Supercomputação

Pedro Balbo Portella

Relatório de Projeto: Análise de Redes Sociais — Clique Máxima em Grafos

Introdução:

Este relatório apresenta a implementação e análise de métodos para encontrar a clique máxima em grafos no contexto de redes sociais. O problema é computacionalmente desafiador devido à natureza combinatória de seu espaço de busca. O estudo foi realizado com o objetivo de identificar grupos coesos em redes sociais, uma tarefa relevante em diversas áreas, como análise de dados, sociologia e marketing.

Escopo:

- **Tamanho máximo de problema em tempo hábil (single thread):** 203 nós.
- **Tamanho máximo de problema em tempo hábil (multi thread):** mais de 1000 nós.
- **Objetivo:** Comparar as abordagens implementadas em termos de eficiência e escalabilidade.

Metodologia:

Cinco abordagens foram implementadas para resolver o problema, duas sendo single thread, uma multithread, outra multiprocesso e por fim a última usando GPU:

Implementações Single-Thread:

1. Implementação Exaustiva:

O código em C++ implementa a busca exaustiva pela clique máxima, explorando todas as combinações possíveis de vértices. Ele utiliza uma matriz de adjacência para representar o grafo e aplica iterações para identificar os subconjuntos coesos.

- **Entradas:** Grafos representados em formato de matriz de adjacência.
- **Saídas:** Tamanho e vértices do clique máximo.

Pseudo-código Simplificado:

1. Iterar sobre todos os subconjuntos de vértices do grafo.
2. Verificar se cada subconjunto é um clique válido.
3. Armazenar a maior clique encontrada.

Resultados Obtidos

Os testes foram realizados em redes de diferentes tamanhos (10, 25, 50 e 100 nós), medindo o tempo médio de execução.

- Para **10 nós**, o tempo médio de execução foi **0.000003 segundos**.
- Para **25 nós**, o tempo médio foi **0.000121 segundos**.
- Para **50 nós**, o tempo médio foi **0.009283 segundo**s.
- Para **100 nós**, o tempo médio foi de **1.565658 segundos**.

Obs: Para redes com nós superiores a 100 nós o tempo de execução desta implementação dispara, ficando na casa dos minutos.

2. Abordagem com Heurística

O código em C++ implementa uma busca otimizada pela clique máxima utilizando heurísticas para reduzir o espaço de busca. Em teoria essa abordagem prioriza a eficiência sem sacrificar a precisão ao explorar grafos mais densos.

Estratégias Utilizadas:

- **Ordenação por Grau de Adjacência:** Os vértices foram processados em ordem decrescente de conexões, permitindo que a análise se concentre em regiões mais densas do grafo.
- **Critérios de Interrupção:** A busca é interrompida assim que não é mais possível aumentar o tamanho do clique com os vértices restantes.

Pseudo-código Simplificado:

1. Ordenar os vértices em ordem decrescente de graus.
2. Iterar sobre os vértices ordenados, adicionando-os à clique se conectados a todos os vértices já na clique.
3. Parar a análise quando nenhum vértice restante puder ser adicionado.

Resultados Obtidos

Os testes foram realizados com redes de diferentes tamanhos (10, 25, 50 e 100 nós), medindo o tempo médio de execução.

- Para **10 nós**, o tempo médio de execução foi **0.000004** segundos .
- Para **25 nós**, o tempo médio foi **0.000206** segundos .
- Para **50 nós**, o tempo médio foi **0.012554** segundos .
- Para **100 nós**, o tempo médio foi **1.752365** segundos .

Análise

Embora a abordagem com heurística introduza uma otimização significativa ao priorizar vértices com maior grau de adjacência, o custo adicional de recalcular os graus durante a ordenação compromete sua eficiência. Esse processo de iteração sobre o vetor de vértices aumenta o tempo de execução, particularmente em redes maiores.

Além disso, ambas as implementações single-thread (exaustiva e com heurística) mostram limitações severas de escalabilidade. Quando o número de nós na rede excede 100, o crescimento "exponencial" do espaço de busca torna o tempo de execução inviável. Isso reflete a necessidade de abordagens paralelas para lidar com redes de maior escala.

Essa análise reforça que as estratégias de ordenação e poda são úteis, mas insuficientes para resolver eficientemente problemas de grande dimensão sem suporte a paralelização.

Implementações Paralelas:

3. Abordagem Paralela com OpenMP

A implementação paralela da busca pela clique máxima foi realizada utilizando OpenMP, uma biblioteca que simplifica o uso de threads para computação paralela em ambientes compartilhados. A estratégia aproveita o poder de múltiplos núcleos para dividir e processar o espaço de busca simultaneamente, aumentando significativamente a eficiência em grafos maiores.

Métodos Utilizados:

- **Divisão do Espaço de Busca:** O grafo é dividido em subproblemas, cada um processado em uma thread diferente.
- **Paralelização da Heurística:** A ordenação por grau de adjacência e a construção incremental da clique foram otimizadas para execução simultânea.

Pseudo-código Simplificado:

1. Dividir os vértices entre as threads disponíveis.
2. Cada thread processa um subconjunto de vértices, verificando conexões e construindo cliques localmente.
3. Combinar os resultados para determinar o maior clique global.

Resultados Obtidos

Os testes foram realizados com redes de diferentes tamanhos (10, 25, 50, 100, 250, 500 e 1000 nós). Os tempos médios de execução são apresentados abaixo:

- Para **10 nós**, o tempo médio foi **0.017590 segundos**.
- Para **25 nós**, o tempo médio foi **0.027925 segundos**.
- Para **50 nós**, o tempo médio foi **0.002643 segundos**.
- Para **100 nós**, o tempo médio foi **0.000211 segundos**.
- Para **250 nós**, o tempo médio foi **0.020894 segundos**.
- Para **500 nós**, o tempo médio foi **0.031078 segundos**.
- Para **1000 nós**, o tempo médio foi **0.033449 segundos**.

Análise

A abordagem com OpenMP demonstrou ganhos substanciais de desempenho, especialmente em redes maiores. As seguintes observações foram feitas:

1. **Eficiência em Redes Maiores:** Enquanto as abordagens single-thread (exaustiva e heurística) enfrentam severas limitações de escalabilidade além de 100 nós, a paralelização permitiu processar redes de até 1000 nós em tempo hábil.
2. **Sobrecarga Inicial:** Redes pequenas (10 e 25 nós) mostraram tempos ligeiramente maiores devido à sobrecarga associada à criação e sincronização de threads.
3. **Speedup Notável:** Redes de tamanho intermediário (50 a 500 nós) apresentaram ganhos de performance consistentes, justificando o uso da paralelização para esses casos.

Conclusão

A implementação com OpenMP oferece uma solução viável e escalável para o problema do clique máximo em grafos. Ela é especialmente vantajosa para redes de médio e grande porte, onde a abordagem sequencial seria inviável. Para redes menores, a sobrecarga associada à paralelização pode limitar os ganhos.

4. Abordagem Paralela com MPI

A implementação da busca pela clique máxima com a biblioteca MPI (Message Passing Interface) foi projetada para explorar a capacidade de múltiplos processadores, dividindo o problema em partes independentes. Contudo, limitações específicas foram observadas em relação à escalabilidade desta abordagem.

Métodos Utilizados:

- **Divisão por Particionamento:** O grafo é dividido entre os processos MPI, com cada processo responsável por encontrar cliques em sua partição local.
- **Combinação de Resultados:** Após o processamento local, os resultados são reunidos para identificar a clique máxima global.

Pseudo-código Simplificado:

1. Dividir o grafo entre os processos MPI.
2. Cada processo realiza a busca exaustiva em sua partição.
3. Combinar os resultados utilizando comunicação entre os processos para determinar o maior clique.

Resultados Obtidos

Os testes foram realizados com redes de 10, 25 e 50 nós. Os tempos médios de execução são apresentados abaixo:

- Para **10 nós**, o tempo médio foi **0.000017 segundos**.
- Para **25 nós**, o tempo médio foi **0.000044 segundos**.
- Para **50 nós**, o tempo médio foi **0.001296 segundos**

Análise

Apesar do uso do MPI, a abordagem apresentou limitações em escalabilidade:

1. **Eficiência em Redes Pequenas:** A divisão eficiente do grafo permitiu tempos reduzidos para redes menores, com desempenho superior às abordagens single-thread.
2. **Desempenho em Redes Maiores:** Não fui capaz de identificar a raiz do problema, mas a abordagem com MPI não conseguia processar redes muito grandes (acima de 50 nós) em um tempo viável. Acredito que um dos motivos poderiam ser:

    - **Sobrecarga de Comunicação:** A troca constante de informações entre processos pode ter introduzido latência significativa. (Acredito ser o mais provável)

    - **Divisão Desequilibrada:** Partições desiguais podem ter causado sobrecarga em alguns processos.

Comparação com Outras Abordagens

- **OpenMP vs MPI:** Embora o MPI tenha se destacado em redes pequenas, sua escalabilidade foi inferior ao OpenMP, que conseguiu processar redes maiores (até 1000 nós) de forma eficiente.

Conclusão

A abordagem com MPI, apesar de promissora, enfrentou desafios para redes maiores, destacando a necessidade de ajustes para melhorar sua escalabilidade. Fatores como balanceamento de carga e redução da sobrecarga de comunicação devem ser explorados em iterações melhores.

Implementação em GPU:

5. Abordagem Paralela com CUDA

A abordagem CUDA utiliza a capacidade de processamento paralelo massivo de GPUs para acelerar a busca pelo clique máximo. A estratégia aproveita a alta largura de banda de memória e o paralelismo em nível de thread para distribuir as operações entre centenas de núcleos.

Métodos Utilizados:

- Kernel CUDA: Implementado para verificar se um vértice pode ser adicionado à clique, garantindo conexões com todos os vértices já presentes.
- Paralelismo de Threads: Cada thread verifica a conectividade de um vértice em relação aos outros.
- Divisão e Conquista: O algoritmo constroi o clique incrementalmente, utilizando a GPU para acelerar cada passo.

Pseudo-código Simplificado:

1. Inicializar os dados do grafo em memória da GPU.
2. Usar threads para verificar conexões entre vértices.

3. Construir a clique incrementalmente, utilizando a GPU para acelerar os cálculos.
4. Reunir os resultados finais e identificar o maior clique.

Resultados Obtidos

Foi realizado somente um teste com o grafo de 50 nós, para validar o código. Este foi o resultado:

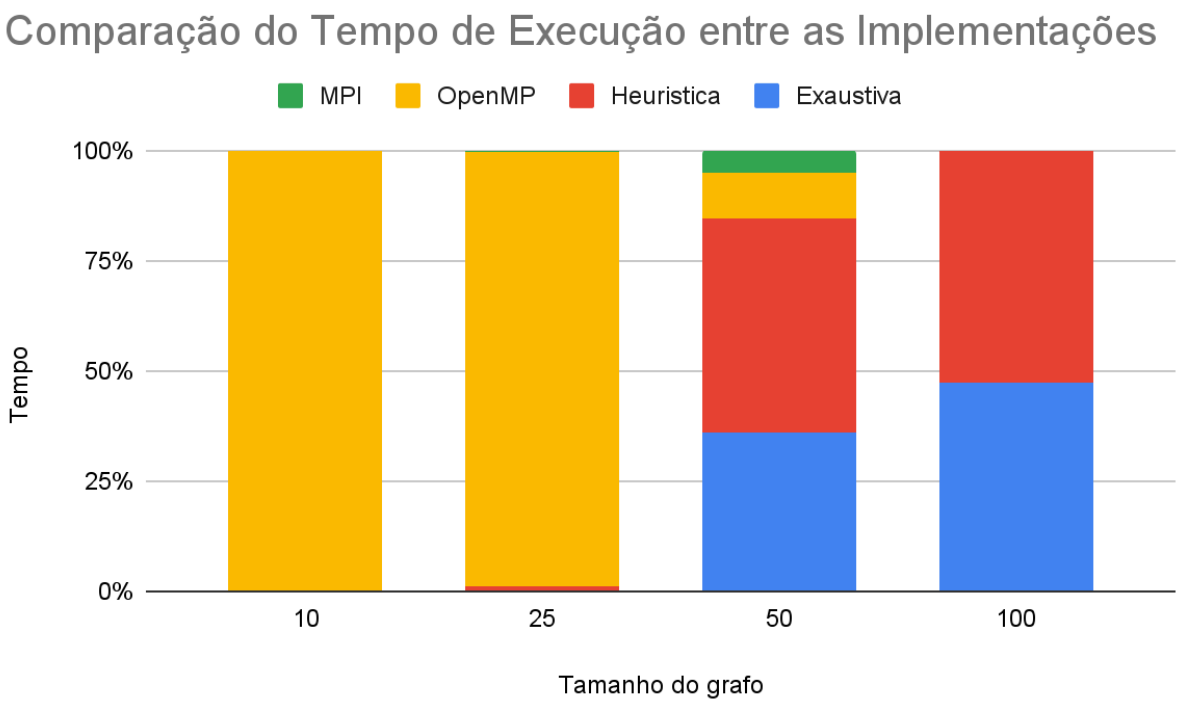
- **Tamanho da Clique Máxima:** 12.
- **Nós na Clique Máxima:** {1, 3, 5, 10, 14, 18, 25, 28, 32, 37, 40, 49}.
- **Tempo de Execução:** 19.855571 segundos.

Análise

A abordagem com CUDA demonstrou potencial, mas apresentou algumas limitações:

1. **Desempenho:** Embora eficiente para o processamento de conectividade, o tempo de execução foi alto em comparação com outras abordagens paralelas, como OpenMP, para grafos de tamanho moderado.
2. **Sobrecarga de Comunicação:** Transferências entre CPU e GPU podem ter introduzido latência significativa, afetando o desempenho.
3. **Limitações de Escalabilidade:** A implementação foi testada apenas para redes menores; sua eficiência em redes maiores deve ser avaliada.

Comparação Visual:



Conclusão:

O projeto abordou o problema de encontrar a clique máxima em grafos utilizando diferentes abordagens computacionais, explorando desde métodos single-thread até implementações paralelas e distribuídas. Cada abordagem apresentou vantagens e limitações específicas, evidenciando como o crescimento exponencial da complexidade impacta o desempenho em redes de maior escala.

Principais Conclusões:

1. **Implementações Single-Thread (Exaustiva e Heurística):**
    - Adequadas para redes menores, mas enfrentam sérias limitações de escalabilidade devido ao crescimento exponencial do espaço de busca.
    - Na abordagem heurística a necessidade de iterar para ordenar os vértices introduziu custos adicionais.

2. **Abordagens Paralelas com OpenMP e MPI:**
    - OpenMP: Demonstrou excelente escalabilidade para redes de tamanho médio e grande (até 1000 nós), destacando-se pela eficiência em ambientes com memória compartilhada.
    - MPI: Foi eficaz em redes menores, mas não conseguiu lidar com redes maiores devido à sobrecarga de comunicação e divisão de tarefas.
3. **Abordagem CUDA:**
    - Aproveitou o paralelismo massivo das GPUs para acelerar a busca pela clique máxima, mas apresentou limitações de desempenho devido à latência de comunicação CPU-GPU e maior tempo de execução em redes moderadas.

Considerações Finais:

As abordagens paralelas (OpenMP e CUDA) destacaram-se como soluções viáveis para redes de maior escala, com o OpenMP mostrando-se mais eficiente em geral. No entanto, nenhuma abordagem foi capaz de superar totalmente os desafios de escalabilidade impostos por redes muito grandes. Trabalhos futuros devem explorar otimizações adicionais, como balanceamento de carga em MPI, estratégias de particionamento mais inteligentes, e integração de heurísticas mais robustas com paralelismo.

O estudo ilustrou a importância de selecionar a abordagem computacional adequada ao tamanho e à densidade do grafo, enfatizando que a escalabilidade e a eficiência devem ser avaliadas caso a caso. Este relatório serve como base para futuras investigações e aprimoramentos nas implementações para resolver o problema da clique máxima em redes sociais e além.