

Conexidade Dinâmica

Projeto de Pesquisa para Iniciação Científica

Orientadora: Cristina Gomes Fernandes

Aluno: Daniel Angelo Esteves Lawand

Este projeto de pesquisa acompanha a requisição de bolsa de Iniciação Científica para o aluno Daniel Angelo Esteves Lawand.

1 Introdução

O objetivo deste projeto é o estudo e implementação de estruturas de dados e algoritmos para problemas de conexidade em grafos num contexto onde o grafo pode sofrer modificações [2].

O problema central será o de manter informações sobre as componentes conexas do grafo, ou mais exatamente, o problema de responder eficientemente consultas do tipo:

- Quantas componentes têm o grafo?
- Os vértices u e v estão numa mesma componente do grafo?

Num contexto estático, em que o grafo não sofre alterações, há um algoritmo linear que determina as componentes conexas do grafo e responde de maneira ótima consultas como estas.

No entanto, se o grafo em questão está sendo alterado, podendo ganhar ou perder arestas, o problema torna-se mais desafiador. Os tipos de modificações permitidas determinam a dificuldade do problema. Podemos considerar as três seguintes possibilidades:

- **conexidade incremental:** arestas podem ser acrescentadas ao grafo;
- **conexidade decremental:** arestas podem ser removidas do grafo;
- **conexidade totalmente dinâmica:** arestas podem ser acrescentadas ou removidas do grafo.

O problema de conexidade incremental é resolvido de maneira bastante eficiente por meio de uma estrutura de dados conhecida como *union-find* [8], que provê resposta às consultas em tempo aproximadamente constante.

O problema de conexidade decremental foi resolvido por Even e Shiloach [3]. O método usa uma tabela onde se mantém o identificador da componente de cada vértice do grafo. A grande questão é como atualizar essa tabela quando da remoção de uma aresta.

Se o grafo em questão é uma floresta, ou seja, não tem circuitos, o problema é mais simples. A ideia é que a remoção de uma aresta sempre quebrará uma componente em duas, e atualizaremos os identificadores do menor dos dois pedaços resultantes. Ao remover por exemplo a aresta uv , devemos determinar se o pedaço em que u ficou é menor ou maior que o pedaço em que v ficou. Isso pode ser feito por exemplo usando-se duas buscas, uma a partir de u e uma a partir de v , que são executadas paralelamente (ou de maneira intercalada) e, ao chegarmos ao final de uma delas, abortamos a segunda, pois determinamos já qual é o menor dos pedaços. O tempo gasto desta maneira será proporcional ao tamanho do menor dos pedaços, implicando que o tempo para a atualização, amortizado pelo número de consultas é $O(\lg n)$, onde n é o número de vértices do grafo.

Para grafos arbitrários, é necessário determinar se a aresta uv removida quebra ou não uma componente em duas. A ideia de novo é executar dois processos em paralelo (ou intercaladamente): um para determinar se u e v estão ainda na mesma componente após a remoção de uv , e o outro para decidir se uma componente foi quebrada em duas. Esse segundo processo é semelhante ao que foi usado no caso em que o grafo é uma floresta. Já o procedimento envolvido no primeiro processo é um pouco mais sofisticado, e utiliza uma estrutura de dados construída a partir de uma BFS do grafo original, que vai sendo atualizada a medida que o grafo vai sofrendo remoções de arestas.

Para conexidade totalmente dinâmica em florestas, podemos usar uma coleção das chamadas *link-cut trees* [6] ou das chamadas *Euler tour trees* [4] para representar a floresta. Estas estruturas implementam uma rotina chamada $\text{FINDROOT}(x)$, semelhante à rotina $\text{FINDSET}(x)$ do *union-find*, que devolve um representante da componente em que o vértice x se encontra. Com isso, podemos responder às consultas facilmente, pois dois vértices x e y estão na mesma componente da floresta se e somente se $\text{FINDROOT}(x) = \text{FINDROOT}(y)$. O tempo amortizado de atualização e consulta é $O(\lg n)$.

Para o caso geral da conexidade totalmente dinâmica, podemos representar um grafo arbitrário por uma floresta geradora maximal do grafo. Se chamarmos uma tal floresta de F , podemos usar por exemplo Euler tour trees para armazená-la. Com isso, inserções e consultas podem ser implementa-

das diretamente usando as operações correspondentes das Euler tour trees. Por outro lado, as remoções são mais desafiadoras. Em especial, se a aresta removida fizer parte da floresta F , isso pode requerer encontrar, de forma eficiente, uma outra aresta do grafo para ser adicionada a F . Esta operação é implementada por meio de uma estrutura de dados mais complexa, que planejamos estudar também, como parte deste projeto [1].

2 Justificativa

O estudo de estruturas de dados e algoritmos mais sofisticados ou com análises mais complexas proporciona uma oportunidade de aprofundamento de conhecimentos importantes adquiridos durante um bom curso de graduação em Ciência da Computação.

Problemas dinâmicos em grafos dependem de estruturas de dados não triviais, para serem resolvidos de maneira eficiente. Várias destas estruturas não são abordadas nos cursos obrigatórios de uma graduação em Ciência da Computação.

Redes como a internet são extremamente dinâmicas, e ampliaram dramaticamente o interesse no estudo de problemas dinâmicos em grafos. O estudo da conexidade de tais redes é uma das primeiras etapas no entendimento da dinâmica da rede, por isso eles têm um papel central nessa área.

O tema escolhido para este projeto serve como motivação para o estudo de tópicos centrais de áreas importantes da Ciência da Computação, como projeto e análise de algoritmos e estruturas de dados.

O Daniel é um aluno do terceiro ano do Bacharelado em Ciência da Computação (BCC) do IME-USP. Embora suas notas no primeiro ano não sejam tão boas, levando-o a uma média 7,0 nas disciplinas que cursou, o Daniel não obteve nenhuma reprovação, e melhorou significativamente seu aproveitamento no segundo ano, obtendo notas todas acima de 7,0. O seu interesse em fazer iniciação científica demonstra esse amadurecimento e certamente vai consolidar ainda mais essa melhora do seu aproveitamento no curso.

As disciplinas do BCC que já foram concluídas pelo Daniel são suficientes para levar adiante este projeto. Dentre as disciplinas que ele cursou, destacamos MAC0121 ESTRUTURAS DE DADOS E ALGORITMOS I e MAC0323 ESTRUTURAS DE DADOS E ALGORITMOS II, que dão uma base essencial para o desenvolvimento desse projeto.

3 Objetivos

Neste projeto, o plano é estudar e implementar estruturas de dados e algoritmos usados no contexto de conexidade dinâmica de grafos. Além do funcionamento destas estruturas e algoritmos, serão estudados também suas análises de correção e seu consumo assintótico de tempo. Ou seja, serão abordados também conceitos e técnicas de análise de algoritmos.

O Daniel fez um rápido estudo preliminar do problema da conexidade dinâmica em grafos, que deve ser aprofundado durante os próximos meses. Ele também já implementou uma primeira estrutura de dados, baseando-se no livro de Sedgewick e Wayne [5], que será usada como base da implementação das chamadas *splay trees* [7]. Estas por sua vez são usadas na implementação das *link-cut trees*. No presente momento, o Daniel está estudando as *splay trees*, e deve começar a implementá-las nas próximas semanas.

O objetivo dessa iniciação científica é o aprendizado de várias estruturas de dados que possuem aplicações não apenas em conexidade dinâmica, mas em diversas outras áreas da Ciência da Computação. Vale destacar que as estruturas de dados que serão abordadas são não triviais e, sendo este um projeto de iniciação científica, é possível que apenas parte do material mencionado seja completamente estudado e implementado.

Como subproduto da iniciação científica, o Daniel deve também produzir um texto com tudo o que for estudado.

4 Plano de trabalho e cronograma

Precisamos ajustar isso para o Daniel.

O Rafael começou recentemente a trabalhar nesse projeto. Numa primeira fase, ele estudou os capítulos iniciais do livro de Carvalho et al. [?] sobre algoritmos de aproximação. Recentemente, ele iniciou o estudo dos capítulos 2 e 7 do livro de Vazirani [?], que tratam, entre outras coisas, de alguns algoritmos de aproximação clássicos para o SCS.

O SCS é abordado em vários livros [?, ?, ?]. A nossa intenção é inicialmente estudar os resultados sobre o SCS por estes livros, pois geralmente a apresentação nos livros é mais acessível e detalhada que nos artigos. Claro que eventualmente serão consultados os artigos originais que descrevem os algoritmos em questão [?, ?, ?].

Durante os primeiros meses o Rafael estudará dois algoritmos clássicos

para o SCS [?, ?, ?], e continuará fazendo o levantamento bibliográfico sobre o assunto. A seguir, ele estudará a apresentação e análise original de um destes dois algoritmos, apresentada por Blum et al. [?]. Essa apresentação original é interessante por tornar esse algoritmo mais parecido com o algoritmo guloso para o SCS. Neste artigo de Blum et al. [?] aparece aliás a primeira análise do algoritmo guloso, que será o objeto de estudo na fase seguinte. Um artigo muito citado pelo de Blum et al. [?] é o de Turner [?], que então também deverá ser estudado nesta fase.

Ao final do estudo dos artigos de Blum et al. [?] e de Turner [?], o Rafael começará a escrever um texto com a descrição de tudo que ele tiver estudado até então. Ao mesmo tempo, ele começará a estudar os próximos tópicos do projeto.

Após esse estudo inicial, que terá coberto os algoritmos clássicos para o problema e a primeira análise do algoritmo guloso, iniciando uma segunda fase, o Rafael estudará um ou dois resultados de complexidade para o SCS [?, ?, ?]. Posteriormente, serão selecionados os tópicos específicos a serem estudados nessa segunda fase. Os tópicos possíveis são alguns dos algoritmos de aproximação mais recentes, apresentados apenas em artigos [?, ?, ?, ?, ?, ?], e algumas das análises aprimoradas do algoritmo guloso [?, ?, ?].

É bom destacar que mesmo esse levantamento bibliográfico inicial já mostra que há muito material sobre o assunto. Além disso, cada um dos tópicos a ser estudado é não-trivial, especialmente levando em conta o estágio acadêmico do Rafael. Assim sendo, podemos levar um tempo maior do que o previsto para cobrir devidamente o material pretendido.

Também a ordem em que os assuntos da segunda fase do projeto serão estudados poderá ser alterada no decorrer do trabalho, se nos parecer conveniente. De qualquer modo, o espírito do projeto será mantido.

Tudo que for estudado pelo Rafael será descrito com detalhes em um texto, que ele produzirá durante esta iniciação científica a medida que for estudando os diversos tópicos.

O cronograma estipulado para 12 meses é o seguinte.

Ativ/Mês	1	2	3	4	5	6	7	8	9	10	11	12
1	✓											
2	✓	✓										
3			✓									
4			✓	✓								
5				✓								
6				✓	✓	✓	✓					
7						✓						
8								✓	✓	✓	✓	
9					✓	✓	✓	✓	✓	✓	✓	✓
10												✓

Legenda:

1. Complementação do estudo de algoritmos de aproximação.
2. Estudo dos algoritmos clássicos para o SCS.
3. Estudo da primeira análise do algoritmo guloso.
4. Início da escrita do texto, referente a 1, 2 e 3.
5. Estudo de alguns resultados de complexidade para o SCS.
6. Estudo de alguns algoritmos mais recentes para o SCS.
7. Preparação do relatório intermediário.
8. Estudo das análises mais recentes do algoritmo guloso para o SCS.
9. Continuação da escrita do texto.
10. Preparação do relatório final.

5 Material e métodos

Como já comentamos, há muito material sobre o SCS na literatura. Primeiramente, o SCS é abordado em vários livros [?, ?, ?], que tratam dos resultados mais clássicos, dos artigos de Blum et al., de Li, e de Turner [?, ?, ?].

Gallant et al. [?] mostraram que o SCS é NP-difícil, mas existem alguns outros resultados de complexidade, sob o ponto de vista de aproximação (resultados de inaproximabilidade), nos artigos de Blum et al. e de Vassilevska [?, ?].

Vários algoritmos de aproximação para o SCS podem ser encontrados na literatura [?, ?, ?, ?, ?, ?, ?]. Também há uma série de artigos apresentando análises aprimoradas do algoritmo guloso [?, ?, ?].

O método que usaremos para conduzir essa iniciação científica é tradicional. O aluno estudará cuidadosamente os diversos resultados, e irá implementando parte do que for estudado, e teremos reuniões a cada duas semanas para discutir os assuntos estudados e as implementações. Ao mesmo tempo que estuda e implementa parte dos tópicos, o aluno escreverá um texto, o que possibilitará uma melhor avaliação de quão bem o material estudado está sendo absorvido.

6 Forma e análise dos resultados

Durante todo o período de estudo, além das implementações, o aluno estará preparando um texto, que, ao final do trabalho, conterá tudo que foi estudado na iniciação científica. Este é o principal objeto que pode ser usado na análise do trabalho que estará sendo desenvolvido. Fora isso, evidentemente esperamos que o aluno mantenha o bom desempenho (ou até melhor) no BCC.

Referências

- [1] E. Demaine and K. Lai (scribe). Dynamic graph problems. Lecture Notes in Advanced Data Structures, 2007.
- [2] C. Demetrescu, I. Finocchi, and G. F. Italiano. *Handbook of Data Structures and Applications*, chapter Dynamic Graphs. Chapman and Hall/CRC, 2004.
- [3] S. Even and Y. Shiloach. An on-line edge-deletion problem. *Journal of the ACM*, 28(1):1–4, 1981.
- [4] M. R. Henzinger and V. King. Randomized dynamic graph algorithms with polylogarithmic time per operation. In *Proceedings of the Twenty-Seventh Annual ACM Symposium on Theory of Computing (STOC)*, page 519, 1995.

- [5] R. Sedgewick and K. Wayne. *Algorithms*. Addison-Wesley, 4 edition, 2011.
- [6] D. D. Sleator and R. E. Tarjan. A data structure for dynamic trees. In *Proceedings of the Thirteenth Annual ACM Symposium on Theory of Computing (STOC)*, page 114, 1983.
- [7] D. D. Sleator and R. E. Tarjan. Self-adjusting binary search trees. *Journal of the ACM*, 32(3):652–686, 1985.
- [8] R. E. Tarjan. Efficiency of a good but not linear set union algorithm. *Journal of the ACM*, 22(2):215–225, 1975.