Replicação do Estudo de Escabilidade Paralela do Algoritmo *Infomap* para Detecção De Comunidades

Dhruv Babani[1], Bernardo Balzan[2], Eduardo Cardoso[3]

[1]Pontificia Universidade Catolica do Rio Grande do Sul(PUCRS)

d.babani001@edu.pucrs.br, b.balzan@edu.pucrs.br, Eduardo.SOO@edu.pucrs.br

Abstract. *This article presents the steps used in research on the detection of communities with graphs, demonstrating the views obtained from the presentation and description of the proposed problem.In addition to comparing the results obtained with the original results of the experiment.*

Resumo. *Este artigo apresenta os passos utilizados na pesquisa sobre a detecção de comunidades com grafos, demonstrando as visões obtidas a partir da apresentação e descrição do problema proposto, e também os procedimentos utilizados para a sua reprodução,além da comparação dos resultados obtidos com os resultados originais do experimento.*

1. Introdução

Atualmente, a temática do desempenho de algoritmos que manipulam estrutura de dados, tais como grafos, atua como um intermédio para geração de resultados com a finalidade de fazer análises descritivas para fins acadêmicos. Demonstrando uma área de estudo com muito potencial e englobando áreas mais abrangentes, como a ciência de dados e as metodologias ágeis, que tem crescido de forma exponencial nos ultimos anos.

A fim de agregar conhecimento por meio da pesquisa ao tema proposto, selecionamos juntamente ao professor orientador Carlos Cano e Professor PPGC Cesar D. Rose, a pesquisa realizada por Gabriel Giordani intitulada como ”Um estudo sobre a acurácia e a escalabilidade de algoritmos paralelos de detecção de comunidades em grafos”. O trabalho apresenta um comportamento de análise de desempenho de algoritmos na detecção de comunidades no modelo da memória compartilhada, com o intuito de ter uma visão analítica sobre os limites dos algoritmos propostos.

Para isso, são realizadas análises significantes, a partir da execução dos algoritmos em grafos de diferentes tamanhos (nodos e arestas) e níveis de complexidade, os quais podem elevar a quantidade de recursos utilizados e a demanda ao passar do tempo, por conta das complicações que o paralelismo pode trazer, inclusive às suas estratégias.

Sendo assim, foram disponibilizados alguns repositórios de códigos e grafos, os quais foram usados durante o processo de replicação, cujos os algoritmos tiveram o papel primário na manipulação dos dados para gerar os resultados, com a finalidade de ter uma análise melhor do processo em si.

2. Fundamentação Teórica

Com base nos estudos a respeito do potencial de paralelização em algoritmos de detecção de comunidades em grafos, diversos tópicos foram trabalhados com o intuito de aprofundar o aprendizado voltado para a temática envolvida.

2.1. Grafos

Um grafo pode ser compreendido como um modelo de estruturação e armazenamento de dados, definidos como G(V, E), onde:

G: Grafo;
V: Conjunto de vértices;
E: Conjunto de arestas.

As comunidades de Rede podem ser detectadas a partir de um grau de vértice, ou seja,o numero de arestas ligadas àquele grafo. Dessa forma, um grau do vértice pode ser classificado em duas terminologias: grau interno e grau externo. significando o numero de arestas que o conectam a outras e o numero de conexões com os elementos de fora da sua comunidade, respectivamente.

O problema desse processo de detecção baseia-se na divisão da rede em partições do conjunto de vértices conectadas entre si, onde há uma dificuldade para identificar a melhor demonstração da relação entre os indivíduos vista pelas arestas no grafo.

Todavia, apresenta-se o descobrimento desse agrupamentos de vértices que possuem uma ou mais características em comum como parte do problema.

Os diferentes tipos de comunidade em grafos configuram maneiras de caracterizar as relações de conexão dentro de uma rede, como por exemplo a vizinhança em comum entre vértices, que pode ser dada pelo número de arestas que os vértices de um mesmo grupo compartilham entre si.

A detecção das comunidades de rede é um processo de análise e identificação do grau dos vertices, no qual a sua interpretação possibilita diferenciar definições relativas à rigidez das conexões:

- Visao Tradicional
  - Estrutura fraca: média do grau interno dos vértices é maior do que a média do grau externo;
  - Estrutura forte: os vértices apresentam grau interno maior do que o grau externo.
- Visão Moderna
  - Estrutura fraca: média da probabilidade de cada vértice se conectar com outro vértice é maior do que a média da probabilidade de se conectarem com vértices externos;
  - Estrutura forte: os vértices possuem uma probabilidade maior de estarem conectados entre si do que com vertices externos.

Com isso, a compreensão dessas diferentes visões sugere uma relativização quando se trata de identificar comunidades, tendo uma abordagem voltada para o não determinismo dos resultados.

Dentre os métodos de detecção, existem as seguintes categorias:

- Inferência estatística: descrevem modelos generativos para os grafos, tendo o MDL (Minimum Description Length) como o mais utilizado;

- Otimização: procuram maximizar funções, como a funcão de modularidade das comunidades presentes em um grafo (ex.: Algoritmo de Louvain);
- Simulação de dinâmica: utilizam simulações na estrutura dos grafos, utilizando difusão, caminhamento (ex.: *Infomap*), sincronização, entre outros métodos.

2.2. Paralelismo para Melhoria da Escabilidade

A fim de compreender as abordagens do paralelismo, existem duas classificações de processamento: memória compartilhada e memória distribuída. Apenas o primeiro será abordado no trabalho devido a complexidade da segunda classificação.

No paralelismo em memória compartilhada é necessário que os processos coordenem seus acessos através de primitivas de sincronização, com o objetivo de haver uma utilização correta em execuções sequenciais ou concorrentes.

- Execução sequencial:
  - O resultado de uma tarefa é comunicado a outra tarefa;
  - Sincronização implícita, em que uma tarefa só é executada após o término da outra.
- Execução Concorrente:
  - Área de memória compartilhada acessada por diversos fluxos de execução;
  - Responsabilidade do usuário fazer uso de mecanismos de sincronização para garantir a correta comunicação.

*P*ara a medição de desempenho com a utilização de threads são utilizadas a função de *Speed Up* e *Eficiência*:

$$Speed\,Up = \frac{Tempo\,paralelo}{Tempo\,sequencial}$$

$$Eficiência = \frac{Speed\,Up}{Threads}$$

2.3. Experimento Original

Os experimentos presente no estudo são realizados em apenas uma máquina multicore, seguindo seguintes parâmetros para realização da análise:

- Tempo de execução do algoritmo;
- Escalabilidade: tempo de execução em relação ao crescimento do grafo;
- Consumo de recursos: consumo de memória e tempo gasto durante a troca de mensagens em algoritmos distribuídos;
- Qualidade das comunidades: utilização de grafos reais.

Em relação aos algoritmos e ambientes, foram apresentados seguintes comentários:

- Os repositórios do código possuem bibliotecas de referência, que por sua vez implementam métodos úteis para interpretação de grafos;
- Existe um arquivo *MAKEFILE* contido nos códigos para execução e compilação, em que os pacotes necessários para criação do ambiente são baixados.

Além disso, foi sugerido que fossem utilizados grafos com tempo de execução em minutos e com o processamento baseado em 6, 12, 24 e 48 threads. Visando uma análise mais comparativa entre os algoritmos abordados, simulando uma estratégia de memória compartilhada.

3. Processo de Replicação

Com o objetivo de descrever o processo de replicação dos experimentos realizados no estudo pertecente ao Gabriel Giordani, foi necessário fazer uma divisão dos tópicos para a preparação do ambiente e instalação dos algoritmos de manipulação dos grafos.

A Maquina usada para a replicação do processo possui as seguintes especificações:

- Sistema Operacional: MacOS Ventura 13.1
- Processador: M1 chip clock base de 1,8Ghz e boost de 3,1GHz.
  - Possui 8 nucleos,sendo 4 para eficiencia e o resto para desempenho
  - Núcleos da GPU: 8
  - Núcleos Neural Engine: 16
- RAM instalada: 8.0 GB.

Apesar de o estudo original ter sido realizado no sistema operacional Linux, foi possivel utilizar o MacOS para realizar a replicação do estudo, não havendo nenhum tipo de conflito que ocasionasse problemas técnicos.

3.1. Seleção dos Grafos

Inicialmente, foram escolhidos alguns grafos que descrevem uma rede gerada com diferentes quantidades de vértices e arestas, que por sua vez estão registradas em arquivos com diferentes tamanhos.

As informações relacionadas a estrutura desses grafos que configuram métricas úteis para a realização das análises:
V - Número de vértices, E - Número de arestas, Dmax - Grau máximo, Davg - Grau médio

Graph	Nodes	Edges
Live Journal	4, 847, 571	68, 993, 773
Pokec	1, 632, 803	30, 622, 564

3.2. Experimento para Detecção de Comunidades

Para realização do experimento, similarmente ao estudo original, foram utilizados dois grafos: *Live Journal* e *Pokec*. Sendo o ultimo um modelo de social network. Portanto, para localizar a detecção de comunidades nesses grafos foram utlizados os algoritmos *InfoMap* e *RelaxMap*.

Infomap é um algoritmo amplamente utilizado para o calculo da estrutura da comunidade de um gráfico. Existe uma gama diversificada de aplicações que o utilizam, sendo principalmente usado na área da biologia. Embora o algoritmo possua uma utilização tão grande, ele possui um número extremamente esparso de implementações paralelas e distribuídas atualmente disponíveis.

Acredita-se que a única implementação de memória compartilhada do *Infomap* existente atualmente é o *Relaxmap*, que adota uma abordagem de paralelização bastante simples. Para a etapa de agrupamento, os cálculos realizados para decidir se um nó deve mudar de comunidade são feitos em paralelo. Bae et al. mostra, que permitir que threads movam nós livremente pode resultar em movimentos cíclicos de nós entre comunidades vizinhas, impactando negativamente o MDL. Para evitar movimentos que impactem as

mesmas comunidades ao mesmo tempo, é utilizada uma seção crítica. Cada vez que um thread encontra um movimento válido, ele tentará acessar a seção crítica e, uma vez dentro, primeiro verificará se o movimento ainda é válido (outro thread poderia ter feito um movimento que invalida o thread atual) e então prosseguirá com o movimento.

Na etapa de ajuste o *Fine Tune* possui a mesma seção crítica utilizada na primeira etapa. Para o *Coarse Tune*, a divisão das comunidades em submódulos também é paralelizada. O limite padrão usado no *Relaxmap* para encerramento do cluster é 1e-3, que é significativamente maior que o limite do *Infomap* de 1e-10.

4. Resultados Obtidos

4.1. Tabelas

TABELA: Tempo médio em segundos do melhor desempenho para os algoritmos de *Infomap* e *RelaxMap*, para cada grafo mencionado anteriormente.

Estudo Original:

Graph	Infomap(s)	RelaxMap(s)
Live Journal	1270	90
Pokec	631.57	41.11

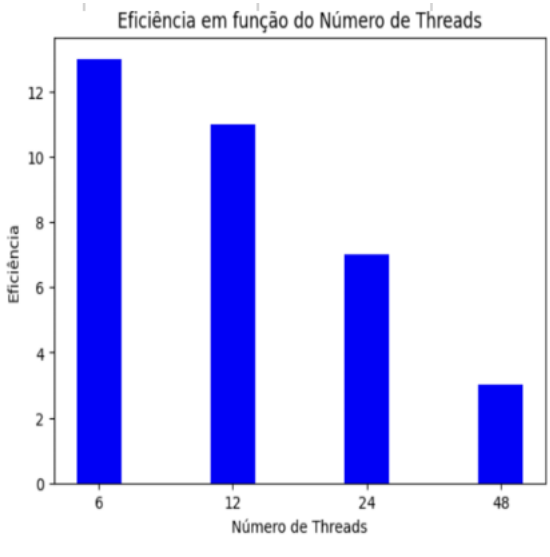
Replicação:

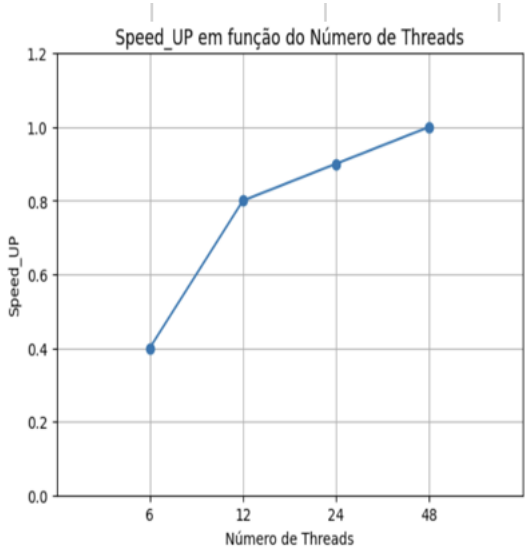
Graph	Infomap(s)	RelaxMap(s)
Live Journal	1150	75
Pokec	629.78	40.76

4.2. Graficos

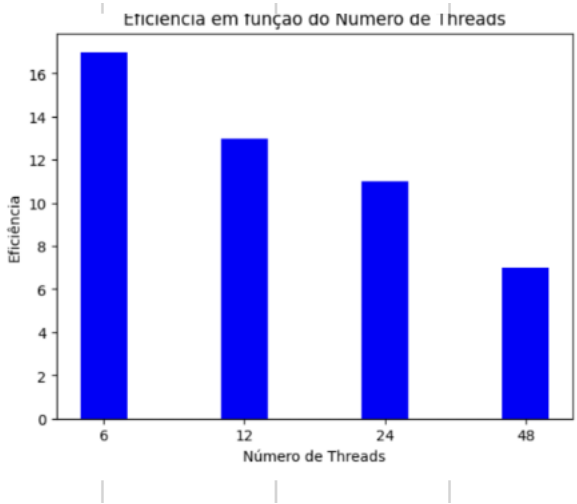
GRÁFICOS: Representação grafica da comparação de resultados de *Speed Up* e *eficiência* dos grafos no algoritmo *RelaxMap* entre este estudo e o original

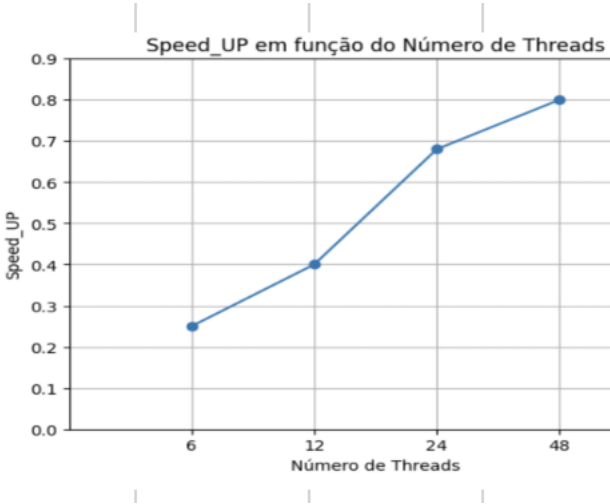
- Estudo Original - Live jornal



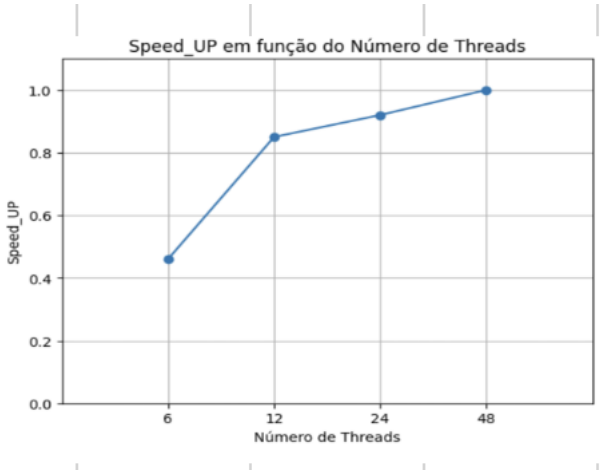


- Estudo Original - Pokec

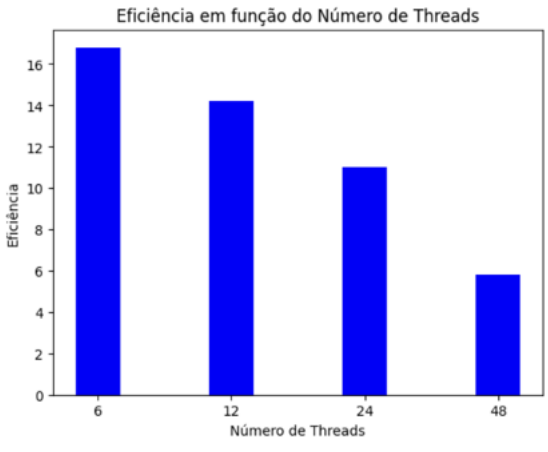


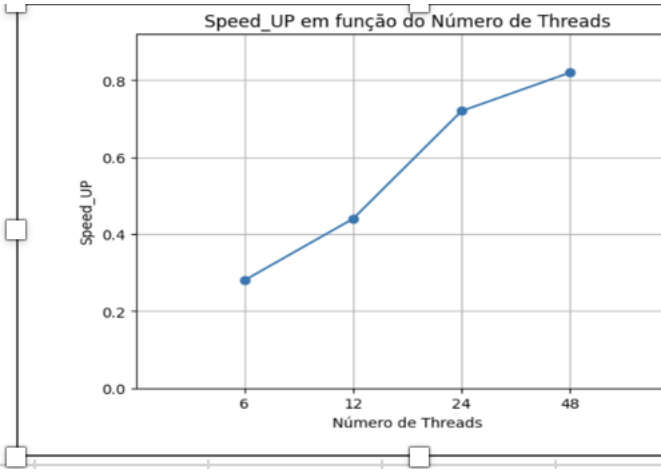


- Replicação - Live Jornal



- Replicação - Pokec





5. Potencial de Contribuição

Com o objetivo de encontrar uma forma viável para contribuição do estudo, os tópicos foram discutidos durante a realização dos encontros com o Dr Carlos Cano e Dr Gabriel Giordani.

6. Conclusão

Portanto, o processo de estudo realizado a partir das análises de desempenho dos algoritmos destacados construiu-se de maneira satisfatória, onde foi possível desenvolver o aprendizado a respeito da identificação e detecção de comunidades em grafos.

Os resultados obtidos com base nos experimentos de replicação foram capazes de demonstrar as métricas e manipulações contidas no estudo original, além das abordagens de paralelismo, mesmo com uma quantidade de núcleos.

Além disso, a partir das análises realizadas em relação ao desempenho dos algoritmos, possibilitou-se encontrar a melhor alternativa para os diferentes grafos, que apresentam um comportamento mais econômico para cenários que não necessitem um maior número de núcleos e opções para a máxima eficiência em processos de detecção.