitas arestas). Outros adotam procedimentos ainda mais complexos, como Cai et al. (2021), que introduzem uma rede neural para aprender como distribuir melhor esses pesos.

2.4.3 Redução de grafos em classificação

Trabalhos mais recentes mostram a utilidade de representações reduzidas de grafos na tarefa de classificação de nós.

Chen et al. (2017) iteram o procedimento de *coarsening* em um método multi-nível, onde os super-vértices de cada uma das camadas, são usados para criar representações vetoriais dos dados originais. Esta técnica obtém sucesso em obter *embeddings* de boa qualidade, porém a um custo computacional elevado. Liang et al. (2020) simplificam este procedimento realizando esta transformação apenas utilizando os super-vértices do grafo mais reduzido. E, para compensar a perda de informação dos outros níveis, após a classificação dos nós um treinamento com uma rede neural para mapear as representações encontradas nos vértices do grafo original.

Nos dois métodos citados no último parágrafo, existe o problema do elevado número de nós, o que inviabiliza o processamento em *hardwares* mais modestos, devido a limitações de memória. Akyildiz et al. (2020) resolvem o problema do tamanho do grafo particionando-o de forma a possibilitar o processamento de partes menores do grafo separadamente.

2.4.4 Uso de *coarsening* em grafos heterogêneos

O uso de *coarsening* no contexto de grafos heterogêneos apresenta um desafio no procedimento de *matching*. Pois realizar a seleção de vértices pertencentes a tipos distintos degeneraria a estrutura do grafo.

Um dos métodos para se sobrepassar este problema é a aplicação de projeções homogêneas no grafo, como já mencionado na seção 2.3, que consiste em considerar cada tipo de vértice em um grafo separado, onde as conexões são construídas a partir das conexões heterogêneas do grafo original.

No trabalho de Valejo et al. (2017), esse conceito de projeção é utilizado em grafos bi-partidos, considerando como vizinhos na projeção, os nós a 2-*hops* de distância, ou seja, os vizinhos dos vizinhos (2.20).

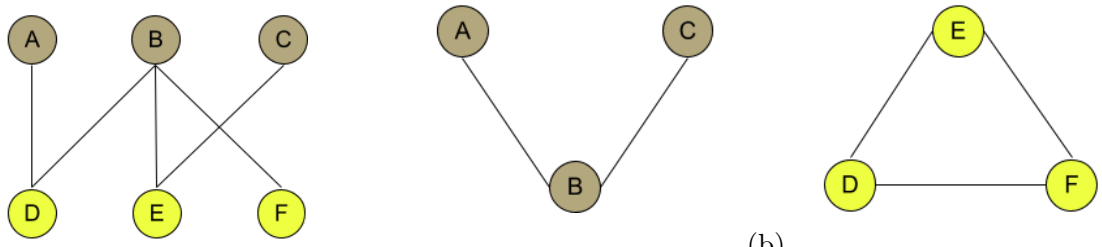


Figura 2.20: Projeção de uma rede bi-partida em duas redes homogêneas.

Após feita a projeção, métodos tradicionais de *coarsening* podem ser aplicados separadamente em cada um dos grafos homogêneos.

Valejo et al. (2020a) apresenta um método que remove a necessidade da projeção. Para isso, estende o trabalho já mencionado de Meyerhenke et al. (2014), para o caso de grafos bi-partidos, utilizando propagação de rótulos como forma de guiar o processo de *coarsening* em ambas as partições.

A estrutura bi-partida pode levar a problemas de oscilação em métodos de propagação de rótulos, por esse motivo (Valejo et al., 2021) realiza procedimento similar, porém utilizando a estratégia de propagação semi-síncrona de *cross-propagation* (Liu and Murata, 2009).

Já para problemas com mais de duas partições, normalmente o método utilizado é o mostrado na seção 2.3, de decompor a estrutura por pares de relações diferentes,

representando-as como bi-partições (Figura 2.9). Neste contexto Zhu et al. (2014) utilizam um modelo de regularização para particionar um grafo tri-partido de redes sociais com o intuito de realizar análise de sentimentos no *Twitter*. Para isso, o grafo é dividido em 3 grafos bi-partidos, realizando uma propagação através das funções de atualização apresentadas em Ding et al. (2006). Os bons resultados levaram Deng et al. (2017) a estendê-lo para o *coarsening* em grafos k-partidos, que fazem particionamento aleatório e posterior regularização através de uma função alvo com todas as conexões entre partições consideradas, porém, o próprio trabalho aponta os problemas de escalabilidade que isso causa, devido a sua complexidade quadrática.

2.5 Avaliação de Classificações

Na avaliação da qualidade de uma classificação, para cada uma das classes existentes, verificam-se métricas relativas à correta classificação das entradas. Em problemas de classificação binária, onde as possíveis classes são *True* ou *False*, os casos onde o classificador identifica um elemento corretamente como *True* são chamados de *True Positive* (TP), já os casos onde um elemento é da classe *True*, mas o classificador identifica como *False*, são chamados de *False Negative* (FN). Por outro lado, quando o elemento deveria ser classificado como *False*, mas o classificador identifica como *True*, são os *False Positive* (FP), e se corretamente identificados, *True Negative* (TN). A matriz de confusão (Figura 2.21) é uma ferramenta para demonstrar visualmente esses valores.

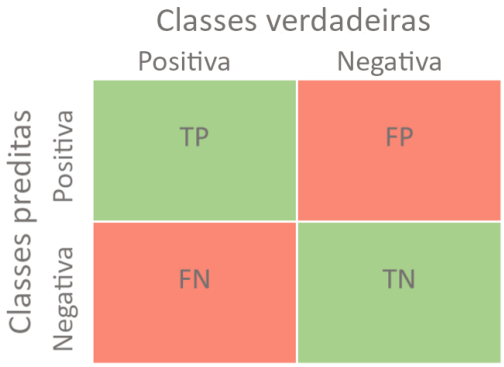


Figura 2.21: Matriz de Confusão para o caso binário (Adaptado de Joydwip Mohajon (2020)).

A partir destes conceitos, as seguintes métricas podem ser calculadas:

Accuracy: Fração de exemplos corretamente identificados pelo classificador

$$\frac{TP + TN}{TP + TN + FP + FN} \tag{2.18}$$

Precision: Diz quantos dos casos classificados como Positivo eram realmente positivos

$$\frac{TP}{TP + FP} \tag{2.19}$$

Recall: Proporção dos casos positivos corretamente identificados, também chamado de *Sensitivity* ou *True Positive Rate* (TPR)

$$\frac{TP}{TP + FN} \tag{2.20}$$

A escolha dentre cada uma destas métricas irá depender do problema em específico, existem problemas onde é mais importante se identificar um caso positivo do que evitar um negativo, e em outros o contrário é verdade. Por exemplo, em diagnósticos médicos, normalmente se privilegia o *Recall*, pois um *False Positive* irá levar a realização de novos exames que poderão identificar o erro, já um *False Negative* pode erroneamente fazer com que o paciente não se preocupe mais com a sua doença. Por outro lado, em um julgamento de um crime, o *Precision* é mais indicado (considerando a condenação como o efeito positivo), pois é mais interessante tentar ao máximo não condenar um inocente (FP), mesmo que seja ao custo de em alguns casos não se condenar o culpado (FN).

$\mathbf{F}_\beta$-score: Combina os valores de *Precision* e *Recall* através de uma média harmônica ponderada,

$$F_\beta score = (1 + \beta^2)\frac{Recall \times Precision}{Recall + \beta^2 Precision} \tag{2.21}$$

onde o parâmetro β identifica quantas vezes o valor do *Recall* é mais importante do que o de *Precision*. Quando $\beta = 1$ essa métrica é reduzida a média harmônica entre *Precision* e *Recall* e é chamada simplesmente de F-score.

2.5.1 Avaliação para problemas multi-classes

Para o caso multi-classes, dado um conjunto de classificações possíveis:

$$C = \{c_1, c_2, ..., c_n\} \tag{2.22}$$

os mesmos termos e equações podem ser utilizados do ponto de vista de uma única classe. Tomando a classe c_i como base, os casos onde o elemento é categorizado pelo classificador como sendo c_i são os casos *Positive*, e se classificados como $c_j, j \neq i$, são os casos *Negative* (Figura 2.22).

Com essa definição, os insumos das métricas apresentadas acima para cada uma das classes são dados por:

$$TP_{c_a} = m_{aa} \tag{2.23}$$

$$TN_{c_a} = \sum_{1}^{n} m_{ij}, i \neq a \wedge j \neq a \tag{2.24}$$

$$FP_{c_a} = \sum_{1}^{n} m_{aj}, j \neq a \tag{2.25}$$

$$FN_{c_a} = \sum_{1}^{n} m_{ia}, i \neq a \tag{2.26}$$

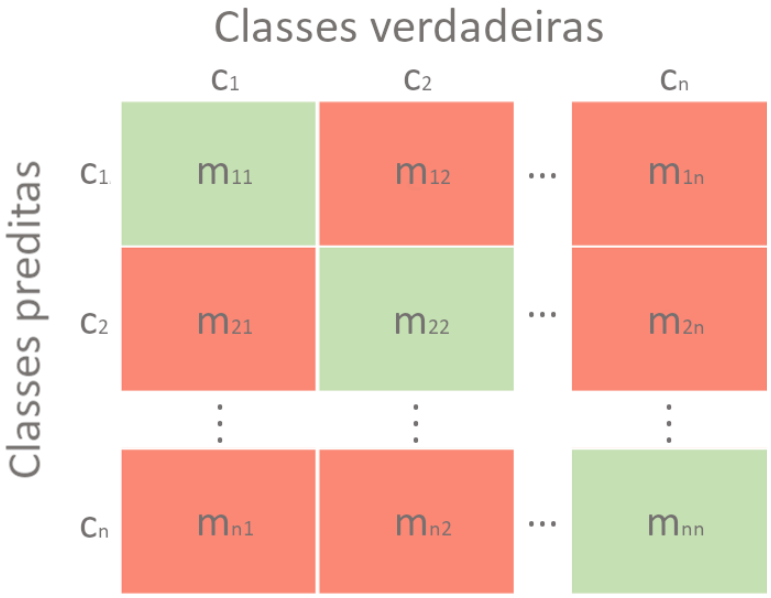


Figura 2.22: Matriz de Confusão para o caso multiclasses (Adaptado de Joydwip Mohajon (2020)).

A partir destes insumos, os cálculos de *Accuracy*, *Precision*, *Recall*, e *F-score* possuem algumas variantes:

***Micro-averaged*:** Soma os valores dos insumos (*TP*, *TN*, *FP*, *FN*) de todas as classes e aplica as mesmas equações do caso binário. Em problemas que cada item só assume uma única classificação, *Accuracy*, *Precision*, *Recall*, e *F-score* assumem todas o mesmo valor na variante *micro*.

***Macro-averaged*:** Para *Precision* e *Recall*, faz a média entre as classes (que pode ser ponderada ou não), e calcula o *F-score* a partir delas.

2.5.2 Geração de grafos sintéticos

Muitas redes do mundo real são compostas de forma que as propriedades estruturais de seus nós possuem informações intrínsecas sobre estruturas comunitárias do grafo. Dessa forma, muitos algoritmos relevantes da área de análises de grafos se utilizam dessas propriedades para realizar agrupamentos e classificações (Valejo, 2019).

Uma dificuldade na avaliação desses algoritmos, é o levantamento de dados de qualidade, e em quantidade suficiente para se variar atributos dos grafos e entender o comportamento de cada propriedade estrutural nos resultados obtidos (Romanetto, 2020). Essa demanda cria a necessidade de ferramentas para geração de grafos sintéticos, que permitam a comparação de soluções alternativas em um conjunto de diversas redes, geradas com ampla gama de recursos controlados, garantindo que a avaliação contemple várias

propriedades representativas, e assim, se obtenha uma avaliação precisa (Angelova et al., 2012).

Apresentada em (Valejo et al., 2020b), a técnica BNOC realiza a criação dos grafos utilizando o conceito de distribuição binomial negativa (Equação 2.27), para gerir a existência ou não de uma aresta entre dois vértices.

$$BN(N; n, s) = \binom{N+n-1}{n-1} s^n (1-s)^n \tag{2.27}$$

Sua vantagem frente a outros métodos de geração ou conjunto pré-determinados de grafos sintéticos está na possibilidade de geração de diferentes tipos de estruturas bi-partidas, permitindo variar parâmetros relativos ao tamanho dos grafos e das comunidades dentro de cada um dos tipos de vértices e a densidade das conexões.

Utilizando a decomposição de redes heterogêneas em redes bi-partidas já mencionada neste trabalho (seção 2.3), o mesmo trabalho apresenta a ferramenta HNOC, para criação de redes heterogêneas com maior número de partições. Dentre os parâmetros utilizados para a geração das redes na ferramenta HNOC, na Tabela 2.2 estão destacadas as relevantes para o presente trabalho, que são usadas no Capítulo 4.

Destacam-se aqui dois parâmetros importantes durante a criação da rede, a *dispersion*, (dispersão), e o *noise* (ruído) .

A ferramenta HNOC, inicialmente cria os vértices perfeitamente alinhados com as suas comunidades definidas. Para cada uma das partições do grafo k-partido, todos os nós de uma comunidade, são ligados com todos os nós de mesma comunidade em outras partições.

Após isso, arestas são removidas segundo o parâmetro da dispersão, que é utilizado para controlar a densidade das comunidades. Valores baixos de d produzem uma distribuição com muitos valores nulos, fazendo com que mais arestas sejam removidas.

Portanto, valores mais baixos de dispersão ($d \approx 0$) produzem comunidades mais esparsas, enquanto valores mais altos ($d \approx 1$) produzem comunidades mais densas.

O *noise*, ou ruído da rede, é a uma proporção diretamente relacionada com a dificuldade em se encontrar comunidades no grafo. Ele é utilizado da seguinte forma: após o HNOC usar a dispersão para definir quais arestas intracomunitárias serão removidos. As arestas restantes são substituídas, com uma probabilidade n, por uma aresta intercomunitária. Por exemplo, um parâmetro de ruído $n = 0.5$ faz com que cerca de 50% das arestas sejam religadas. Valores baixos de ruído ($n \approx 0$) implicam na inserção de poucos bordas intercomunitárias, gerando redes com comunidades claras e facilmente separáveis. À medida que o ruído aumenta, mais bordas intercomunitárias são inseridas, aumentando a sobreposição e fazendo com que os limites da comunidade fiquem mais difíceis de serem identificados. Em termos gerais, $n > 0.5$ produz redes com estruturas de comunidade

Tabela 2.2: Parâmetros da ferramenta HNOC (Valejo et al., 2020b) relevantes para o presente trabalho

Parâmetro	**Descrição**
-m, –layers	Número de tipos de vértices, ou camadas
-v, –vertices	Número de vértices em cada camada
-e, –schema	Esquema do grafo
-c, -communities	Número de comunidades nas camadas do grafo
-p, –probabilities	Probabilidades dos vértices pertencerem a cada uma das comunidades
-d, –dispersion	Número de vértices em cada camada
-n, –noise	Número de vértices em cada camada

mal definidas e esparsas, representando mais bordas intercomunitárias do que intracomunitárias. Escolhas de $n \in [0.1, 0.4]$ são mais adequados para aumentar a dificuldade de a tarefa de detecção da comunidade sem prejudicar drasticamente a estrutura do grafo.

Capítulo 3

Coarsening de grafos k-partidos usando label propagation

Redes k-partidas são uma classe de redes encontrada com frequência no mundo real, onde os nós podem ser divididos em conjuntos disjuntos. Comumente são encontradas em representações de características de uma determinada classe. Por exemplo, em uma rede de publicação, os artigos poderiam se conectar com os autores que os escreveram, com os termos presentes no seu texto, e com o congresso em que foram publicados (Newman, 2001; Romanetto, 2020).

Levantamentos da literatura recentes apontam a como lacunas importantes a serem investigadas a heterogeneidade de grafos e a busca pela escalabilidade de métodos nessas redes (Zhou et al., 2020; Liu et al., 2018; Wu et al., 2021). Uma estratégia identificada na literatura para tratar este problema de escalabilidade, é o uso de técnicas de *coarsening* (Walshaw, 2001; Sakellaridi et al., 2008; Zhu et al., 2012; Minatel et al., 2018). Recentemente, estes métodos foram provados válidos no problema de escalar algoritmos de classificação (Chen et al., 2017; Liang et al., 2020), porém estes apenas em grafos homogêneos.

A partir disso, foi definida como hipótese a ser verificada no presente trabalho:

Métodos de coarsening poderiam ser utilizados como forma de reduzir grafos heterogêneos, com o intuito de economia de espaço de armazenamento e escalabilidade de métodos aplicados a eles, se obtendo baixas perdas de qualidade em problemas de classificação.

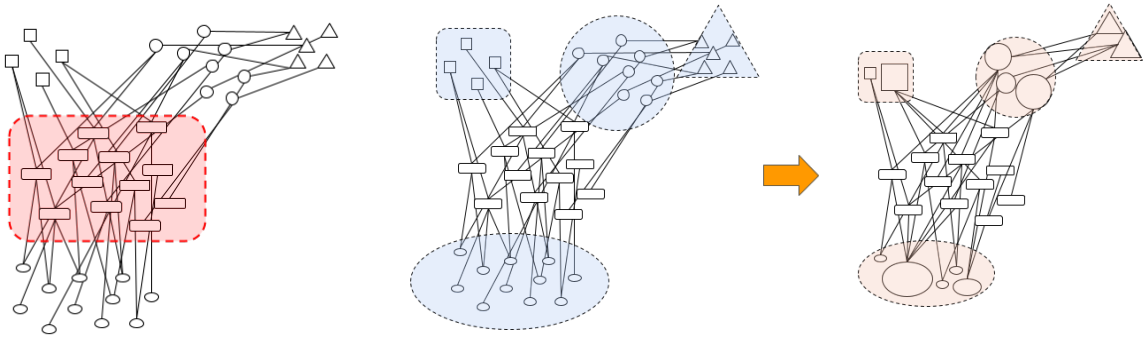
Para verificá-la, definiu-se duas questões de pesquisa a serem respondidas:

1. Quais os níveis de economia de armazenamento e tempo de classificação proporcionados por esses procedimentos *coarsening* se aplicados em grafos k-partidos?

2. Quais os impactos da redução nos grafos em suas métricas de qualidade de classificação?

Com o objetivo de responder estas questões, neste capítulo será apresentada a proposta de um algoritmo de *coarsening* que realiza a redução de grafos k-partidos (Figura 3.2) visando uma posterior tarefa de classificação.

A ideia central do algoritmo consiste em, dada uma partição a qual se deseja classificar, utilizar dados de rótulos conhecidos nesta partição para guiar a redução da quantidade de vértices e conexões. Como os métodos de classificação em grafos possuem complexidade normalmente associada ao número de vértices e arestas, a intenção deste procedimento é reduzir o tempo de execução da classificação, e assim poder averiguar os impactos na qualidade dos resultados.



(a) (b)

Figura 3.1: Exemplo de, dada uma partição que se deseje classificar (marcada de vermelho em (a)), efetua-se a redução das outras partiçÕes (b)

3.1 Notações

A Tabela 3.1 descreve as notações necessárias para compreensão da descrição do algoritmo proposto neste capítulo.

Tabela 3.1: Notações adotadas no presenta capítulo

Notação	**Descrição**
$G(V, E)$	grafo
$V = \{v_1, v_2, ..., v_{\|V\|}\}$	conjunto de vértices
E	conjunto de arestas
$\mathcal{N}(v_i)$	vizinhança, conjunto de vértices adjacentes ao vértice v_i
A	matriz de adjacências
$w(v_i, v_j)$, W	mapeamento dos pesos das arestas
$deg(v_i)$	grau do vértice v_i
$d_G(i, j)$	distância entre os vértices v_i e v_j
$V_i \subset V$	i-ésima partição do conjunto de vértices V
$V^L \subset V$	conjunto vértices rotulados
$V^U \subset V$	conjunto vértices não rotulados
$C = c_1, c_2, ..., c_n$	conjunto de classes possíveis para rotulação dos vértices
Y	distribuição dos rótulos nos vértices
$D = diag\{D_{ii} = \sum_j W_{ij}\}$	matriz diagonal dos graus
T	matriz de transição para propagação
$\mathcal{B}$	matriz bi-adjacente normalizada
$\mathcal{T}$	numero máximo de iterações da propagação
d	*dispersion* ou dispersão do grafo
n	*noise* ou ruído do grafo
$V_a \subset V$	partição alvo do problema de classificação em grafos k-partidos
$S(G)$	esquema das meta-relações de um grafo $G(V, E, W)$
S_i	vértice do esquema $S(G)$, que representa a partição $V_i \subset V$
P_i	menor meta-caminho entre S_i e S_a, onde S_a é o nó representante da partição alvo V_a no esquema $S(G)$
$G_i = (V_i, E_i)$	subgrafo de G contendo os vértices de V_i e as arestas incidentes em um destes vértices
$G_i^c = (V_i^c, E_i^c)$	subgrafo G_i após o procedimento de *coarsening*
$C(v) \in C \mid v \in V_a^L$	rótulo de v conhecido inicialmente
$V_p \subset V$	partição propagadora
$V_r \subset V$	partição receptora
$\mathcal{Y}^{(k)}(v) \mid v \in V$	rótulo de v em uma dada iteração k

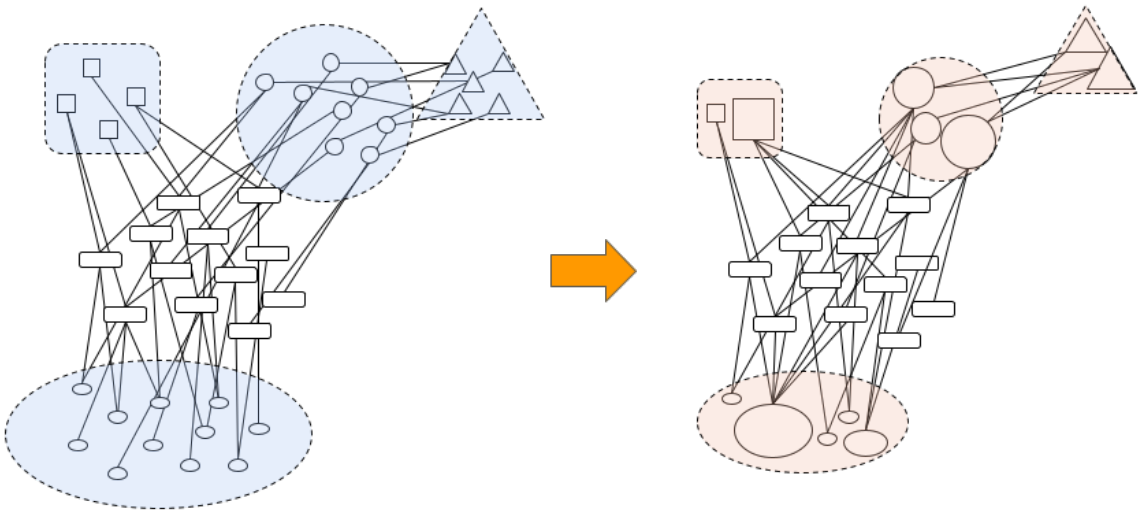


Figura 3.2: Exemplo de grafo antes e após o procedimento, reduzindo todas as partições, exceto a que se deseja classificar

3.2 Método de coarsening

Dentre os algoritmos de *matching* citados na Seção 2.4 foi escolhido o baseado em *label propagation*, proposto por Meyerhenke et al. (2014). Essa escolha se deve a dois motivos, primeiramente porque já foi utilizado com sucesso para o caso de grafos bi-partidos (Valejo et al., 2021), dessa forma atende a decomposição mencionada na seção 2.3 (Figura 3.3), que permite a utilização de algoritmos para grafos bi-partidos, realizando a sua extensão para grafos k-partidos (Zhu et al., 2016; Deng et al., 2017; Valejo et al., 2020b; Romanetto, 2020).

Um segundo motivo, é que, diferentemente destes dos trabalhos de Meyerhenke et al. (2014) e Valejo et al. (2021), que tratavam do problema de detecção de comunidades, o presente trabalho busca resolver um problema de classificação, onde é presumida a existência de dados previamente rotulados. Esse fato cria uma oportunidade de alteração que consiste em incorporar estes rótulos no procedimento de *matching* por *label propagation*, com o objetivo de que a escolha dos nós a serem condensados ocorra não só influenciado pelas características topológicas do grafo, mas também pela classificação já conhecida. O procedimento é realizado por pares de partições, uma partição $\mathcal{V}_p$ denominada partição propagadora e a outra $\mathcal{V}_r$ partição receptora, de forma semi-síncrona. A figura 3.4 mostra o procedimento passo a passo para uma bi-partição.

Para cada etapa, o procedimento pode ser descrito pelos seguintes passos:

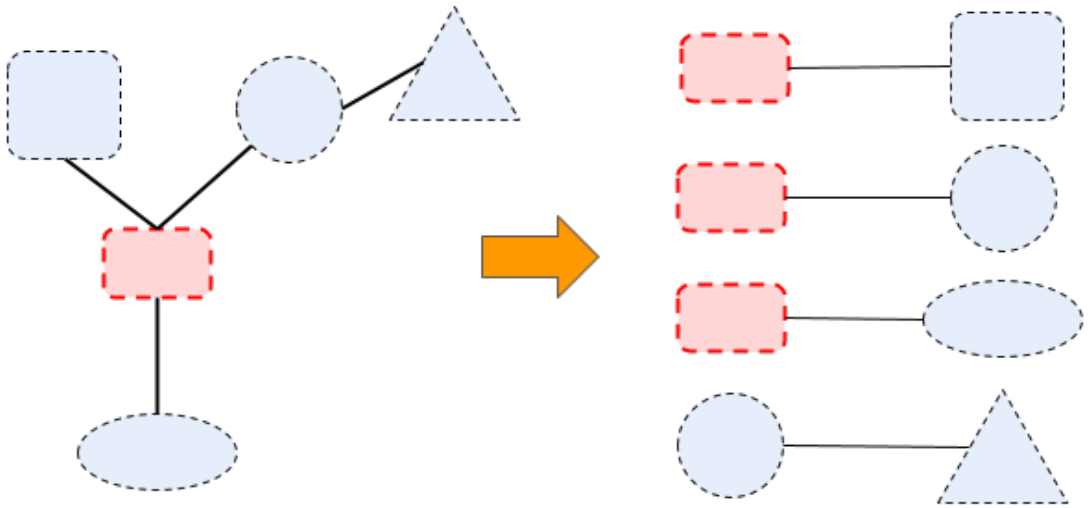


Figura 3.3: Decomposição de grafo k-partido em bi-partições.

1. Inicialize os rótulos dos vértices na partição propagadora,

$$\mathcal{Y}^{(0)}(v) = \begin{cases} C(v) & \text{se } v \in V_p^L \\ l_v & \text{caso contrário} \end{cases} \tag{3.1}$$

ou seja, atribua o próprio rótulo caso conhecido na partição propagadora, ou crie um novo rótulo diferente para cada vértice onde é desconhecido. Aqui, cada entrada de $\mathcal{Y}$ é um vetor com uma posição para cada rótulo obtido na inicialização.

2. Ordene, sem perda de generalidade, os vértices do subgrafo de G formado por $\mathcal{V}_p$ e $\mathcal{V}_r$, posicionando os vértices de $\mathcal{V}_p$ no início. E construa uma matriz de transição T desse subgrafo ordenado

$$T = D^{-1/2} A D^{-1/2} \tag{3.2}$$

onde A corresponde a matriz de adjacências e D é a matriz diagonal dos graus dos vértices do subgrafo ordenado. Essa normalização simétrica é feita por motivos de convergência da propagação, como feito em Zhou et al. (2003). Já o uso da matriz de graus tem como objetivo limitar a influência de vértices *hubs*, como explicado em Valejo et al. (2021).

3. O próximo passo de um algoritmo de *label propagation* seria realizado pela atualização dos rótulos por agregação dos vértices vizinhos (Equação 2.14). Porém, como

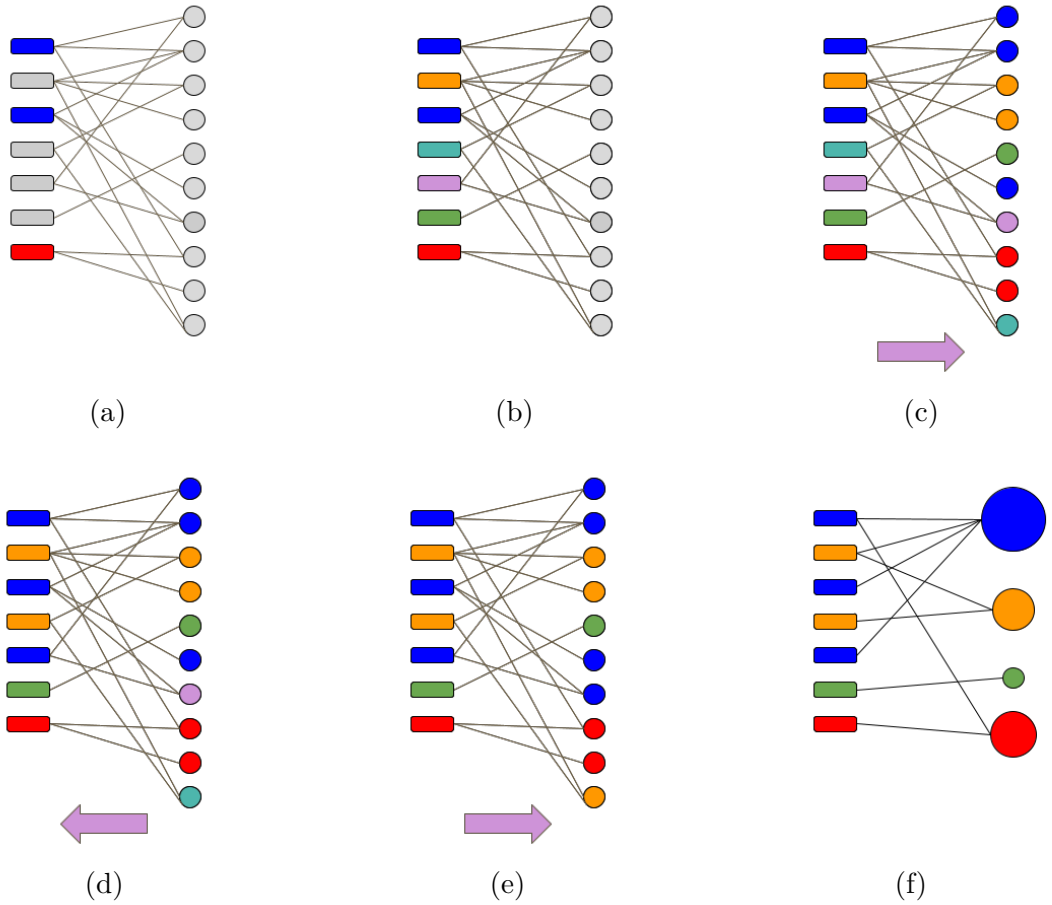


Figura 3.4: O procedimento para uma bi-partição se inicia com a atribuição dos rótulos conhecidos na partição propagadora (a), em seguida, os vértices da propagadora não rotulados recebem um rótulo diferente para cada um (b). A propagação segue de forma semi-síncrona, iniciando no sentido da propagadora para a receptora (c), e alternando com outro sentido (d), até a convergência (e), quando então os vértices da partição receptora, que possuírem rótulos iguais, são agrupados (f).

indicado no trabalho de Liu and Murata (2009), realizar esta atualização em matrizes bi-partidas pode levar a não convergência por oscilação. Para se evitar este problema o passo de propagação deve ser realizado por *cross-propagation*, método também usado no caso bi-partido por Valejo et al. (2021). No caso deste trabalho será primeiro na direção de V_p para V_r (Equação 3.4), e em seguida na direção contrária (Equação 3.5).

Como a matriz de transição T é simétrica, e foi ordenada pela bi-partição, pode ser decomposta em função da matriz bi-adjacente normalizada $\mathcal{B}$, descrita na Equação 3.3.

$$T = \begin{bmatrix} 0 & \mathcal{B} \\ \mathcal{B}^t & 0 \end{bmatrix} \tag{3.3}$$

Dessa forma, a atualização por *cross-propagation* poderia ser representada por:

$$\begin{bmatrix} \mathcal{Y}_{\mathcal{V}_p}^{(i+1)} \\ \mathcal{Y}_{\mathcal{V}_r}^{(i+1)} \end{bmatrix} = \begin{bmatrix} \mathcal{I} & 0 \\ \mathcal{B}^t & 0 \end{bmatrix} . \begin{bmatrix} \mathcal{Y}_{\mathcal{V}_p}^{(i)} \\ \mathcal{Y}_{\mathcal{V}_r}^{(i)} \end{bmatrix} = \begin{bmatrix} \mathcal{Y}_{\mathcal{V}_p}^{(i)} \\ \mathcal{B}^t.\mathcal{Y}_{\mathcal{V}_p}^{(i)} \end{bmatrix} \tag{3.4}$$

$$\begin{bmatrix} \mathcal{Y}_{\mathcal{V}_p}^{(i+1)} \\ \mathcal{Y}_{\mathcal{V}_r}^{(i+1)} \end{bmatrix} = \begin{bmatrix} 0 & \mathcal{B} \\ 0 & \mathcal{I} \end{bmatrix} . \begin{bmatrix} \mathcal{Y}_{\mathcal{V}_p}^{(i)} \\ \mathcal{B}^t.\mathcal{Y}_{\mathcal{V}_p}^{(i)} \end{bmatrix} = \begin{bmatrix} \mathcal{B}.\mathcal{B}^t.\mathcal{Y}_{\mathcal{V}_p}^{(i)} \\ \mathcal{B}^t.\mathcal{Y}_{\mathcal{V}_p}^{(i)} \end{bmatrix} \tag{3.5}$$

este procedimento é repetido até a convergência, ou até que um número máximo de iterações, $\mathcal{T}$, atue como critério de parada

$$\mathcal{Y}_{\mathcal{V}_r} = \lim_{k \to \mathcal{T}} \mathcal{B}^t (\mathcal{B}.\mathcal{B}^t)^{(k-1)} \mathcal{Y}_{\mathcal{V}^1}^{(0)} \tag{3.6}$$

4. Em seguida, é atribuído a cada vértice de V_r o rótulo de maior predominância:

$$y_i = \arg\max_j \mathcal{Y}_{ij} \tag{3.7}$$

5. Por fim, vértices com o mesmo rótulo são agrupados em um super-vértice, o processo é repetido até se atingir um nível de redução determinado ou até que o número de nós não seja mais reduzido entre uma iteração e a próxima.

3.3 Determinação das partições para o coarsening

Como citado na seção anterior, a intenção em se usar um método de *coarsening* para grafos bi-partidos, é usar a extensão para grafos k-partidos explicada na seção 2.3. O primeiro ponto a se elencar no procedimento proposto, é que decidiu-se por realizar o *coarsening* apenas nas partições não-alvo. O motivo dessa escolha é se manter individualizada a representação dos vértices da partição alvo, onde classificações executadas no grafo reduzido, já encontrariam a resposta do problema para o grafo completo, sem ser necessário algum mapeamento posterior. Quanto ao possível impacto desta escolha nas economias buscadas com a redução dos grafos, em levantamento realizado com em centenas de redes reais disponibilizados em Leskovec and Krevl (2014), foi identificado que o número de arestas dos grafos, é, em média, centenas de vezes maior do que o número de vértices, limitando a perda de compressão causada por essa escolha.

Com o algoritmo de *coarsening* definido para cada bi-partição, é necessário definir a estratégia de seleção dos pares que serão utilizados em cada etapa. Uma abordagem trivial seria utilizar todos os pares possíveis entre partições. Porém essa abordagem pode levar a um elevado número de procedimentos em grafos cujo esquema possui alta conectividade, culminando em uma complexidade quadrática no número de vértices (Zhu et al., 2016). Além disso, a existência de ciclos no esquema do grafo levaria a propagação de informações repetidas. Dessa forma, buscou-se identificar, para cada uma das partições, qual seria uma boa escolha de partição vizinha para realizar o procedimento.

Uma forma de limitar o número de pares utilizados, é identificar caminhos no seu esquema que são relevantes para o problema tratado (Luo et al., 2014). No caso do problema em questão, a única partição que no início do problema possui informações de rótulos é a partição alvo. Logo, um bom guia para o procedimento de *coarsening* seria propagar as informações da partição alvo em direção as outras, para assim aproveitar a alteração proposta na seção anterior de se incluir os rótulos já possuídos na fase de *matching*.

Como apontado em Gupta et al. (2017), caminhos menores tendem a ter mais chance de representar uma relação de alta influencia, por isso optou-se pelo menor meta-caminho entre essas duas partições.

Alguns trabalhos optam por usar caminhos definidos na estrutura para estender o conceito de vizinhança entre nós, como explicado na seção 2.3. Que pode ser feita pela transformação do grafo (Opsahl, 2010; Padrón et al., 2011; Valejo et al., 2017). Porém, como se pretende realizar o *coarsening* em todas as partições não-alvo, escolheu-se realizar o procedimento sincronicamente, partição por partição ao longo do caminho. Ainda, optou-se por uma seleção dos pares de partição, de forma radial a partir da partição alvo, fazendo o *coarsening* das partições a 1-*hop* de distancia (vizinhas) da alvo, em seguida as

a 2-*hops*, e assim por diante. E sempre usando como a outra partição envolvida, a vizinha dentro do meta-caminho partindo da partição alvo.

Repare na Figura 3.5, que esta ordem escolhida, além de propagar, na topologia das partições, informações de rótulo presentes inicialmente na partição alvo, faz com que a partição vizinha escolhida para o *coarsening*, sempre já tenha passado por uma redução (exceto na primeira iteração). Esse fato faz com que o procedimento seja acelerado, já que como será mostrado em Valejo et al. (2021), a complexidade é aproximadamente linear com o número de vértices e arestas envolvidos na bi-partição. Além disso, essa ordem é um possível ponto de otimização por paralelismo, pois para cada nível de distância para a alvo tratado, as propagações seriam independentes.

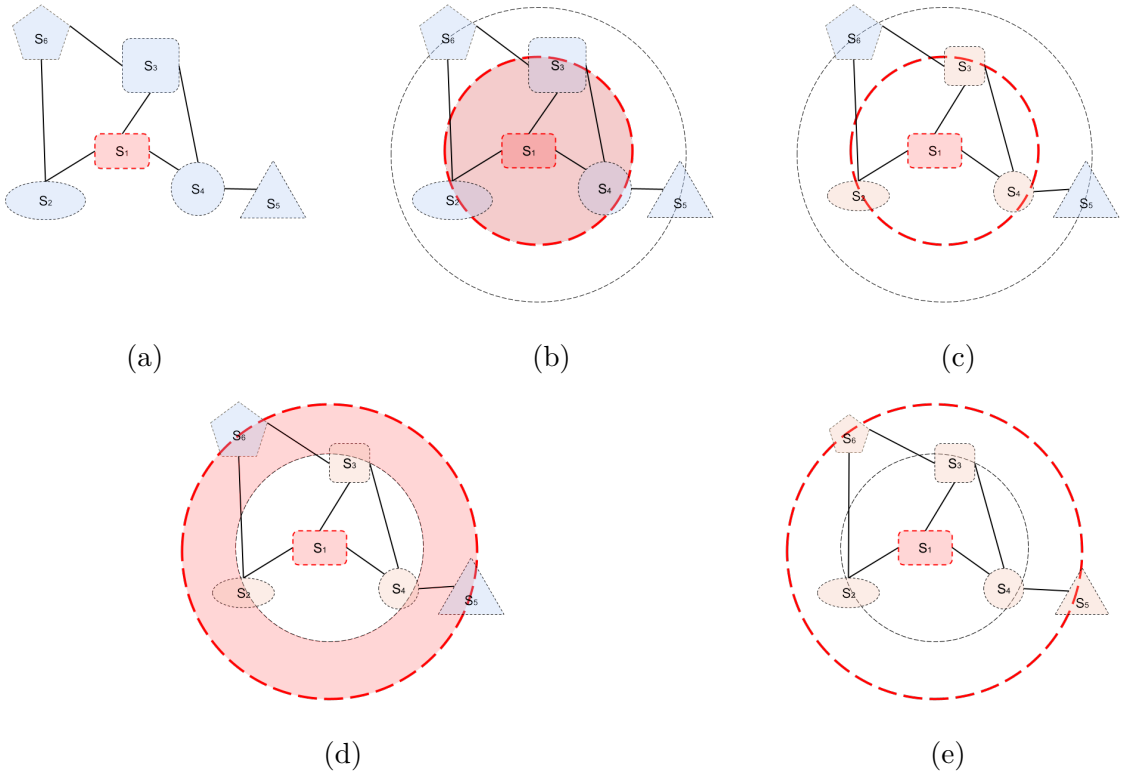


Figura 3.5: Ordem de execução do *coarsening* de forma radial em relação a partição alvo $S_a = S_1$. De b para c o *coarsening* ocorre nos vizinhos de S_1, que recebem influência apenas da partição alvo, e em um segundo momento, de d para e as partições a um nível a mais de distância de S_a recebem influência apenas de suas partições guia, que carregam em si informações estruturais de S_a.

3.4 Algoritmo

Seja $S(G)$ o esquema de um grafo k-partido G (exemplo na Figura 3.6), onde cada nó S_i representa uma das k partições $(V_1, V_2, ..., V_k)$ de G.

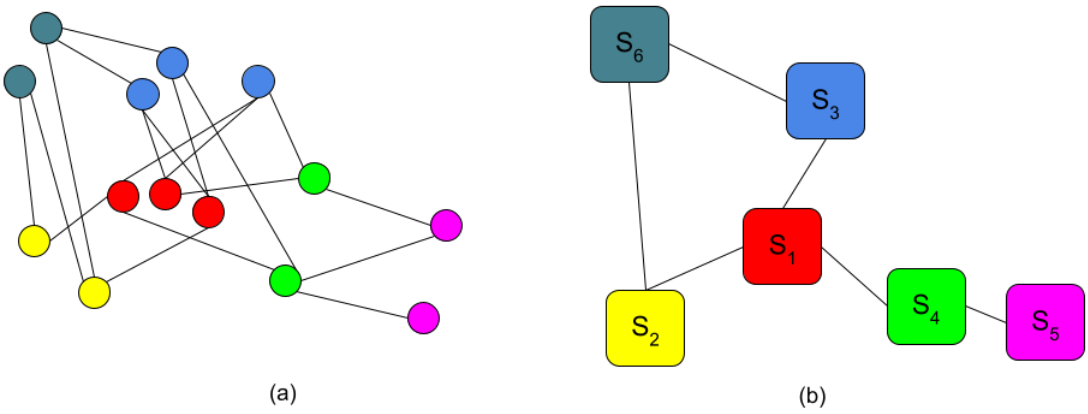


Figura 3.6: Exemplo de identificação de esquema de um grafo para aplicação do algoritmo: (a) grafo G k-partido a ser tratado pelo problema, cada uma das partições tem seus nós representados por uma cor $V = \{V_1 \cup V_2 \cup V_3 \cup V_4 \cup V_5 \cup V_6\}$ onde deseja-se classificar os nós vermelhos; (b) esquema $S(G)$ do grafo, conde $S_a = S_1$

Identifica-se inicialmente a partição-alvo V_a do grafo, como sendo aquela que deseja-se classificar, sendo S_a o nó no esquema correspondente a esta partição (Figura 3.7). As outras partições do grafo serão denominadas "não-alvo".

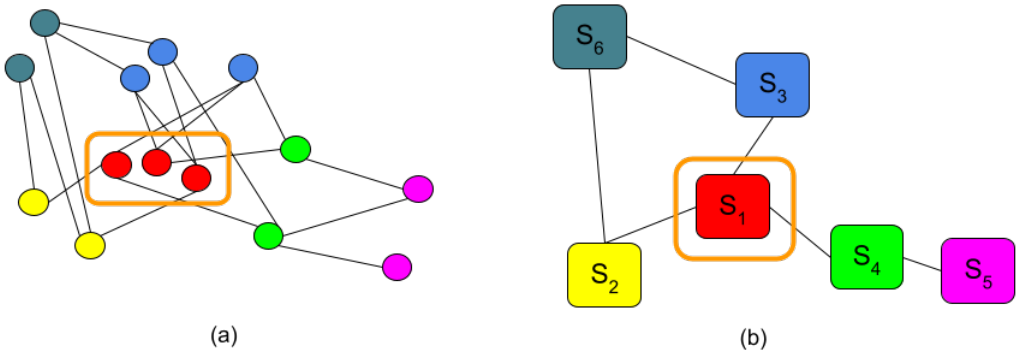


Figura 3.7: Exemplo de identificação da partição alvo do grafo (a) e de seu nó correspondente no esquema (b)

Para cada partição não alvo S_i, denota-se por P_i o menor meta-caminho partindo de S_a e chegando em S_i, onde a é o índice da partição alvo do problema de classificação.

No caso de meta-caminhos mínimos múltiplos, é escolhido o primeiro a ser encontrado. A partir disso, para cada S_i, seleciona-se uma outra partição $S_{g(i)}$, dita **partição guia**, que é a partição há uma distância de *1-hop* de S_i em P_i. As Figura 3.8 e Figura 3.9 exemplificam este procedimento.

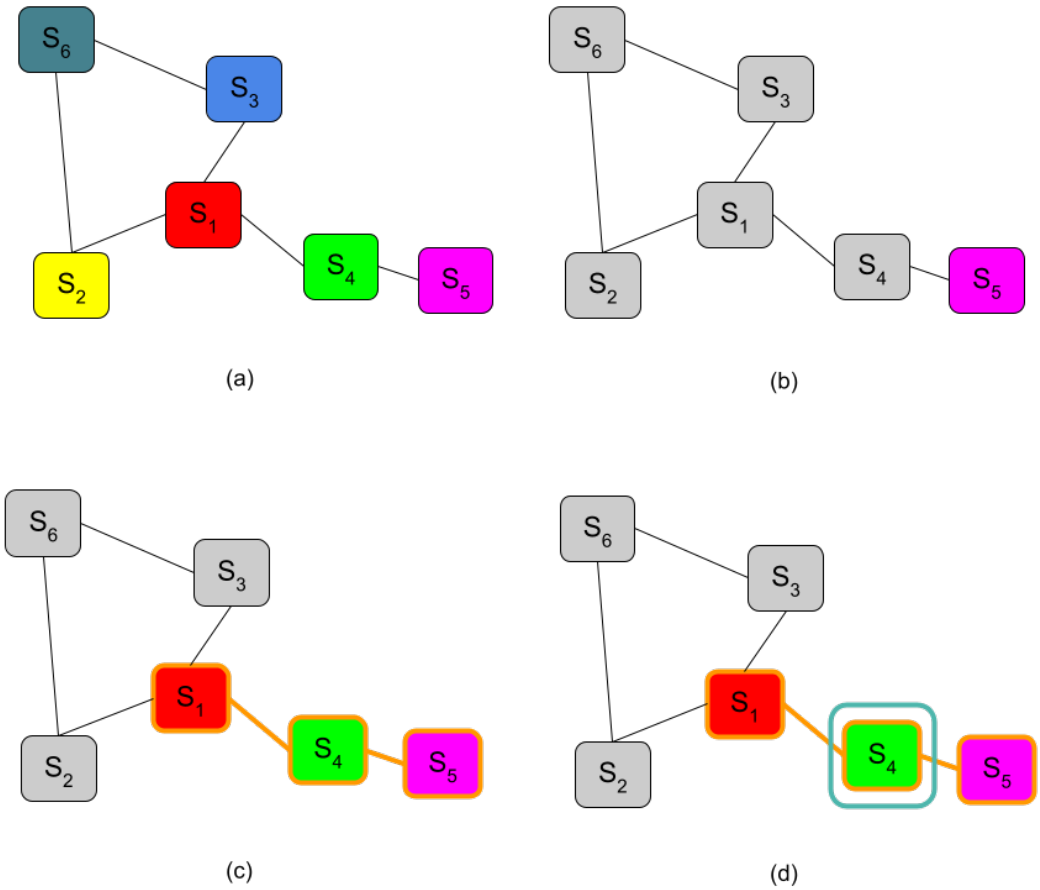


Figura 3.8: Exemplo de identificação da partição guia da partição não-alvo do esquema em (a); Identificada a partição não-alvo S_5 (b), traça-se P_5 (c) o menor meta-caminho entre S_5 e S_1 (alvo); Por pertencer a P_5 e estar a um-hop de distância de S_5, a partição S_4 é identificada como sendo sua partição guia (d)

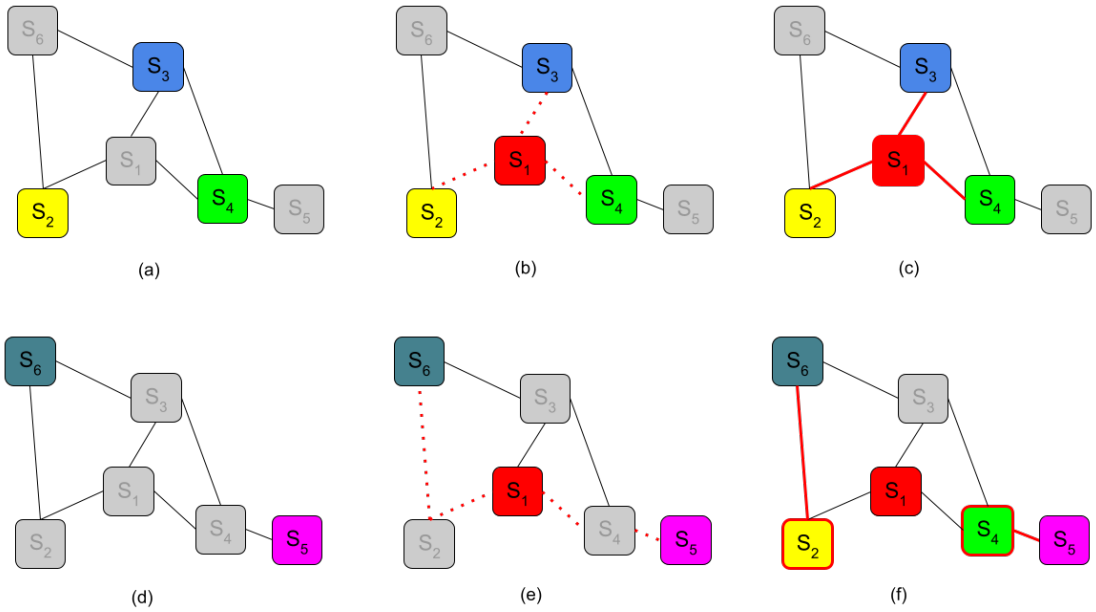


Figura 3.9: Exemplo de extração das partições guia: (a) selecionadas as partições S_2, S_3 e S_4; (b) encontram-se P_2, P_3 e P_4 que são o caso trivial de menor caminho para $S_a = S_1$; (c) identifica-se a partição guia das três como sendo a própria S_1; (d) selecionadas as partições S_5 e S_6; (e) encontra-se P_5 e P_6, menores meta-caminhos entre S_5 e S_1 e entre S_6 e S_1, respectivamente; (f) identifica-se a partição guia de S_5 como sendo S_4 e a guia de S_6 como sendo S_2.

Algoritmo 1: Procedimento de *coarsening* das partições não-alvo

```
   Dados: G(V,E,W), S(G)
1  bfsList ← [S_a]
2  adicionados ← [S_a]
3  guia ← {}
4  while bfsList.length ≠ 0 do
5      particao ← pull(bfsList)
6      forall s ∈ N(particao) do
7          if s ∉ adicionados then
8              adicionados.push(s)
9              bfsList.push(s)
10             guia[s] ← particao
11     if particao ≠ S_0 then
12         sgPart ← getSubgraph(G, particao)
13         sgGuia ← getSubgraph(G, guia[particao])
14         sgPart ← coarseningBiPartido(sgPart, sgGuia)
15         atualizar(G, particao, sgPart)
16 return G
```

A aplicação do *coarsening* nas partições não-alvo, segue a ordem de uma busca em largura em S partindo-se de S_a (*loop* da linha 4). Com isso busca-se substituir o subgrafo $G_i = (V_i, E_i)$, por sua versão condensada $G_i^c = (V_i^c, E_i^c)$ composta pelos super-nós e super-arestas obtidas no processo de *coarsening*.

É importante ressaltar que, mesmo que a partição guia seja usada como apoio para o procedimento do *coarsening* bi-partido (linha 14), apenas a partição de índice i será atualizada neste momento em G (linha 15). Nesta etapa procede-se a aplicação do *coarsening* por *label propagation* descrito em 3.2.

Depois de realizar o *coarsening* em todas as partições não-alvo do grafo, O algoritmo retorna o grafo resultante G^c.

3.5 Trabalhos Relacionados

Esta seção destaca, dentre os trabalhos relacionados já mencionados no Capítulo 2, as semelhanças e diferenças para o algoritmo proposto neste capítulo.

Como principais semelhanças dos trabalhos de Chen et al. (2017) e Liang et al. (2020) com o algoritmo proposto no presente trabalho, estão o uso de métodos de redução de grafos para melhoria de escalabilidade, porém, os dois métodos são apenas aplicáveis a redes homogêneas. Além disso, apresentam métodos multiníveis, com um fluxo completo para ser usado em cada classificação, enquanto o proposto neste trabalho aborda a fase de redução como uma etapa prévia de armazenamento dos grafos.

Como já citado, Akyildiz et al. (2020) resolvem problemas relacionados ao tamanho dos grafos nos dois métodos supracitados, particionando-o de forma a possibilitar o processamento de partes menores do grafo separadamente. No caso do presente trabalho, o processamento por partes é realizado aproveitando a estrutura k-partida já presente no problema.

Os trabalhos de Valejo et al. (2020a) e sua extensão proposta em Valejo et al. (2021), são a principal inspiração para o método de *coarsening* apresentada neste Capítulo, que pode ser considerado como uma extensão de do proposto em Valejo et al. (2021) para estruturas k-partidas.

A semelhança do trabalho apresentado por Zhu et al. (2016), para o presente trabalho, está na realização de redução de um grafo k-partido por meio de *coarsening*. Já a diferença foi buscada para resolver sua complexidade quadrática já mencionada, se propondo aqui, um método de escolha de uma única partição para guiar o *coarsening*, e um procedimento baseado em propagação de rótulos, que tem complexidade aproximadamente linear (Valejo et al., 2021).

Capítulo 4

Análise dos efeitos do coarsening em grafos k-partidos

O método proposto no capítulo 3, foi utilizado como ferramenta em testes de classificação comparando os resultados obtidos nos grafos em suas versões originais, com versões reduzidas pelo método. A intenção foi a de, para entender melhor os efeitos de uma compressão feita por *coarsening*, mensurar qual a perda de qualidade na classificação conforme o grafo é diminuído.

No presente capítulo, a Seção 4.1 mostra a análise realizada em centena de milhares de grafos gerados sinteticamente, já a seção 4.2 mostra como as análises em redes sintéticas podem ser usadas para guiar o uso de *coarsening* em redes reais.

4.1 Análises em redes sintéticas

Foi adotada a geração de dados artificiais devido a dificuldade em se obter uma grande quantidade de dados rotulados. Para deixar os experimentos bastante variados, não se limitando a poucos exemplos, foi utilizada uma ferramenta de geração automática de grafos (seção 2.5.2). A ferramenta escolhida foi a HNOC (Valejo et al., 2020b), um gerador de grafos sintéticos desenvolvido com o intuito de auxiliar a análise de métodos de aprendizado em redes. As características que levaram a essa escolha foram a capacidade da ferramenta em variar:

- o número de elementos em cada partição;
- o número de classificações possíveis em cada partição;
- a probabilidade das classificações possíveis; e ainda,
- os níveis de ruído e dispersão, conforme explicados na seção 2.5.2.

Serão gerados grafos variando estas métricas baseando-se em diferentes topologias inspiradas em problemas do mundo real, em todas elas a partição alvo é representada pelos nós vermelhos nas figuras:

Topologia hierárquica em estrela

Este tipo de topologia, representado pela Figura 4.1a, é encontrado em problemas envolvendo redes de computadores, hierarquias empresariais, dentre outros. Ela é caracterizada pela presença de nós *hubs* em diferentes níveis (com grande quantidade de arestas em comparação aos outros).

Os nós a serem classificados neste exemplo serão as folhas, que seriam dispositivos de interface com o usuário no caso das redes de computadores, e funcionários sem funções de gerência nas empresas.

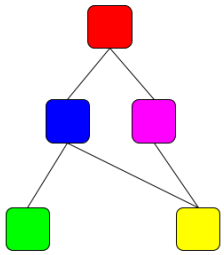
Topologia de Teia hierárquica

Topologia característica de redes biológicas, como por exemplo a cadeia alimentar. Sua principal características é a relativa ordem entre as partições, e a presença de maior quantidade de vértices nos níveis inferiores (Figura 4.1b).

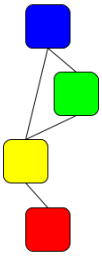
Da perspectiva de um tipo do topo da cadeia, este esquema se comporta de forma similar ao hierárquico, por este motivo escolheu-se um tipo da base para a classificação.

Topologia bi-partida

O esquema da rede bi-partida é composto apenas por duas partições. Este tipo de rede é muito explorado na literatura no problema de classificação textual (Rossi et al., 2012; Faleiros et al., 2016; Redmond and Rozaki, 2017). Neste caso, vértices da partição alvo representam documentos, e os da partição não-alvo, palavras (Figura 4.1c). As arestas nesse exemplo representariam a presença de uma palavra em um documento.

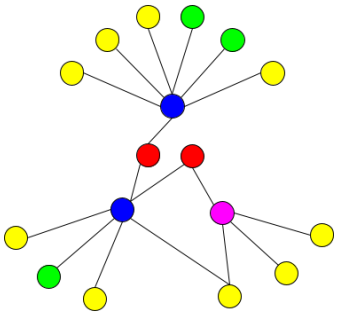


(a) Esquema hierárquico em estrela

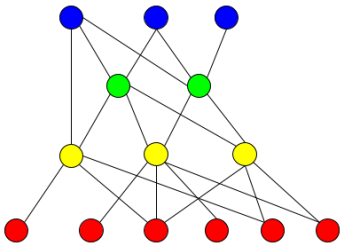


(b) Esquema de teia hierárquica

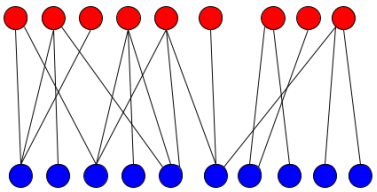
(c) Esquema bi-partido



(d) Grafo com hierarquia em estrela



(e) Grafo de teia hierárquica



(f) Grafo bi-partido

Figura 4.1: Topologias distintas usadas nos experimentos representadas por exemplos de grafos (d, e, f) e seus respectivos esquemas (a, b, c).

O algoritmo de classificação utilizado nos testes foi o GNetMine (Ji et al., 2010), por ser um algoritmo de referência na área de classificação heterogênea, e usado como base de comparação em diversos trabalhos (Luo et al., 2014; Zhi et al., 2015; Bangcharoensap et al., 2016; Faleiros et al., 2016; Luo et al., 2018; Ding et al., 2019; Romanetto, 2020). Os parâmetros λ_{ij} foram distribuídos igualmente e todos os parâmetros de confiança α_{ij} determinados iguais.

Todos os grafos gerados foram submetidos inicialmente a classificação pelo GNetMine e, dos resultados obtidos, foram calculadas as métricas de, *Precision*, *Recall*, *F-score* (nas suas variantes *macro*) e *Accuracy*. Como só é considerado um rótulo para cada vértice, as variantes *micro* assumiriam todas o valor de *Accuracy*. Também foram medidos os tempos necessários para o GNetMine classificar cada um dos grafos.

Os dados obtidos nesta classificação são utilizados como referência de comparação para as mesmas métricas calculadas a partir de classificações com o GNetMine e versões reduzidas dos grafos, obtidas após utilização do método proposto no Capítulo 3. Para todas as reduções, também é comparado o tamanho de armazenamento ocupado pelos arquivos que representam os grafos reduzidos com os originais.

Foram realizados testes com reduções de 0% (base comparativa) até 80%, variando também o número de rótulo dados inicialmente em 1%, 10%, 20% e 50%.

Para o primeiro experimento com dados sintéticos, os grafos foram gerados variando:

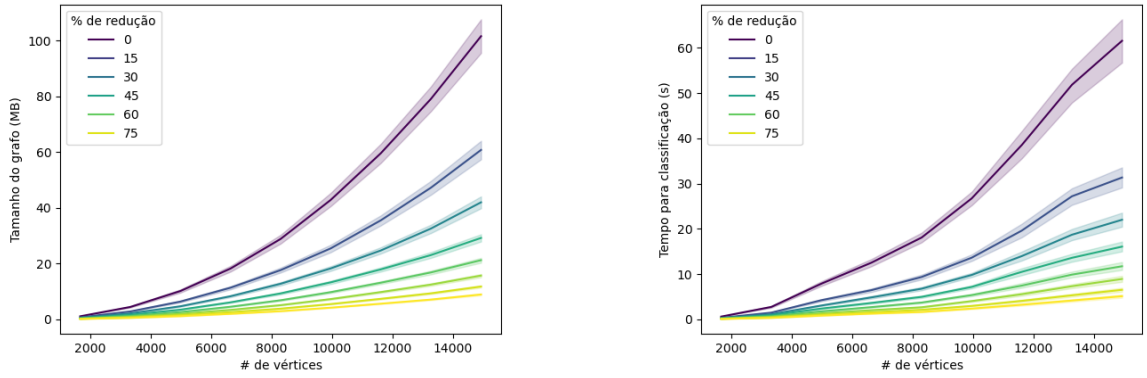
- o número de vértices de 2000 a 15000, em 9 configurações distintas para cada esquema;
- entre os três tipos de esquemas apresentados anteriormente (Figura 4.1);
- o número de classes possíveis variando entre 4 e 10;
- a dispersão em 9 valores entre 0.1 e 0.9; e
- o ruído variando em 9 valores entre 0.1 e 0.9.

Para cada uma das configurações descritas, foram gerados 10 diferentes grafos, chegando num total de mais de 130 mil grafos. A distribuição do número de vértices é igualitária entre as partições não-alvo, com cada uma delas tendo seu tamanho definido em 5 vezes o tamanho da partição alvo. Uma ressalva quanto a esta disposição dos dados, é que em grafos onde a partição alvo fosse muito maior do que outras partições, a economia de memória seria mascarada pelo arquivo dos vértices alvo ser uma constante, nesses casos seria indicada uma análise separada da economia considerando somente as partições não-alvo.

A primeira análise foi feita quanto aos principais objetivos da redução, que foram testes relativos à economia de memória, e tempo de classificação. A Figura 4.2 mostra a

evolução das métricas de espaço ocupado pelos arquivos que representam os grafos, e o tempo de necessário para classificação destes.

Os dados estão dispostos mostrando essa evolução para níveis distintos de reduções no número de vértices realizado pelo *coarsening*. Já a Tabela 4.1 apresenta estes dados dispondo a economia relativa destas duas métricas, para cada nível de redução, em relação ao grafo original não reduzido. Nela é possível perceber uma economia relativa maior nos primeiros níveis do *coarsening*, obtendo bons resultados já com apenas 20% de redução no número de vértices, isso provavelmente se dá pelos primeiros níveis já agruparem parte considerável dos vértices com maior número de conexões em comum.



(a) (b)

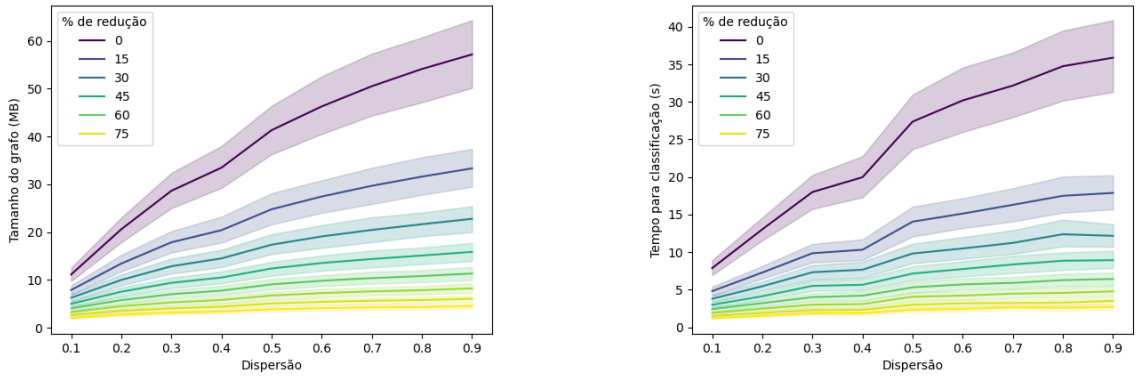
Figura 4.2: Variações do tamanho dos grafos em MB (a) e do tempo necessário para classificação dos mesmos (b) com o aumento do número de vértices nos grafos.

Tabela 4.1: Economia, em tamanho de armazenamento e tempo de classificação, das reduções dos grafos em comparação com suas formas originais

	Economia em Armazenamento			**Economia em tempo para classificação**		
# de vértices / **Redução**	2000	Média	15000	2000	Média	15000
20%	$32.87 \pm 0.48\%$	$36.92 \pm 0.47\%$	$38.75 \pm 0.39\%$	$39.76 \pm 0.84\%$	$45.35 \pm 0.89\%$	$47.19 \pm 0.82\%$
35%	$47.59 \pm 0.73\%$	$53.43 \pm 0.67\%$	$56.92 \pm 0.47\%$	$50.61 \pm 0.89\%$	$59.03 \pm 0.94\%$	$62.41 \pm 0.71\%$
50%	$59.88 \pm 0.80\%$	$65.29 \pm 0.71\%$	$69.51 \pm 0.47\%$	$59.39 \pm 0.82\%$	$68.57 \pm 0.87\%$	$72.09 \pm 0.66\%$
60%	$68.78 \pm 0.79\%$	$73.75 \pm 0.67\%$	$77.46 \pm 0.45\%$	$65.86 \pm 0.86\%$	$75.88 \pm 0.81\%$	$79.43 \pm 0.56\%$
80%	$84.41 \pm 0.54\%$	$88.12 \pm 0.43\%$	$90.11 \pm 0.30\%$	$79.30 \pm 0.75\%$	$88.12 \pm 0.59\%$	$90.73 \pm 0.33\%$

Como mencionado na seção 3.3, a quantidade de arestas em um grafo é normalmente muito superior a quantidade de vértices. Esse fato ensejaria em uma maior economia em grafos com maior densidade de arestas (menos esparsos). Para verificar isso, foi realizada uma análise das mesmas economias em relação a essa densidade, controlada pelo parâmetro de dispersão (seção 2.5.2). A Figura 4.3 mostra que, de fato, maiores densidades de arestas levam a grafos mais custosos tanto no quesito armazenamento,

quanto no tempo de classificação. Porém, também maior é a economia proporcionada pelo *coarsening* nesses casos, como mostra a Tabela 4.2.



(a) (b)

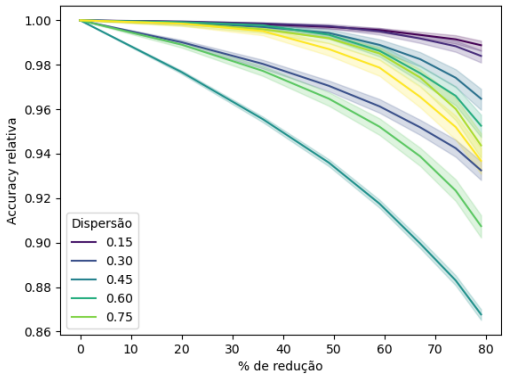
Figura 4.3: Variações do tamanho dos grafos em MB (a) e do tempo necessário para classificação dos mesmos (b) com a mudança na densidade de arestas.

Tabela 4.2: Comparação de economias com a densidade de arestas nos grafos

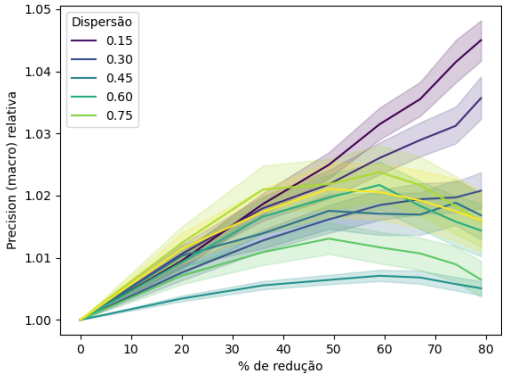
Redução	**Economia em Armazenamento**		**Economia em tempo para classificação**	
20%	$27.22 \pm 0.28\%$	$40.50 \pm 0.17\%$	$36.24 \pm 1.01\%$	$48.50 \pm 0.69\%$
35%	$35.85 \pm 0.52\%$	$58.28 \pm 0.24\%$	$48.05 \pm 0.98\%$	$63.21 \pm 0.70\%$
50%	$40.57 \pm 0.59\%$	$70.39 \pm 0.23\%$	$57.84 \pm 0.92\%$	$72.81 \pm 0.60\%$
60%	$51.34 \pm 0.57\%$	$78.60 \pm 0.20\%$	$64.51 \pm 0.89\%$	$80.12 \pm 0.50\%$
80%	$65.71 \pm 0.34\%$	$91.22 \pm 0.12\%$	$80.01 \pm 0.78\%$	$90.96 \pm 0.33\%$
Dispersão	0.1	0.9	0.1	0.9

Analisados os ganhos obtidos pela execução do *coarsening*, resta verificar quais os impactos da redução nas métricas de qualidade de classificações realizadas nos mesmos. Para complementar a análise realizada sobre a influência da dispersão na economia, foi realizado esse teste inicialmente variando a dispersão dos grafos (Figura 4.4). Nessa análise é possível perceber, que apesar do encontrado na análise anterior, de que dispersões baixas levariam a baixa economia de memória, em contraponto a isto, baixas dispersões levam a baixíssimas perdas nas métricas calculadas, levando inclusive a melhora delas em alguns casos.

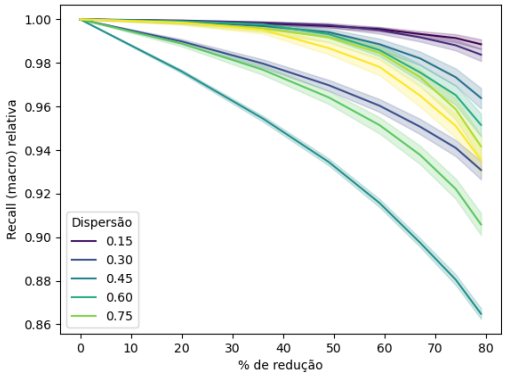
Em seguida, foi realizada uma análise sobre o efeito da redução em relação ao tamanho do grafo (Figura 4.5). É possível perceber que a perda de qualidade se acentua com o crescimento dos grafos, como mostra a Tabela 4.3. Por exemplo, a perda relativa, ao se reduzir o grafo entre os níveis de 20% para 80%, no *F-score* em um grafo de 2000 vértices, era de cerca de 5% (de 1% pata 6%). Já em grafos de cerca de 15000 vértices essa perda se acentua para em torno de 10%.



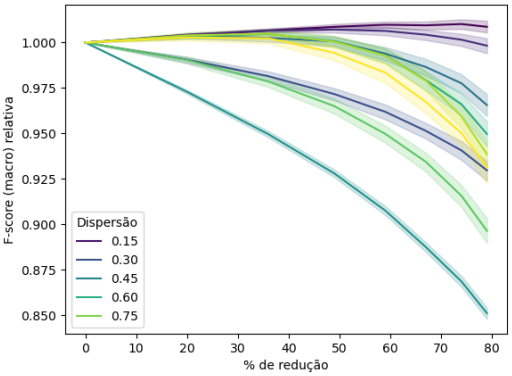
(a) % de redução × *Accuracy*



(b) % de redução × *Precision(macro)*



(c) % de redução × *Recall (macro)*

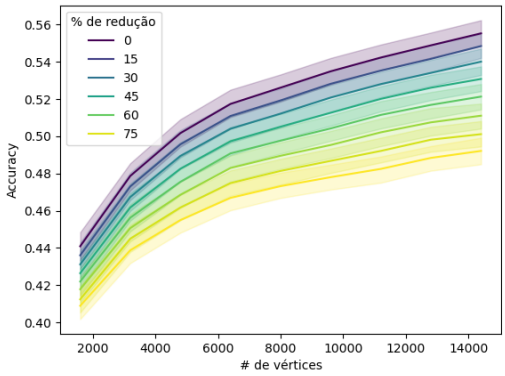


(d) % de redução × *F-score (macro)*

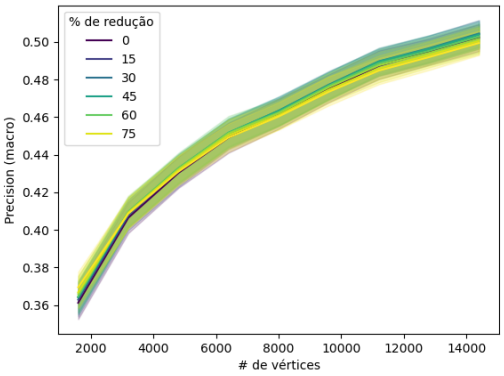
Figura 4.4: Variações nas métricas de *Accuracy* (a), *Precision (macro)* (b), *Recall (macro)* (c) e *F-score (macro)* (d) conforme o grafo é reduzido, para níveis distintos de dispersão.

# de vértices	**Redução**	***Accuracy***	***Precision (macro)***	***Recall (macro)***	***F-score (macro)***
2000	20%	$1.01 \pm 0.32\%$	$0.76 \pm 0.51\%$	$1.08 \pm 0.34\%$	$0.92 \pm 0.47\%$
	80%	$6.20 \pm 1.12\%$	$2.54 \pm 1.05\%$	$6.49 \pm 1.15\%$	$6.12 \pm 1.42\%$
15000	20%	$1.50 \pm 0.52\%$	$0.57 \pm 0.40\%$	$1.51 \pm 0.53\%$	$1.69 \pm 0.76\%$
	80%	$10.48 \pm 1.73\%$	$0.63 \pm 0.82\%$	$10.63 \pm 1.75\%$	$11.97 \pm 2.24\%$

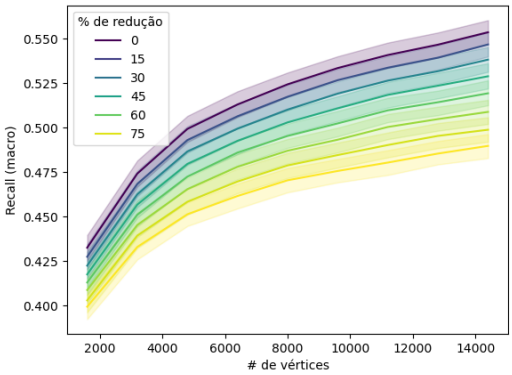
Tabela 4.3: Aumento da perda com a redução dos grafos acentuada pelo maior tamanho dos grafos (comparativo entre 2000 e 15000 vértices)



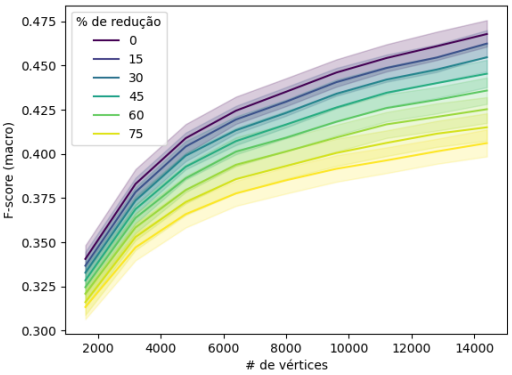
(a) # de vértices × *Accuracy*



(b) # de vértices × *Precision(macro)*



(c) # de vértices × *Recall (macro)*



(d) # de vértices × *F-score (macro)*

Figura 4.5: Métricas de *Accuracy* (a), *Precision (macro)* (b), *Recall (macro)* (c) e *F-score (macro)* (d) conforme o tamanho do grafo aumenta.

Dois fatos interessantes surgem dessa análise. Primeiro é a aparente estabilidade da métrica de *Precision* na sua modalidade *macro*. Como as classes neste experimento são distribuídas de forma aproximadamente igual, esse fato poderia ser um indício de que a técnica apresentada neste trabalho poderia ser melhor aproveitada em problemas onde falso negativos não são um problema mas deseja-se reduzir o número de falsos positivos. O segundo fato é a baixa perda apresentada para reduções em torno de 20%, em contraste com as altas economias de armazenamento e tempo de classificação observadas já nesse primeiro nível de redução.

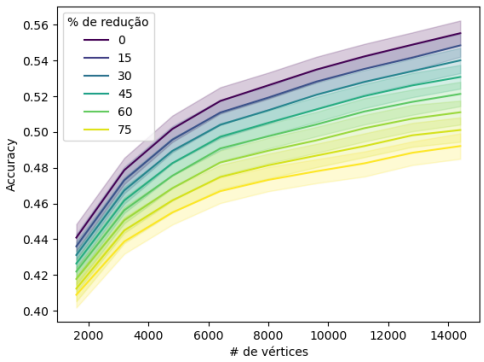
Para se investigar mais a fundo estes dois fatos, foi realizado um segundo experimento, aumentando o tamanho dos grafos até 100 mil vértices. Como os procedimentos tendem a demorar mais com o aumento dos vértices, neste experimento foi fixado o valor do ruído, e o número de grafos gerados para cada configuração diminuídas de 10 para 5 em grafos a partir de 10 mil vértices, e para 3 em grafos com mais de 60 mil vértices. O valor do ruído fixado foi em 0.3, para se obter grafos de mais difícil classificação, porém não degradados. Importante destacar que essa diminuição no número de testes deixa a média dos resultados menos estável, criando mais oscilações nos resultados obtidos.

Primeiramente foram comparados os resultados desse segundo experimento, limitando o tamanho dos grafos a 15000 vértices, com o primeiro experimento, com o intuito de verificar se as limitações impostas na geração dos grafos para o segundo alterariam o comportamento.

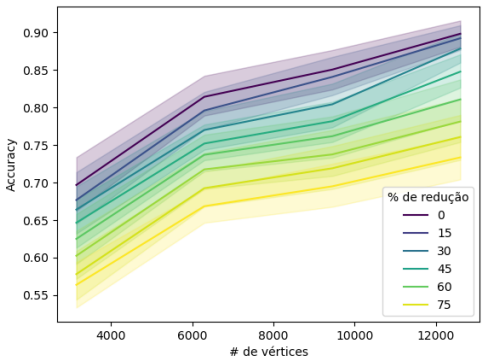
A Figura 4.6 compara o comportamento dos dois experimentos e a Tabela 4.4 faz a mesma análise da Tabela 4.3 para os grafos na configuração do experimento 2.

# de vértices	**Redução**	***Accuracy***	***Precision (macro)***	***Recall (macro)***	***F-score (macro)***
2000	20%	$2.55 \pm 0.40\%$	$0.76 \pm 0.14\%$	$3.10 \pm 0.46\%$	$3.85 \pm 0.62\%$
	80%	$15.86 \pm 1.61\%$	$4.02 \pm 0.40\%$	$18.31 \pm 1.82\%$	$19.32 \pm 1.92\%$
15000	20%	$0.64 \pm 0.24\%$	$0.33 \pm 0.14\%$	$0.64 \pm 0.24\%$	$1.00 \pm 0.33\%$
	80%	$20.76 \pm 2.47\%$	$4.06 \pm 0.79\%$	$20.64 \pm 2.46\%$	$24.57 \pm 3.08\%$

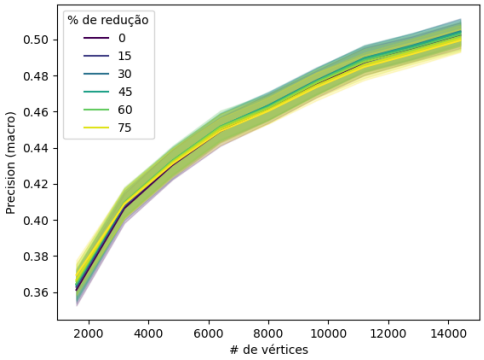
Tabela 4.4: Aumento da perda com a redução dos grafos acentuada pelo maior tamanho dos grafos



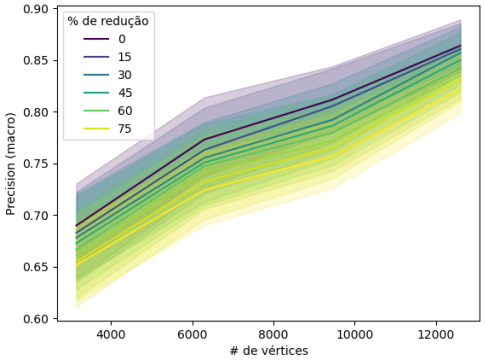
(a) # de vértices × *Accuracy*



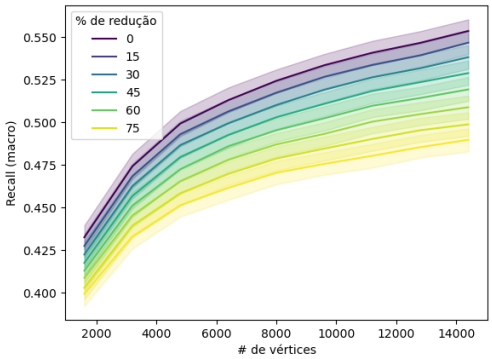
(b) # de vértices × *Accuracy*



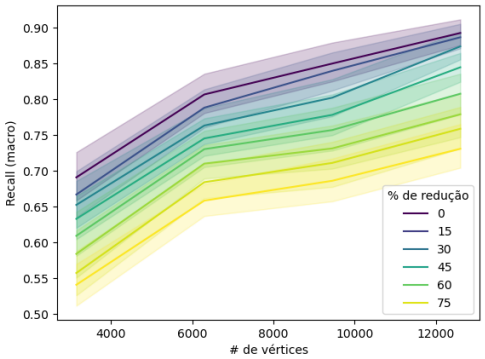
(c) # de vértices × *Precision(macro)*



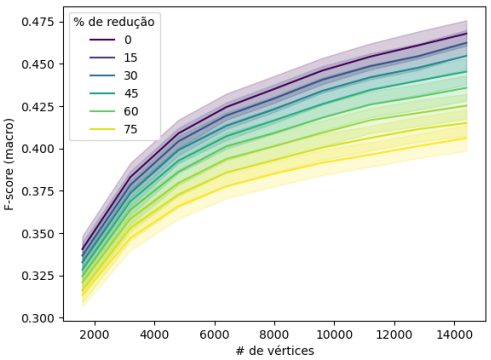
(d) # de vértices × *Precision(macro)*



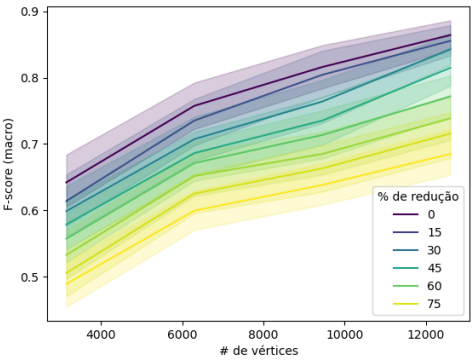
(e) # de vértices × *Recall (macro)*



(f) # de vértices × *Recall (macro)*



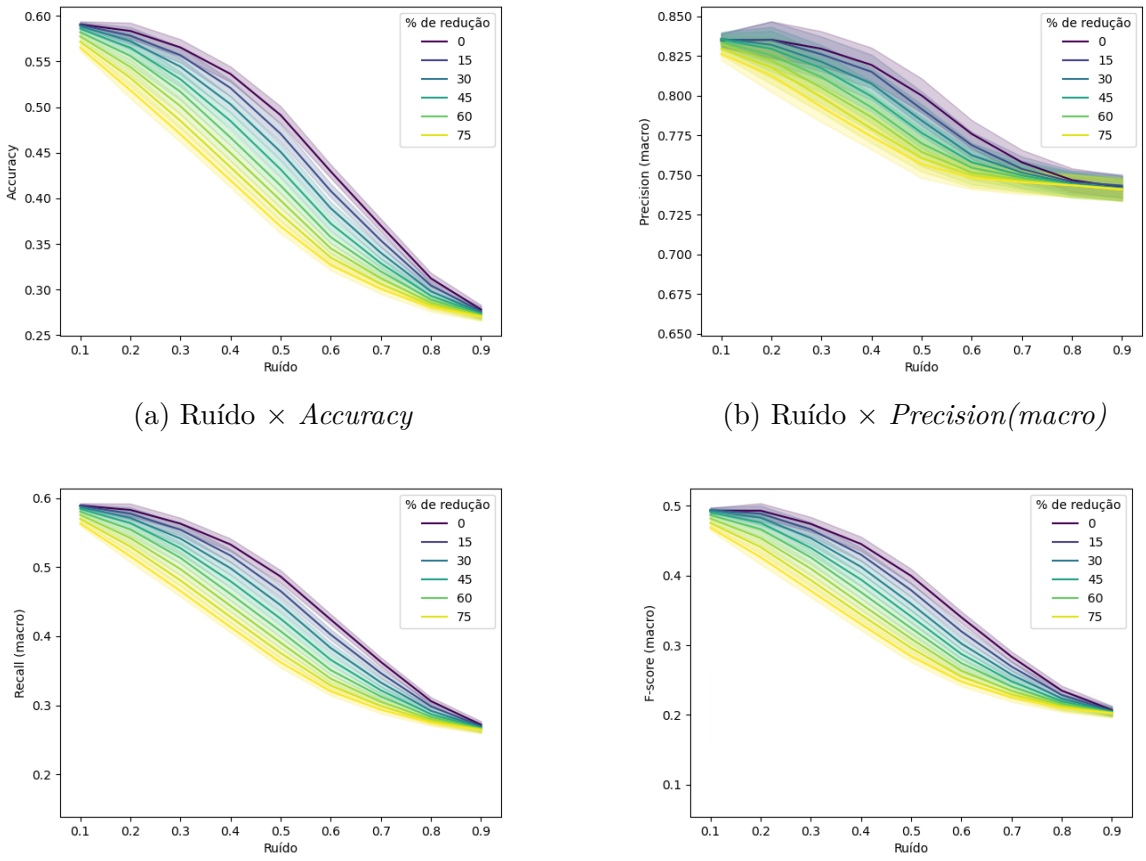
(g) # de vértices × *F-score (macro)*



(h) # de vértices × *F-score (macro)*

Figura 4.6: Comparativos entre experimentos 1 e 2, para mesmos tamanho de grafos, nas métricas de *Accuracy* (a, b), *Precision (macro)* (c, d), *Recall (macro)* (e, f), e *F-score (macro)* (g, h) conforme o tamanho do grafo aumenta.

Nessa comparação é possível perceber o mesmo comportamento global do primeiro experimento, com a diferenças de apresentar uma maior perda nas métricas com a redução em grafos pequenos. Isso gerou um questionamento se essa diferença se deve a fixação do ruído em um valor elevado, aumentando a discrepância entre as métricas nos diferentes níveis de redução. Portanto, decidiu-se realizar uma análise extra, no sentido de verificar a sensibilidade do procedimento de *coarsening* a respeito no nível de ruído na rede (seção 2.5.2), que representaria distúrbios na qualidade dos dados usados para a classificação, dada pela proporção de arestas do grafo conectariam vértices com classificações diferentes. Esse parâmetro foi variado entre 0.1 e 0.9 (Figura 4.7).



(a) Ruído × *Accuracy*

(b) Ruído × *Precision(macro)*

(c) Ruído × *Recall(macro)*

(d) Ruído × *F-score(macro)*

Figura 4.7: Métricas de *Accuracy* (a), *Precision (macro)* (b), *Recall (macro)* (c) e *F-score (macro)* (d) com a variação do ruído nos grafos.

A sensibilidade ao ruído atinge os maiores valores, para todas as métricas testadas, entre 0.4 e 0.5. A Tabela 4.5 mostra a perda da qualidade das métricas, para cada redução, em comparação ao grafo original, nesse ponto de maior perda. Acima desse valor, a diferença causada pelas reduções diminui pelo fato de que no próprio grafo original as métricas já são baixas. Esse fato é condizente com o previsto em Valejo et al. (2020b),

de que valores de ruído acima de 0.5 degradariam a qualidade do grafo ao ponto de gerar redes mal definidas. Já as perdas na faixa de reduções até 30% foram baixas, orbitando em torno de 5% para todas as métricas.

Redução	***Accuracy***	***Precision (macro)***	***Recall (macro)***	***F-score (macro)***
20%	$2.96 \pm 0.68\%$	$0.18 \pm 0.25\%$	$3.09 \pm 0.70\%$	$3.68 \pm 0.99\%$
35%	$6.12 \pm 1.11\%$	$0.30 \pm 0.37\%$	$6.36 \pm 1.14\%$	$7.36 \pm 1.53\%$
50%	$9.30 \pm 1.44\%$	$0.29 \pm 0.47\%$	$9.62 \pm 1.47\%$	$11.08 \pm 1.92\%$
60%	$12.35 \pm 1.68\%$	$0.21 \pm 0.55\%$	$12.76 \pm 1.71\%$	$14.71 \pm 2.23\%$
80%	$20.04 \pm 2.01\%$	$0.09 \pm 0.74\%$	$20.64 \pm 2.03\%$	$23.51 \pm 2.63\%$

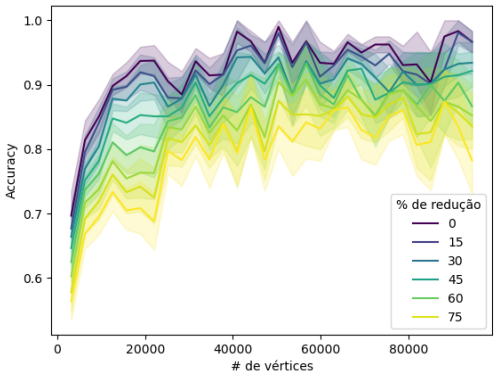
Tabela 4.5: Perda relativa das métricas em comparação ao grafo sem reduções em seus valores máximos

Após essa comparação, para verificar se os dois fatos destacados no experimento com até 15 mil vértices se manteria para grafos maiores, foi realizada a análise completa variando de 2 a 100 mil vértices (Figura 4.8 e Tabela 4.6), onda ainda foi incluída uma análise do comportamento dos experimentos de acordo com a % de rótulos passadas inicialmente, conforme o nível de redução aumenta 4.9.

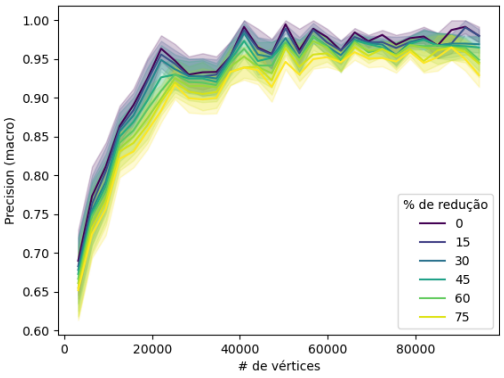
# de vértices	**Redução**	***Accuracy***	***Precision (macro)***	***Recall (macro)***	***F-score (macro)***
2000	20%	$2.55 + -0.40\%$	$0.76 + -0.14\%$	$3.10 + -0.46\%$	$3.85 + -0.62\%$
	35%	$4.07 + -0.56\%$	$1.27 + -0.18\%$	$4.85 + -0.63\%$	$5.73 + -0.84\%$
	50%	$6.22 + -0.77\%$	$1.81 + -0.24\%$	$7.31 + -0.86\%$	$8.31 + -1.05\%$
	60%	$8.69 + -1.00\%$	$2.48 + -0.31\%$	$10.09 + -1.10\%$	$10.89 + -1.23\%$
	80%	$15.86 + -1.61\%$	$4.02 + -0.40\%$	$18.31 + -1.82\%$	$19.32 + -1.92\%$
100000	20%	$3.25 + -1.13\%$	$1.08 + -0.38\%$	$3.25 + -1.13\%$	$3.92 + -1.36\%$
	35%	$4.79 + -1.16\%$	$1.55 + -0.41\%$	$4.90 + -1.17\%$	$5.77 + -1.45\%$
	50%	$11.25 + -1.80\%$	$3.24 + -0.51\%$	$11.06 + -1.78\%$	$14.24 + -2.27\%$
	60%	$12.67 + -1.93\%$	$3.59 + -0.57\%$	$12.56 + -1.93\%$	$15.91 + -2.41\%$
	80%	$19.81 + -2.58\%$	$5.24 + -0.61\%$	$19.65 + -2.56\%$	$23.22 + -2.97\%$

Tabela 4.6: Dados do experimento completo para o menor (2000) e maior (100000) número de vértices
).

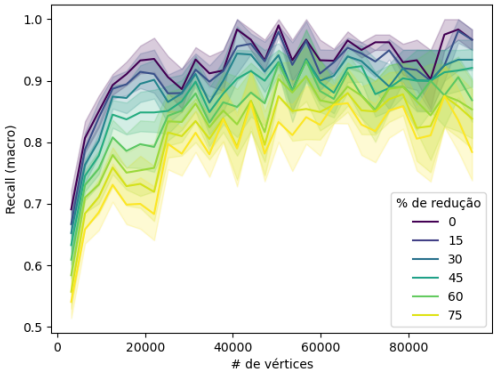
Os dados coletados nesse experimento confirmam os bons resultados para reduções em torno de 20%, e principalmente para a métrica de *Precision* com variação média de $0.55 \pm 0.17\%$ em relação ao grafo original. Nas métricas de *Accuracy*, *Recall* e *F-score*, as variações foram de $1.72 \pm 0.54\%, 1.78 \pm 0.55\%, 2.36 \pm 0.77\%$, respectivamente. Esse fato indica a existência de redundância nas informações topológicas do grafo, o que é um indício de que o *coarsening* é uma boa técnica para uso visando a economia de recursos de armazenamento e processamento de grafos.



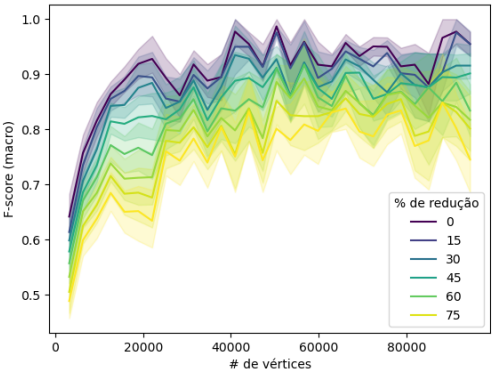
(a) # de vértices × *Accuracy*



(b) # de vértices × *Precision(macro)*

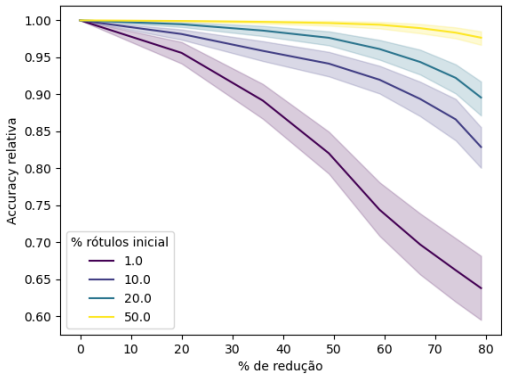


(c) # de vértices × *Recall(macro)*

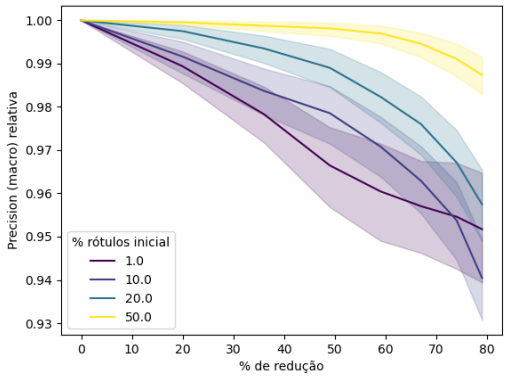


(d) # de vértices × *F-score(macro)*

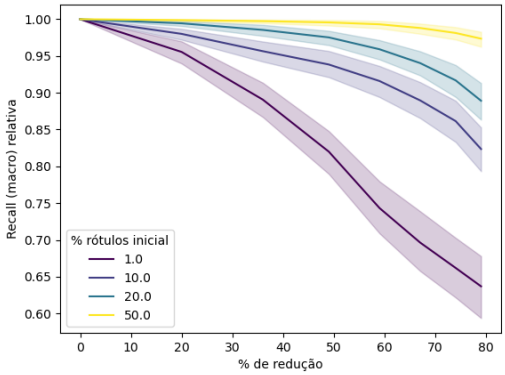
Figura 4.8: Métricas de *Accuracy* (b), *Precision (macro)* (b), *Recall (macro)* (c) e *F-score (macro)* (d) conforme o tamanho do grafo aumenta, com o número de vértices variando de 2 mil a 100 mil.



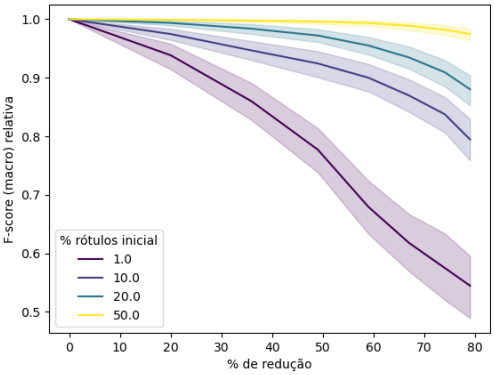
(a) # de vértices × *Accuracy*



(b) # de vértices × *Precision(macro)*



(c) # de vértices × *Recall(macro)*



(d) # de vértices × *F-score(macro)*

Figura 4.9: Métricas de *Accuracy* (b), *Precision (macro)* (b), *Recall (macro)* (c) e *F-score (macro)* (d) conforme o tamanho do grafo aumenta, no experimento com até 100 mil vértices.

4.2 Dados reais

Nesta seção é dado um exemplo de como as análises em redes sintéticas podem ser utilizadas no tratamento de dados reais. Para isso será usada a mesma base de dados utilizada no artigo original do GNetMine (Ji et al., 2010). A base DBLP[1] é um banco de informações bibliográficas abertas sobre as principais revistas e anais da ciência da computação. O estrato de dados utilizado está descrito na Tabela 4.7.

Tabela 4.7: Distribuição dos vértices, da base DBLP, usados no experimento.

Tipo	**Quantidade**
Autores (V_A)	14475
Artigos (V_P)	14376
Conferências (V_C)	20
Termos (V_T)	8920
Total	37791

O problema analisado sobre esta base foi o de classificação dos autores dentro de quatro áreas do conhecimento. Sendo os autores a partição alvo, as partições não-alvo usadas foram: diretamente, os artigos escritos por este autor, e indiretamente as conferências em que cada artigo do autor foi publicado, e os termos presentes nesses artigos.

O problema de se classificar esta base seria matematicamente modelado como:

- um grafo k-partido $G = (V, E, W)$, com as partições $V = \{V_P, V_A, V_C, V_T\}$ representando respectivamente os artigos, os autores, as conferências e os termos.
- o esquema de G seria definido como na Figura 4.10, tendo a partição V_A definida como alvo, a qual se deseja classificar;
- um conjunto de 4 classes $C = \{$*DataMining*, *Database*, *Information Retrieval*, *Machine Learning*$\}$, representativas de áreas de pesquisa; e
- um conjunto $V_A^L \subset V_A$ de autores já rotulados com uma das áreas de C.

O número de arestas neste grafo é 170795, um número considerado baixo, que corresponde a uma dispersão de cerca de 0.002. Como mostrado na seção anterior, em grafos com níveis de dispersão próximos de 0 é esperada uma perda baixa na qualidade das classificações (Figura 4.4), portanto para este problemas, do ponto de vista da qualidade das classificações seria recomendado deixar o *coarsening* agir até a convergência.

Por outro lado, podemos esperar uma baixa economia de armazenamento (Figura 4.3a). Para se determinar até que nível valeria a pena efetuar o *coarsening* nesse problema

[1] https://dblp.org/

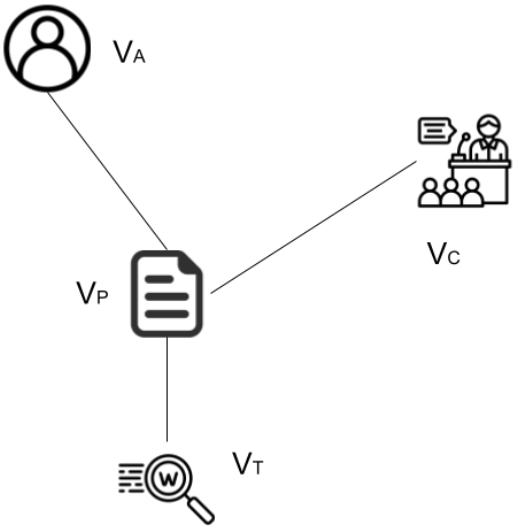
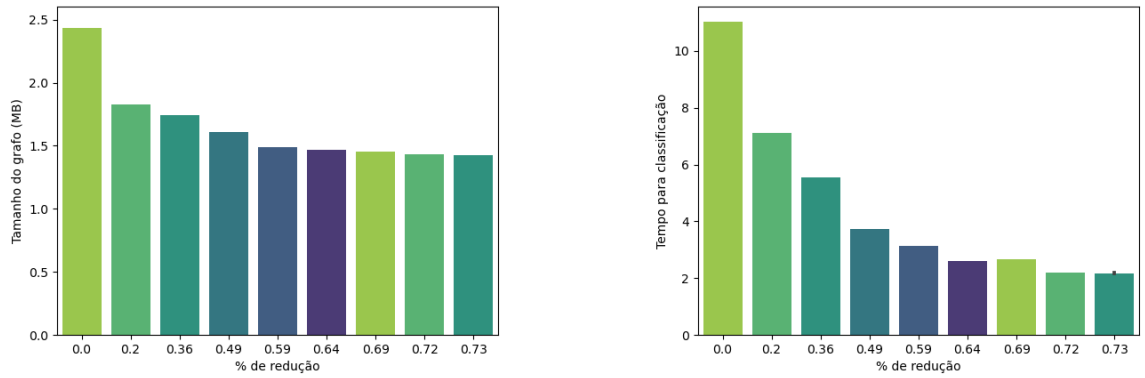


Figura 4.10: Esquema da rede DBLP

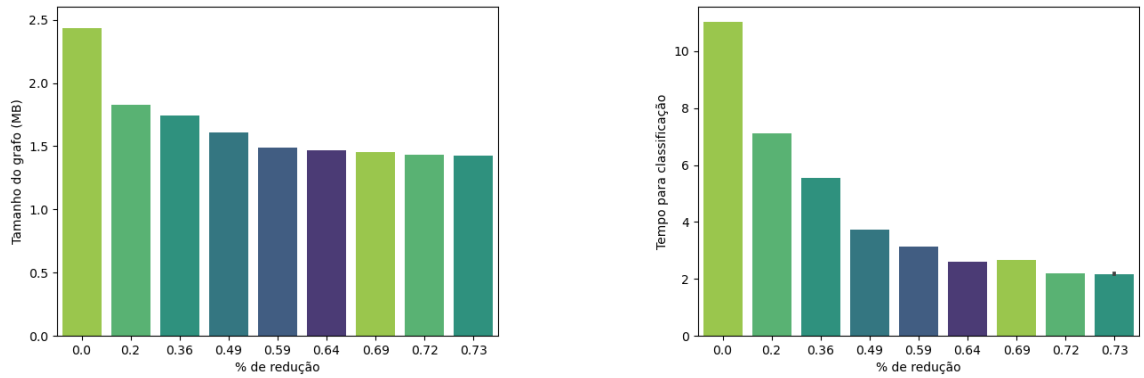
especifico, pode-se realizar uma análise em grafos sintéticos de mesma dispersão (Figura 4.11). Onde é possível perceber que reduções acima de 50% apresentam poucos ganhos neste tipo de grafos, o que poderia indicar uma redução alvo para se utilizar com o grafo real.



(a) (b)

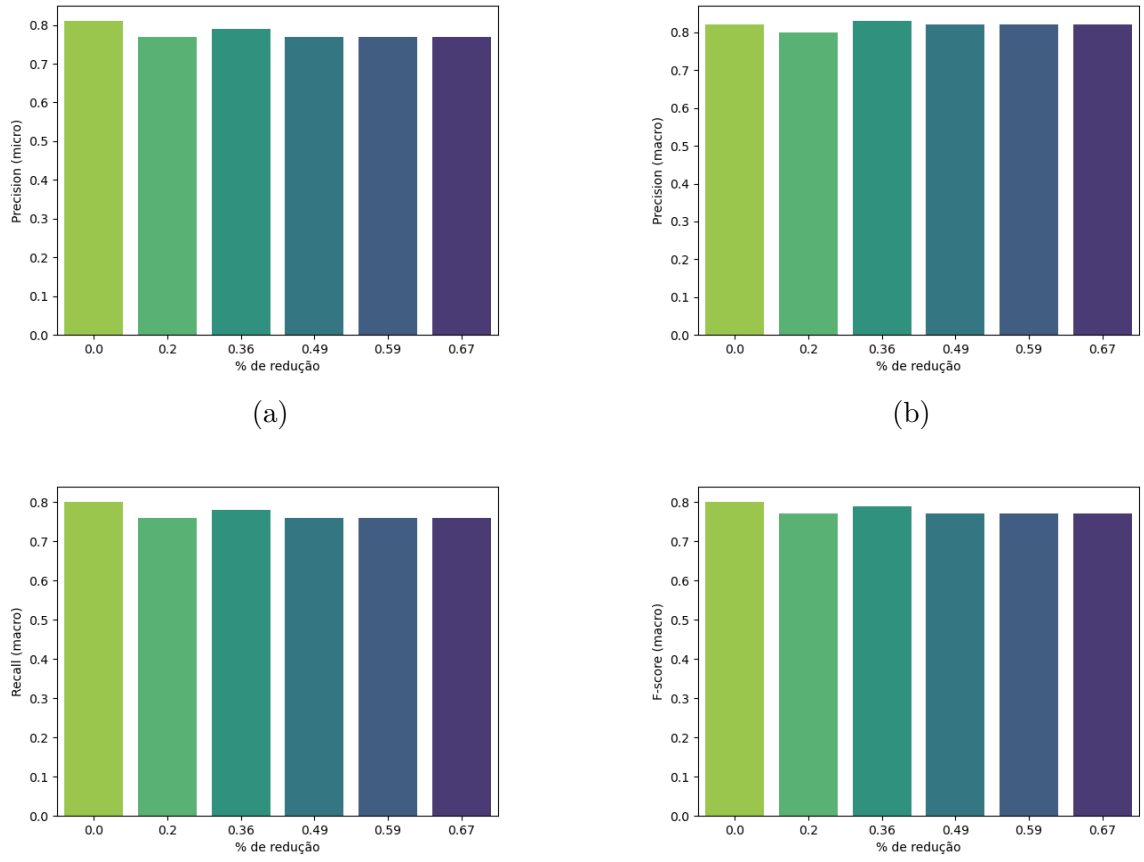
Figura 4.11: Análise de economias de armazenamento e tempo e grafo sintético de dispersão 0.02.

Foi realizado o *coarsening* no grafo real até a convergência. A Figura 4.12 mostra que, no caso real, uma redução de 36% já teria sido suficiente. A redução alvo estimada com o grafo sintético teria sido um bom resultado, apesar de não ótimo. Uma possível forma de se melhorar essa previsão, seria ter determinado o nível de ruído da rede real. Já a qualidade das métricas permaneceu constante como previsto.



(a) (b)

Figura 4.12: Análise de economias de armazenamento e tempo e grafo sintético de dispersão 0.02.



(a) (b)

(c) (d)

Figura 4.13: Métricas de *Accuracy* (a), *Precision (macro)* (b), *Recall (macro)* (c) e *F-score (macro)* (d) conforme o grafo é reduzido.

Capítulo 5

Conclusão

Neste trabalho foram realizadas análises dos efeitos do *coarsening* em grafos k-partidos, no que se diz respeito a métricas de qualidade em classificações. A hipótese a ser validada era se métodos de *coarsening* poderiam ser utilizados como forma de reduzir grafos, com o intuito de economia de espaço de armazenamento e escalabilidade de métodos aplicados a eles, se obtendo baixas perdas de qualidade em problemas de classificação. Na literatura encontram-se exemplos deste tipo de procedimento, porém, apenas em grafos homogêneos (Chen et al., 2017; Liang et al., 2020).

Para validar essa hipótese, duas questões de pesquisa foram definidas:

1. Quais os níveis de economia de armazenamento e tempo de classificação proporcionados pelo *coarsening*?
2. Quais os impactos da redução nos grafos em suas métricas de qualidade de classificação?

Com o objetivo de responder estas perguntas, foi realizado um estudo de técnicas de *coarsening* em grafos, e, em seguida, levantados trabalhos que realizam o tratamento da heterogeneidade, estendendo métodos aplicáveis a grafos homogêneos, e generalizando-os para grafos com mais partições. Como resultado disso, chegou-se a proposta de um algoritmo voltado para *coarsening* de grafos k-partidos, utilizando técnicas de *label propagation* (Capítulo 3). Esse algoritmo pode ser considerado uma extensão do procedimento para grafos bi-partidos apresentado em Valejo et al. (2021). Inclue-se como contribuição deste trabalho, a seção 3.3, que propõe um método de ordenação das partições e de escolha de caminhos no esquema para conduzir o *coarsening*, se diferenciando de outros trabalhos que a fazem de forma aleatória (Zhu et al., 2016).

O algoritmo proposto foi utilizado em análises empíricas sobre centenas de milhares de grafos sintéticos, variando características desses grafos para se obter, sob diferentes pontos

de vista, métricas relativas a economia de recursos providas pela redução dos grafos, e compará-las com possíveis perda na qualidade de classificação.

Quanto a economia de recursos, as reduções nos grafos apresentaram resultados expressivos, mesmo para os primeiros níveis de redução, onde reduções de apenas 20% no número de vértices realizadas pelo *coarsening* chegam a representar economias de armazenamento de mais de 1/3 e tornar as classificações cerca de duas vezes mais rápidas.

Quanto a perda na qualidade das classificações, destaca-se como fato positivo a baixa perda apresentada para reduções em torno de 20%, em contraste com as altas economias de armazenamento e tempo de classificação observadas já nesse primeiro nível de redução, com variações médias de $1.72 \pm 0.54\%$, $0.55 \pm 0.17\%$, $1.78 \pm 0.55\%$, $2.36 \pm 0.77\%$, em relação ao grafo original, para as métricas de *Accuracy*, *Precision*, *Recall* e *F-score*, respectivamente. Com especial destaque para a perda da métrica de *Precision*, que, além de níveis médios baixos, apresentou uma relativa estabilidade com a variação dos tamanhos dos grafos, variando menos de 4% na comparação entre grafos de 2 mil e de 100 mil vértices. Isso é um indício de que a técnica apresentada neste trabalho poderia ser melhor aproveitada em problemas onde falso negativos não são um problema mas deseja-se reduzir o número de falsos positivos.

Além disso os resultados levantados permitem a identificação da existência de um *threshold* a ser encontrado para cada tipo de problema. Dependendo das restrições de tempo/memória, e da aceitabilidade da introdução de perdas na qualidade da classificação de um problema em específico, existe um nível de redução apropriado para ser usado. Destaca-se assim, como principal contribuição do presente trabalho, essa extensa análise dos impactos do *coarsening* em métricas de classificação de grafos k-partidos. Os resultados apresentados na seção 4.1 apontam para o uso de *coarsening* em grafos heterogêneos como uma opção promissora para se lidar com problemas de escalabilidade e limitações de recursos, podendo ser usado como base para futuras pesquisas na área. Além disso, as análises contidas neste trabalho podem ser utilizados para resoluções de problemas reais na tarefa de se identificar esse mencionado *threshold*, como mostrado na seção 4.2, ao se buscar um ponto de otimização do *trade-off* economia de recursos $\times$ perda na qualidade.

5.1 Limitações do Trabalho e Ameaças a Validade

O presente trabalho se limita ao escopo de grafos k-partidos, não sendo diretamente aplicável a todo tipo de grafo heterogêneo sem as devidas alterações. Além disso o trabalho se propôs a analisar uma classe específica de problemas de classificação de vértices, que são aqueles onde existe uma partição alvo a ser classificada no grafo heterogêneo. Por utilizar características específicas desta delimitação de escopo, seus resultados a princípio não são imediatamente aplicáveis em outros tipos de classificação.

É também importante ressaltar que o uso de redes sintéticas ao invés de redes do mundo real pode enviesar o problema para o método de geração. Porém, o foco deste trabalho foi prover uma análise que atenda a grafos de uma maneira genérica, e a criação de redes permite um maior número de testes variando os parâmetros da rede. No caso em específico deste trabalho, o uso do algoritmo HNOC gera apenas grafos com um elevado nível de assortatividade, em que elementos similares tem uma maior probabilidade de se conectarem. Existem redes no mundo real que não atendem a esta característica, sendo necessária uma transformação, baseada em suas características para obter resultados similares aos encontrados neste texto. Para se investigar a influência deste fator, será proposto como trabalho futuro um estudo comparativo das mesmas análises efetuadas em grafos do mundo real de diferentes características. E, para se endereçar um problema real é interessante que uma análise similar a deste trabalho seja conduzida com redes reais do problema em específico, como mostrado na 4.2.

Outro potencial problema na geração dos grafos está nos parâmetros escolhidos para a geração. Para limitar esta limitação realizaram-se testes com uma ampla gama de variação desses parâmetros. Mas, como não seria possível se abranger todo tipo de rede possível, este trabalho disponibiliza toda a base de código usada nos experimentos em *https://github.com/pealthoff/CoarseKlass*. Dessa forma, outros parâmetros podem ser mais facilmente testados em trabalhos futuros.

Por fim, ressalta-se que o modo como os experimentos foram conduzidos, não incluíram uma análise da viabilidade da inclusão do *coarsening* em um classificador com o uso de métodos multi-níveis, assim como é feito por Chen et al. (2017) e Liang et al. (2020) no caso homogêneo. As análises se limitaram a utilização do *coarsening* como ferramenta para um processamento prévio visando o armazenamento. Porém, os resultados indicam esse outro potencial uso, que será indicado como trabalho futuro.

5.2 Trabalhos futuros

Com o fim do presente trabalho, as seguintes oportunidades para estender seus resultados foram visualizadas:

Criação de um classificador multinível: Os bons resultados nas métricas de classificação, sugerem que o método proposto poderia ser utilizado para a criação de um método multinível para escalar algoritmos de classificação, assim como Chen et al. (2017) e Liang et al. (2020) realizam para o caso de grafos homogêneos. Para tal fim, o procedimento de *coarsening* teria de ser otimizado em relação a tempo de execução. Na seção 3.3 é apontado um potencial ponto de uso de processamento paralelo para este fim.

Estudo comparativo de tipos de redes reais: Realizar análises similares as deste trabalho, comparando diferente tipos de redes reais, com o intuito de se identificar que tipos de redes, ou, que características presentes em redes reais, levariam a um maior ou menor ganho com o uso do *coarsening*, e quais sofreriam menores impactos na qualidade das classificações.

Estudar os impactos no ruído de redes reais pelo *coarsening*: A análise proposta na seção 4.2, poderia ser mais precisa caso fosse possível determinar o nível de ruído da rede real analisada. Além disso. os resultados com a rede real DBLP, apresentaram um interessante efeito onde o *coarsening* poderia não só melhorar a escalabilidade com pouca perda de qualidade, mas inclusive melhorar a qualidade dos resultados. Uma hipótese que se levantou é a de que a melhora poderia vir de uma eventual eliminação de ruído no grafo. Um possível trabalho futuro seria realizar uma análise, calculando o ruído de redes reais (utilizando o numero de arestas inter-classes) e verificando os impactos do *coarsening* nesta métrica.

Referências

C. C. Aggarwal. *Machine learning for text*. Springer, 2018. ISBN 9783319735313. doi: 10.1007/978-3-319-73531-3. 1

T. A. Akyildiz, A. A. Aljundi, and K. Kaya. GOSH: Embedding Big Graphs on Small Hardware. *ACM International Conference Proceeding Series*, 2020. doi: 10.1145/3404397.3404456. 32, 52

R. Angelova, G. Kasneci, and G. Weikum. *Graffiti: Graph-based classification in heterogeneous networks*, volume 15. 2012. ISBN 1128001101. doi: 10.1007/s11280-011-0126-4. 37

P. Bangcharoensap, T. Murata, H. Kobayashi, and N. Shimizu. Transductive classification on heterogeneous information networks with edge betweenness-based normalization. *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*, 2016. 56

A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of the eleventh annual conference on Computational learning theory - COLT' 98*, pages 92–100, New York, New York, USA, 1998. ACM Press. ISBN 1581130570. doi: 10.1145/279943.279962. URL `http://portal.acm.org/citation.cfm?doid=279943.279962`. 8

S. Brin and L. Page. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30(1-7):107–117, apr 1998. ISSN 01697552. doi: 10.1016/S0169-7552(98)00110-X. URL `https://linkinghub.elsevier.com/retrieve/pii/S016975529800110X`. 14, 27

T. N. Bui and C. Jones. A heuristic for reducing fill-in in sparse matrix factorization. 12 1993. URL `https://www.osti.gov/biblio/54439`. 28

C. Cai, D. Wang, and Y. Wang. Graph Coarsening with Neural Networks. feb 2021. URL `http://arxiv.org/abs/2102.01350`. 31

I. C. Campbell. Twitter will label covid-19 vaccine misinformation and enforce a strike system, Mar 2021. URL `https://www.theverge.com/2021/3/1/22307919/twitter-covid-19-vaccine-labels-five-strike-system`. [Online; accessado em 11-Mar-2021]. 2

H. Chen, B. Perozzi, Y. Hu, and S. Skiena. HARP: hierarchical representation learning for networks. *CoRR*, abs/1706.07845, 2017. URL `http://arxiv.org/abs/1706.07845`. 31, 39, 52, 70, 72, 73

D. Deng, F. Bai, Y. Tang, S. Zhou, C. Shahabi, and L. Zhu. Label propagation on k-partite graphs with heterophily, 2017. 21, 33, 42

R. Diestel. *Graph Theory*, volume 173 of *Graduate Texts in Mathematics*. Springer, Heidelberg; New York, fourth edition, 2010. ISBN 9783642142789 3642142788 9783642142796 3642142796. 8

C. Ding, T. Li, W. Peng, and H. Park. Orthogonal nonnegative matrix tri-factorizations for clustering. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, volume 2006, pages 126–135, 2006. ISBN 1595933395. doi: 10.1145/1150402.1150420. 33

P. Ding, C. Shen, Z. Lai, C. Liang, G. Li, and J. Luo. Incorporating multisource knowledge to predict drug synergy based on graph co-regularization. *Journal of Chemical Information and Modeling*, XXXX, 12 2019. doi: 10.1021/acs.jcim.9b00793. 56

T. Faleiros, R. Rossi, and A. Lopes. Optimizing the class information divergence for transductive classification of texts using propagation in bipartite graphs. *Pattern Recognition Letters*, 87, 04 2016. doi: 10.1016/j.patrec.2016.04.006. 14, 20, 21, 54, 56

T. d. P. Faleiros. *Propagação em grafos bipartidos para extração de tópicos em fluxo de documentos textuais*. PhD thesis, Instituto de Ciências Matemáticas e de Computação - USP, São Carlos, 2016. 21

M. Gupta, P. Kumar, and B. Bhasker. HeteClass: A Meta-path based framework for transductive classification of objects in heterogeneous information networks. *Expert Systems with Applications*, 68:106–122, feb 2017. ISSN 09574174. doi: 10.1016/j.eswa.2016.10.013. URL `https://linkinghub.elsevier.com/retrieve/pii/S0957417416305462`. 19, 21, 23, 46

B. Hendrickson. A multilevel algorithm for partitioning graphs. 02 1994. doi: 10.1145/224170.224228. 28

M. Ji, Y. Sun, M. Danilevsky, J. Han, and J. Gao. Graph regularized transductive classification on heterogeneous information networks. In J. L. Balcázar, F. Bonchi, A. Gionis, and M. Sebag, editors, *Machine Learning and Knowledge Discovery in Databases*, pages 570–586, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg. ISBN 978-3-642-15880-3. 19, 20, 56, 67

Josh James. Data never sleeps 9.0 - how much data is generated every minute, 2021. URL `https://www.domo.com/blog/what-data-never-sleeps-9-0-proves-about-the-pandemic/`. [Online; accessado em 19-Abr-2022]. 1

G. Karypis and V. Kumar. Analysis of multilevel graph partitioning. In S. Karin, editor, *Proceedings Supercomputing '95, San Diego, CA, USA, December 4-8, 1995*, page 29. ACM, 1995. doi: 10.1145/224170.224229. URL `https://doi.org/10.1145/224170.224229`. 28

S. Kullback and R. A. Leibler. On Information and Sufficiency. *The Annals of Mathematical Statistics*, 22(1):79 – 86, 1951. doi: 10.1214/aoms/1177729694. URL https://doi.org/10.1214/aoms/1177729694. 18

J. Leskovec and C. Faloutsos. Sampling from large graphs. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '06, page 631–636, New York, NY, USA, 2006. Association for Computing Machinery. ISBN 1595933395. doi: 10.1145/1150402.1150479. URL https://doi.org/10.1145/1150402.1150479. 26, 27

J. Leskovec and A. Krevl. SNAP Datasets: Stanford large network dataset collection. http://snap.stanford.edu/data, June 2014. 46

J. Liang, S. Gurukar, and S. Parthasarathy. Mile: A multi-level framework for scalable graph embedding, 2020. 2, 30, 31, 39, 52, 70, 72, 73

S.-D. Lin, M.-Y. Yeh, and C.-T. Li. Sampling and summarization for social networks. 2013. 25, 27

X. Liu and T. Murata. How does label propagation algorithm work in bipartite networks? WI-IAT '09, page 5–8, USA, 2009. IEEE Computer Society. ISBN 9780769538013. doi: 10.1109/WI-IAT.2009.217. URL https://doi.org/10.1109/WI-IAT.2009.217. 32, 45

Y. Liu, T. Safavi, A. Dighe, and D. Koutra. Graph summarization methods and applications: A survey. *ACM Computing Surveys*, 51(3), dec 2018. ISSN 15577341. doi: 10.1145/3186727. URL http://arxiv.org/abs/1612.04883. 2, 4, 24, 39

C. Luo, R. Guan, Z. Wang, and C. Lin. HetPathMine: A novel transductive classification algorithm on heterogeneous information networks. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 8416 LNCS, pages 210–221. Springer, 2014. ISBN 9783319060279. doi: 10.1007/978-3-319-06028-6_18. URL http://link.springer.com/10.1007/978-3-319-06028-6_18. 19, 21, 23, 46, 56

J. Luo, P. Ding, C. Liang, and X. Chen. Semi-supervised prediction of human mirna-disease association based on graph regularization framework in heterogeneous networks. *Neurocomputing*, 294:29–38, 2018. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2018.03.003. URL https://www.sciencedirect.com/science/article/pii/S0925231218302674. 56

T. Lyons, Jun 2018. URL https://about.fb.com/news/2018/06/increasing-our-efforts-to-fight-false-news/. 2

S. Macskassy and F. Provost. A simple relational classifier. 08 2003. 14

P. C. Mahalanobis. On the generalized distance in statistics. *Proceedings of the National Institute of Sciences (Calcutta)*, 2:49–55, 1936. 14

V. Mallawaarachchi. Label propagation demystified - a simple introduction to graph-based label propagation. https://towardsdatascience.com/label-propagation-demystified-cd5390f27472, march 2020. 14

R. Mansouri, M. Naderan-Tahan, and M. J. Rashti. A semi-supervised learning method for fake news detection in social media. In *2020 28th Iranian Conference on Electrical Engineering (ICEE)*, pages 1–5, 2020. doi: 10.1109/ICEE50131.2020.9261053. 2

H. Meyerhenke, P. Sanders, and C. Schulz. Partitioning complex networks via size-constrained clustering. *CoRR*, abs/1402.3281, 2014. URL http://arxiv.org/abs/1402.3281. 30, 32, 42

D. Minatel, A. Valejo, and A. A. Lopes. Trajectory network assessment based on analysis of stay points cluster. In *2018 7th Brazilian Conference on Intelligent Systems (BRACIS)*, pages 564–569, 2018. doi: 10.1109/BRACIS.2018.00103. 39

M. E. J. Newman. Scientific collaboration networks. ii. shortest paths, weighted networks, and centrality. *Phys. Rev. E*, 64:016132, Jun 2001. doi: 10.1103/PhysRevE.64.016132. URL https://link.aps.org/doi/10.1103/PhysRevE.64.016132. 39

T. Opsahl. Triadic closure in two-mode networks: Redefining the global and local clustering coefficients, 2010. URL https://arxiv.org/abs/1006.0887. 23, 46

B. Padrón, M. Nogales, and A. Traveset. Alternative approaches of transforming bimodal into unimodal mutualistic networks. the usefulness of preserving weighted information. *Basic and Applied Ecology*, 12(8):713–721, 2011. ISSN 1439-1791. doi: https://doi.org/10.1016/j.baae.2011.09.004. URL https://www.sciencedirect.com/science/article/pii/S1439179111001174. 23, 46

U. N. Raghavan, R. Albert, and S. Kumara. Near linear time algorithm to detect community structures in large-scale networks. *Phys. Rev. E*, 76:036106, Sep 2007. doi: 10.1103/PhysRevE.76.036106. URL https://link.aps.org/doi/10.1103/PhysRevE.76.036106. 30

S. Redmond and E. Rozaki. Using bipartite graphs projected onto two dimensions for text classification. 09 2017. doi: 10.15224/978-1-63248-131-3-19. 54

L. d. M. Romanetto. *Classificação transdutiva em redes heterogêneas de informação, baseada na divergência KL*. PhD thesis, Instituto de Ciências Matemáticas e de Computação - USP, São Carlos, 2020. 2, 8, 9, 19, 21, 22, 36, 39, 42, 56

R. G. Rossi, T. de Paulo Faleiros, A. de Andrade Lopes, and S. O. Rezende. Inductive model generation for text categorization using a bipartite heterogeneous network. In *2012 IEEE 12th International Conference on Data Mining*, pages 1086–1091, 2012. doi: 10.1109/ICDM.2012.130. 54

S. Sakellaridi, H.-r. Fang, and Y. Saad. Graph-based multilevel dimensionality reduction with applications to eigenfaces and latent semantic indexing. In *2008 Seventh International Conference on Machine Learning and Applications*, pages 194–200, 2008. doi: 10.1109/ICMLA.2008.140. 39

M. K. Sanches. *"Aprendizado de máquina semi-supervisionado: proposta de um algoritmo para rotular exemplos a partir de poucos exemplos rotulados* . PhD thesis, Universidade de São Paulo, São Carlos, aug 2003. URL http://www.teses.usp.br/teses/disponiveis/55/55134/tde-12102003-140536/. 8

C. E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27(3):379–423, 1948. doi: 10.1002/j.1538-7305.1948.tb01338.x. 18

M. M. Silva. Uma abordagem evolucionária para aprendizado semi-supervisionado em máquinas de vetores de suporte. Master's thesis, Escola de Engenharia - UFMG, Belo Horizonte, 2008. 1, 7, 8

A. Subramanya and J. Bilmes. Semi-supervised learning with measure propagation. *Journal of Machine Learning Research*, 12(102):3311–3370, 2011. URL http://jmlr.org/papers/v12/subramanya11a.html. 18, 21

M. Szummer and T. Jaakkola. Partially labeled classification with markov random walks. *Neural Inf Process Syst*, 14, 02 2002. 15

Szymański and Rathman. As internet user numbers swell due to pandemic, un forum discusses measures to improve safety of cyberspace, 2021. URL https://www.un.org/sustainabledevelopment/blog/2021/12/. [Online; accessado em 19-Abr-2022]. 1

A. Valejo, V. Ferreira, G. P. R. Filho, M. C. F. d. Oliveira, and A. d. A. Lopes. One-mode projection-based multilevel approach for community detection in bipartite networks. In *Annual International Symposium on Information Management and Big Data - SIMBig*. CEUR-WS, 2017. 4, 23, 32, 46

A. Valejo, V. Ferreira, M. C. de Oliveira, and A. de Andrade Lopes. Community detection in bipartite network: A modified coarsening approach. In *Communications in Computer and Information Science*, volume 795, pages 123–136. CEUR-WS, 2018a. ISBN 9783319905952. doi: 10.1007/978-3-319-90596-9_9. URL http://link.springer.com/10.1007/978-3-319-90596-9_9. 4

A. Valejo, M. C. Ferreira de Oliveira, G. P. Filho, and A. de Andrade Lopes. Multilevel approach for combinatorial optimization in bipartite network. *Knowledge-Based Systems*, 151:45–61, 2018b. ISSN 0950-7051. doi: https://doi.org/10.1016/j.knosys.2018.03.021. URL https://www.sciencedirect.com/science/article/pii/S0950705118301539. 4

A. Valejo, T. Faleiros, M. C. F. de Oliveira, and A. de Andrade Lopes. A coarsening method for bipartite networks via weight-constrained label propagation. *Knowledge-Based Systems*, 195:105678, 2020a. ISSN 0950-7051. doi: https://doi.org/10.1016/j.knosys.2020.105678. URL https://www.sciencedirect.com/science/article/pii/S0950705120301180. 4, 32, 52

A. Valejo, F. Góes, L. Romanetto, M. C. Ferreira de Oliveira, and A. de Andrade Lopes. A benchmarking tool for the generation of bipartite network models with overlapping communities. *Knowledge and Information Systems*, 62(4):1641–1669, Apr 2020b.

ISSN 0219-3116. doi: 10.1007/s10115-019-01411-9. URL https://doi.org/10.1007/s10115-019-01411-9. 21, 37, 42, 53, 63

A. Valejo, P. Althoff, T. Faleiros, M. Chuerubim, J. Yan, W. Liu, and L. Zhao. Coarsening algorithm via semi-synchronous label propagation for bipartite networks. In *Anais da X Brazilian Conference on Intelligent Systems*, Porto Alegre, RS, Brasil, 2021. SBC. URL https://sol.sbc.org.br/index.php/bracis/article/view/19047. 4, 5, 32, 42, 43, 45, 47, 52, 70

A. D. B. Valejo. Refinamento multinível em redes complexas baseado em similaridade de vizinhança. Master's thesis, Instituto de Ciências Matemáticas e de Computação - USP, São Carlos, 2014. 24

A. D. B. Valejo. *Métodos multi-nível em redes bi-partidas*. PhD thesis, Instituto de Ciências Matemáticas e de Computação - USP, São Carlos, 2019. 2, 4, 19, 26, 28, 31, 36

J. E. van Engelen and H. H. Hoos. A survey on semi-supervised learning. *Machine Learning*, 109(2):373–440, Feb 2020. ISSN 1573-0565. doi: 10.1007/s10994-019-05855-6. URL https://doi.org/10.1007/s10994-019-05855-6. 1

C. Walshaw. A Multilevel Algorithm for Force-Directed Graph Drawing. pages 171–182. 2001. doi: 10.1007/3-540-44541-2_17. URL http://link.springer.com/10.1007/3-540-44541-2_17. 39

C. Walshaw. Multilevel refinement for combinatorial optimisation problems. *Annals of Operations Research*, 131(1):325–372, Oct 2004. ISSN 1572-9338. doi: 10.1023/B:ANOR.0000039525.80601.15. URL https://doi.org/10.1023/B:ANOR.0000039525.80601.15. 2

Wikipedia contributors. Seven bridges of königsberg — Wikipedia, the free encyclopedia, 2021. URL https://en.wikipedia.org/wiki/Seven_Bridges_of_K%C3%B6nigsberg. [Online; accessado em 21-Mar-2021]. 9

Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu. A Comprehensive Survey on Graph Neural Networks. *IEEE Transactions on Neural Networks and Learning Systems*, 32(1):4–24, jan 2021. ISSN 2162-237X. doi: 10.1109/TNNLS.2020.2978386. URL https://ieeexplore.ieee.org/document/9046288/. 4, 39

S. Zhi, J. Han, and Q. Gu. Robust classification of information networks by consistent graph learning. In A. Appice, P. P. Rodrigues, V. Santos Costa, J. Gama, A. Jorge, and C. Soares, editors, *Machine Learning and Knowledge Discovery in Databases*, pages 752–767, Cham, 2015. Springer International Publishing. 56

D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf. Learning with local and global consistency. In *Proceedings of the 16th International Conference on Neural Information Processing Systems*, NIPS'03, page 321–328, Cambridge, MA, USA, 2003. MIT Press. 17, 18, 19, 20, 43

J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun. Graph neural networks: A review of methods and applications. *AI Open*, 1:57–81, dec 2020. ISSN 26666510. doi: 10.1016/j.aiopen.2021.01.001. URL `http://arxiv.org/abs/1812.08434`. 4, 39

L. Zhu, A. Galstyan, J. Cheng, and K. Lerman. Tripartite graph clustering for dynamic sentiment analysis on social media. In *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, SIGMOD '14, page 1531–1542, New York, NY, USA, 2014. Association for Computing Machinery. ISBN 9781450323765. doi: 10.1145/2588555.2593682. URL `https://doi.org/10.1145/2588555.2593682`. 33

L. Zhu, M. Ghasemi-Gol, P. Szekely, A. Galstyan, and C. A. Knoblock. Unsupervised entity resolution on multi-type graphs. In P. Groth, E. Simperl, A. Gray, M. Sabou, M. Krötzsch, F. Lecue, F. Flöck, and Y. Gil, editors, *The Semantic Web – ISWC 2016*, pages 649–667, Cham, 2016. Springer International Publishing. 5, 21, 42, 46, 52, 70

M. Zhu, F. Meng, Z. Yong, and G. Yuan. An approximate spectral clustering for community detection based on coarsening networks. *International Journal of Advancements in Computing Technology*, 4:235–243, 03 2012. doi: 10.4156/ijact.vol4.issue4.29. 30, 39

X. Zhu and Z. Ghahramani. Learning from labeled and unlabeled data with label propagation. Technical report, Carnegie Mellon University, 2002. 14, 16, 17

O. Zoidi, E. Fotiadou, N. Nikolaidis, and I. Pitas. Graph-Based Label Propagation in Digital Media. *ACM Computing Surveys*, 47(3):1–35, apr 2015. ISSN 0360-0300. doi: 10.1145/2700381. URL `https://dl.acm.org/doi/10.1145/2700381`. 13, 14