Relatório Bibliográfico sobre a Geração de Grafos Aleatórios Conexos com Densidade Alvo

Introdução

A geração de grafos sintéticos é uma pedra angular da ciência de redes, da teoria dos algoritmos e da análise de sistemas complexos. A capacidade de construir grafos que aderem a restrições específicas — como número de vértices (n), número de arestas (m) e conectividade — é fundamental para o teste de algoritmos, a simulação de processos de rede (por exemplo, difusão de informação ou propagação de doenças) e a compreensão das propriedades estruturais intrínsecas das redes. Um desafio central neste domínio é a geração de grafos que não apenas satisfaçam estas restrições, mas que também sejam amostrados uniformemente do conjunto de todos os grafos possíveis que cumprem esses critérios. A amostragem uniforme é crucial para evitar vieses na avaliação de algoritmos e na análise estatística.

Existem duas filosofias metodológicas principais para abordar este problema. A primeira é uma abordagem **construtiva**, que dissocia as restrições. Primeiro, garante-se a conectividade através da construção de uma subestrutura mínima, como uma árvore geradora, que utiliza n−1 arestas. Em seguida, arestas adicionais são inseridas aleatoriamente até que a densidade alvo (definida por m) seja alcançada. Esta abordagem garante a conectividade por construção, tornando-a inerentemente adequada para grafos de qualquer densidade, especialmente os esparsos.

A segunda é uma abordagem **probabilística**, mais comumente associada aos modelos de Erdős-Rényi. Neste caso, um grafo é gerado para satisfazer a restrição de densidade (ou seja, com m arestas), e subsequentemente, a sua conectividade é verificada. Se o grafo não for conexo, é descartado e o processo é repetido. A viabilidade deste método de amostragem por rejeição está intrinsecamente ligada à densidade do grafo alvo; é altamente eficiente para grafos densos, mas

computacionalmente intratável para grafos esparsos.

Este relatório apresenta uma análise bibliográfica aprofundada destas metodologias. A Seção 1 disseca a abordagem constutiva de aumento de árvore geradora, validando os seus fundamentos teóricos e avaliando o desempenho dos seus componentes constituintes com base na literatura seminal e de ponta. A Seção 2 explora metodologias alternativas de alto desempenho, fornecendo um contexto comparativo que elucida as suas respectivas forças e fraquezas. Finalmente, a Seção 3 sintetiza estas descobertas num conjunto de recomendações práticas e especializadas, culminando num quadro de decisão algorítmica para orientar a seleção da pipeline de geração mais apropriada com base nas características do grafo desejado.

Seção 1: O Método de Aumento da Árvore Geradora: Uma Abordagem Fundacional

Esta seção analisa em profundidade o método construtivo que consiste em gerar primeiro uma árvore geradora e depois adicionar arestas. Este método é validado teoricamente e os seus componentes são avaliados em termos de desempenho, com base na literatura científica.

1.1 O Princípio da Dissociação: Conectividade Primeiro, Densidade Depois

A estratégia de gerar um grafo conexo começando com uma árvore geradora e depois adicionando arestas aleatoriamente é um método robusto e teoricamente sólido. O seu princípio fundamental é a dissociação de restrições: a conectividade é garantida na primeira fase, e a densidade é ajustada na segunda.

1.1.1 Validade Teórica e Uniformidade

A geração de uma árvore geradora com n vértices estabelece uma estrutura base com exatamente n−1 arestas que, por definição, conecta todos os vértices.

Subsequentemente, a adição de m−(n−1) arestas restantes eleva o número total de arestas para m, cumprindo a restrição de densidade. Este processo garante a conectividade por construção, uma vantagem decisiva sobre métodos baseados em rejeição, particularmente quando o número de arestas m é próximo de n−1 (regime de grafos esparsos), onde um grafo gerado de forma puramente aleatória seria quase certamente desconexo.[1]

Para que o grafo final seja uma amostra uniforme do conjunto de todos os grafos conexos e rotulados com n vértices e m arestas, ambas as fases do processo devem ser uniformes. A primeira fase deve gerar uma **Árvore Geradora Uniforme (UST - Uniform Spanning Tree)**, na qual cada árvore geradora possível do grafo completo Kn tem a mesma probabilidade de ser selecionada. A segunda fase deve então amostrar as m−(n−1) arestas restantes uniformemente e sem reposição do conjunto de todas as arestas possíveis que não foram incluídas na árvore geradora. A combinação destes dois passos de amostragem uniforme resulta num processo globalmente uniforme. A ideia de modificar uma estrutura base, como uma árvore, através da adição de arestas é um conceito bem estabelecido na teoria dos grafos. Comentários em fóruns de investigação sugerem a modificação de árvores geradoras uniformes como uma estratégia viável [2], e a sua aplicação prática é até encontrada em domínios como o desenvolvimento de jogos para garantir a conectividade de mapas gerados processualmente.[3]

1.2 Fase 1: Geração de uma Árvore Geradora Uniforme (UST)

A literatura sobre a geração de USTs revela uma clara trajetória de evolução no desempenho e na sofisticação dos algoritmos. Esta evolução pode ser categorizada em três grandes eras: métodos de codificação combinatória, algoritmos baseados em passeios aleatórios e, mais recentemente, técnicas de tempo quase-linear que utilizam a teoria de grafos algébrica e espectral. A abordagem baseada em sequências de Prüfer, embora válida, representa o ponto de partida desta trajetória, com alternativas modernas a oferecerem ganhos de desempenho substanciais.

1.2.1 Codificação Combinatória: A Sequência de Prüfer

A sequência de Prüfer estabelece uma correspondência biunívoca entre as árvores rotuladas de n vértices e as sequências de comprimento n−2 formadas por rótulos de vértices de 1 a n.[4] Para gerar uma UST no grafo completo

Kn, pode-se simplesmente gerar uma sequência aleatória de n−2 inteiros (com reposição) no intervalo [1,n] e, em seguida, descodificar esta sequência para construir a árvore correspondente. Este método é elegante devido à sua simplicidade conceitual e prova direta da fórmula de Cayley para o número de árvores geradoras em Kn.[5]

A complexidade computacional é dominada pelo algoritmo de descodificação. Implementações ingénuas podem levar O(nlogn) devido à necessidade de encontrar repetidamente o vértice de menor grau, mas algoritmos mais sofisticados podem alcançar uma complexidade de tempo linear, O(n).[4]

A principal limitação da sequência de Prüfer é a sua especificidade: ela é projetada para gerar árvores no **grafo completo** Kn. Não se generaliza para o problema mais amplo de amostrar uma UST de um grafo de entrada arbitrário e esparso, G. No entanto, para o objetivo específico de gerar um grafo conexo a partir do zero, onde o universo de arestas potenciais é efetivamente o de Kn, este método é um ponto de partida válido e correto. A sua utilidade também se estende a contextos teóricos, como na análise de árvores com sequências de graus fixas.[6]

1.2.2 Algoritmos Baseados em Passeios Aleatórios: Aldous-Broder e Wilson

Para a tarefa mais geral de amostrar uma UST de um grafo arbitrário G, os algoritmos baseados em passeios aleatórios são os métodos clássicos e fundamentais.

O **algoritmo de Aldous-Broder**, desenvolvido independentemente por David Aldous e Andrei Broder, funciona realizando um passeio aleatório simples nos vértices do grafo.[7] Começando num vértice arbitrário, o algoritmo adiciona à árvore a aresta através da qual um vértice é visitado pela primeira vez. O processo termina quando todos os

n vértices foram visitados. A coleção de n−1 arestas forma uma UST.[9] A sua correção é elegantemente provada através do Teorema da Árvore da Cadeia de Markov.

O **algoritmo de Wilson** oferece uma melhoria significativa de desempenho.[10] Ele

utiliza

passeios aleatórios com eliminação de laços (loop-erased random walks). O algoritmo começa com uma árvore contendo um único vértice-raiz arbitrário. Em seguida, seleciona um vértice ainda não na árvore e inicia um passeio aleatório a partir dele. Quando o passeio atinge um vértice que já pertence à árvore, o caminho percorrido é analisado, quaisquer laços (ciclos) são removidos, e o caminho resultante sem laços é adicionado à árvore. Este processo é repetido até que todos os vértices sejam incluídos.[9]

A principal diferença de desempenho reside nas suas complexidades temporais. O tempo de execução do algoritmo de Aldous-Broder é limitado pelo *tempo de cobertura* do grafo, que é o número esperado de passos para visitar todos os vértices. No pior caso, este tempo pode ser da ordem de O(mn), que para grafos densos é O(n3).[9] O algoritmo de Wilson, por outro lado, tem um tempo de execução relacionado com o

tempo médio de alcance (mean hitting time), que é geralmente muito menor que o tempo de cobertura. O algoritmo de Wilson nunca é mais lento que o de Aldous-Broder por mais do que um fator constante e é frequentemente muito mais rápido na prática.[10] Estes dois algoritmos são considerados os pilares da geração de UST e são amplamente referenciados na literatura.[12]

1.2.3 Alternativas de Alto Desempenho: Algoritmos de Tempo Quase-Linear

A vanguarda da geração de USTs afastou-se dos passeios aleatórios puros para integrar técnicas sofisticadas da teoria de grafos espectral e algébrica, nomeadamente o uso de solucionadores rápidos para sistemas lineares Laplacianos. Esta abordagem explora a profunda conexão entre passeios aleatórios, árvores geradoras e o conceito de resistência efetiva em redes elétricas.[11]

Um resultado de destaque neste domínio é o algoritmo apresentado por Schild, que gera uma UST em tempo m1+o(1).[14] Isto representa uma melhoria substancial sobre os limites de pior caso de

O(mn) dos métodos baseados em passeios aleatórios, especialmente para grafos esparsos de grande escala. Outros trabalhos, como os de Kelner, Mądry e colaboradores, introduziram o paradigma de usar solucionadores Laplacianos para

"atalhar" passeios aleatórios, acelerando drasticamente o processo.[14] Embora a implementação destes algoritmos seja consideravelmente mais complexa, envolvendo rotinas de álgebra linear numérica, eles representam o estado da arte em termos de escalabilidade e desempenho. Para aplicações que exigem a geração de grafos extremamente grandes, estes métodos modernos são a escolha superior para a Fase 1.

A tabela seguinte resume a comparação entre os principais algoritmos de geração de UST.

Algoritmo	Princípio Central	Complexidade (Pior Caso)	Desempenho Típico	Limitações
Sequência de Prüfer	Codificação combinatória	O(nlogn) ou O(n)	Rápido para o seu domínio	Apenas para o grafo completo Kn; não geral
Aldous-Broder	Passeio aleatório simples	O(mn)	Lento, limitado pelo tempo de cobertura	Geral, mas frequentemente superado por Wilson
Algoritmo de Wilson	Passeios aleatórios com eliminação de laços	O(mn)	Geralmente muito mais rápido que Aldous-Broder	Geral, excelente escolha para implementação prática
Tempo Quase-Linear	Solucionadores Laplacianos, resistência efetiva	m1+o(1)	O mais rápido conhecido, especialmente para grafos grandes	Implementação complexa, requer bibliotecas numéricas

1.3 Fase 2: Aumento da Árvore para a Densidade Alvo

Uma vez que a árvore geradora inicial com n−1 arestas é criada, a segunda fase do algoritmo foca-se em adicionar as m−(n−1) arestas restantes para atingir a densidade alvo.

1.3.1 Amostragem Eficiente de Arestas

A tarefa consiste em selecionar uniformemente m−(n−1) arestas do conjunto de (2n)−(n−1) arestas não-árvore disponíveis. A abordagem de amostragem em lote com uma máscara booleana é altamente eficiente. Este método envolve a geração de uma representação de todas as arestas potenciais (por exemplo, os índices de uma matriz de adjacência triangular superior), a exclusão daquelas já na árvore geradora, a sua mistura aleatória e a seleção das primeiras m−(n−1) arestas. Com bibliotecas otimizadas como NumPy ou SciPy, esta operação é extremamente rápida.

Para grafos com um número muito grande de vértices n, a materialização da lista completa de ≈n2/2 arestas potenciais pode ser proibitiva em termos de memória. Nesses cenários, uma abordagem de amostragem iterativa é preferível. Pares de vértices (u,v) são gerados aleatoriamente e, se a aresta (u,v) ainda não existir no grafo, ela é adicionada. Este processo é repetido até que o número total de arestas atinja m. Embora evite a alocação de memória massiva, esta abordagem incorre no custo de verificar a existência da aresta a cada passo, o que pode ser mitigado com estruturas de dados eficientes como conjuntos de hash.

1.3.2 Enquadramento Teórico: A Perturbação Aleatória de Grafos

A validade da segunda fase do método encontra um forte suporte teórico num campo de investigação relacionado: a análise de grafos densos com perturbações aleatórias, por vezes referida como análise suavizada de grafos. Embora esta linha de investigação se concentre frequentemente em começar com um grafo denso arbitrário (por exemplo, um grafo com grau mínimo δ(G)≥dn para alguma constante d), as suas conclusões sobre como as propriedades do grafo evoluem com a adição de arestas aleatórias são amplamente aplicáveis.[16]

Uma árvore geradora é uma estrutura altamente ordenada e não aleatória. A adição de arestas escolhidas uniformemente ao acaso atua como um poderoso mecanismo de "aleatorização". Este processo "suaviza" a estrutura rígida da árvore, empurrando o grafo resultante em direção a uma instância mais típica do espaço de todos os

grafos conexos com n vértices e m arestas.

A literatura neste domínio quantifica este efeito. Por exemplo, Bohman, Frieze, Krivelevich e Martin mostram que adicionar apenas ω(1) arestas aleatórias (uma função que tende para o infinito, por mais lentamente que seja) a qualquer grafo denso é suficiente para garantir a conectividade e reduzir o diâmetro para no máximo 5, com alta probabilidade.[17] Outros resultados estabelecem os limiares para o aparecimento de cliques de tamanho fixo (

Kr) e outras subestruturas.[16] Este corpo de trabalho demonstra que a adição de arestas aleatórias é uma forma eficaz de transformar a estrutura de um grafo, validando teoricamente que a Fase 2 do método construtivo serve para randomizar eficazmente a árvore inicial.

Seção 2: Metodologias Alternativas e Comparativas

Enquanto o método de aumento da árvore geradora é robusto e geral, existem outras estratégias cuja adequação é ditada principalmente pela densidade do grafo alvo. A escolha de um algoritmo de geração ótimo é governada por uma dicotomia fundamental: a amostragem por rejeição é superior para grafos densos, enquanto os métodos construtivos são essenciais para grafos esparsos.

2.1 Amostragem por Rejeição via Modelos de Erdős-Rényi

A abordagem mais simples para gerar um grafo uniformemente aleatório com n vértices e m arestas é a amostragem por rejeição baseada no modelo G(n,m) de Erdős-Rényi.

O algoritmo é o seguinte:

1. Gerar um grafo G amostrando uniformemente m arestas distintas do conjunto de todas as (2n) arestas possíveis.
2. Testar se G é conexo.
3. Se G for conexo, retorná-lo. Caso contrário, descartar G e voltar ao passo 1.

A eficiência deste método depende inteiramente da probabilidade de um grafo G(n,m) ser conexo. A obra seminal de Paul Erdős e Alfréd Rényi estabeleceu que existe um limiar acentuado para a conectividade.[1] Um grafo torna-se conexo com alta probabilidade quando o número de arestas

m excede aproximadamente 2nlogn.

A análise de desempenho revela uma clara divisão:

- **Grafos Densos:** Se a densidade alvo for alta (por exemplo, m=Θ(n2)), então m está muito acima do limiar de conectividade. A probabilidade de gerar um grafo desconexo é infinitesimal, tornando este método extremamente rápido e simples de implementar. A taxa de rejeição é praticamente nula.[2]
- **Grafos Esparsos:** Se a densidade alvo for baixa (por exemplo, m=O(n)), o grafo está bem abaixo do limiar e será quase certamente desconexo. A amostragem por rejeição torna-se computacionalmente intratável, pois exigiria um número astronómico de tentativas para gerar uma única instância conexa.

Portanto, a amostragem por rejeição é a alternativa de alto desempenho preferida para **grafos densos**, mas é completamente inadequada para a geração de grafos esparsos.

2.2 Abordagens de Monte Carlo via Cadeias de Markov (MCMC)

Os métodos MCMC oferecem uma estrutura flexível para amostrar de distribuições de probabilidade complexas, incluindo o espaço de grafos conexos. Uma abordagem comum é uma cadeia de "troca de arestas" (edge-swap).

O algoritmo funciona da seguinte forma:

1. Começar com um grafo conexo arbitrário G0 com n vértices e m arestas.
2. Em cada passo i, propor uma transição para um novo grafo G′ modificando ligeiramente Gi. Por exemplo, escolher uma aresta aleatória (u,v)∈E(Gi) e uma não-aresta aleatória (x,y)∈/E(Gi), e definir G′ como o grafo com (u,v) removido e (x,y) adicionado.
3. Aceitar a transição (Gi+1=G′) se G′ for conexo. Caso contrário, rejeitar a transição (Gi+1=Gi).

Para garantir a amostragem de uma distribuição uniforme, critérios de aceitação mais

sofisticados, como o de Metropolis-Hastings, podem ser usados para corrigir quaisquer vieses na proposta de transição.[23]

Apesar da sua flexibilidade conceitual, os métodos MCMC enfrentam desafios significativos que limitam o seu uso como uma alternativa de alto desempenho para esta tarefa específica:

- **Tempo de Mistura:** O principal obstáculo é determinar o *tempo de mistura* da cadeia de Markov — o número de passos necessários para que a distribuição da cadeia se aproxime da distribuição uniforme alvo. A prova de que a cadeia converge para a distribuição uniforme e a estimativa do seu tempo de mistura são notoriamente difíceis. O espaço de estados é imenso, e os limites teóricos para o tempo de mistura são frequentemente polinómios de ordem muito elevada em n, com alguns estudos a mencionarem limites tão altos como O(n45), tornando-os lentos na prática.[23]
- **Custo por Passo:** Cada passo proposto requer uma verificação de conectividade, o que adiciona uma sobrecarga computacional significativa (O(n+m) por passo).

Em conclusão, embora o MCMC seja uma ferramenta poderosa e geral, não é competitivo em termos de desempenho para a geração de grafos conexos uniformes em comparação com a abordagem construtiva, devido às dificuldades em garantir tanto a uniformidade como a eficiência. A sua força reside na amostragem de distribuições de grafos mais complexas (por exemplo, modelos de grafos exponenciais), onde outros métodos não são aplicáveis.

Seção 3: Síntese e Recomendações Práticas

Esta seção final sintetiza as análises anteriores para fornecer uma avaliação especializada do método proposto e um quadro claro e baseado em evidências para a seleção do algoritmo ótimo.

3.1 Avaliação Especializada do Método Baseado em Prüfer

A abordagem que utiliza uma sequência de Prüfer para gerar uma árvore geradora, seguida pela adição em lote de arestas restantes, é **correta** para gerar um grafo conexo uniformemente aleatório com n vértices rotulados e m arestas. A sua natureza de duas fases é uma forma robusta e conceitualmente clara de lidar com as restrições de conectividade e densidade de forma separada.

- **Pontos Fortes:** A implementação é relativamente simples e o método é teoricamente sólido para o domínio pretendido (geração de grafos a partir do zero).
- **Pontos Fracos:** A principal fraqueza reside no desempenho e na generalidade da Fase 1. Embora a descodificação de Prüfer possa ser implementada em tempo O(n), ela é superada em escalabilidade pelos algoritmos de passeio aleatório e, especialmente, pelos métodos de tempo quase-linear. Além disso, está conceitualmente ligada à geração de árvores no grafo completo, enquanto outros algoritmos de UST são aplicáveis a qualquer grafo de entrada.

3.2 Recomendações de Alto Desempenho e Escalabilidade

Com base na análise da literatura, as seguintes pipelines são recomendadas para obter o máximo desempenho e escalabilidade.

- **Para Grafos Esparsos (e Uso Geral):** A abordagem construtiva de duas fases continua a ser a melhor, mas com uma substituição na Fase 1.
  1. **Fase 1 (Conectividade):** Gerar uma Árvore Geradora Uniforme (UST) utilizando um algoritmo de ponta.
     - **Máximo Desempenho:** Utilizar um **algoritmo de tempo quase-linear** como o descrito por Schild [14], que tem um tempo de execução de m1+o(1).
     - **Melhor Equilíbrio Desempenho/Implementação:** Utilizar o **Algoritmo de Wilson**.[9] É significativamente mais rápido que o de Aldous-Broder, mais fácil de implementar do que os solucionadores Laplacianos e oferece um excelente desempenho na prática.
  2. **Fase 2 (Densidade):** Adicionar as m−(n−1) arestas restantes através de amostragem uniforme sem reposição. O método de amostragem em lote com máscara booleana é uma excelente implementação para esta fase.
- **Para Grafos Densos:** O método mais simples e rápido é a **amostragem por rejeição**.
  1. Gerar um grafo G(n,m) amostrando m arestas uniformemente.

2. Verificar a conectividade. Como a probabilidade de um grafo denso ser desconexo é vanishingly small, a rejeição raramente ocorrerá.[1] Este método evita a sobrecarga de gerar explicitamente uma árvore geradora.

3.3 Quadro de Decisão Algorítmica

A escolha ótima do algoritmo é, em última análise, uma função da densidade do grafo. O ponto de decisão crítico é a relação entre o número de arestas alvo m e o limiar de conectividade de Erdős-Rényi, mcrit≈2nlogn. A tabela seguinte formaliza este quadro de decisão.

Regime de Densidade Alvo	Algoritmo Recomendado	Justificação e Compromissos Chave
Esparso / Geral ($m \lesssim 2n\log n$)	**Aumento de Árvore Geradora** (com UST de Wilson ou quase-linear)	**Justificação:** A amostragem por rejeição é computacionalmente intratável. Este método garante a conectividade por construção. **Compromissos:** Ligeiramente mais complexo de implementar do que a rejeição. O desempenho é ditado pela escolha do algoritmo de UST.
Denso ($m \gg 2n\log n$)	**Amostragem por Rejeição** (baseada no modelo G(n,m))	**Justificação:** O mais simples e rápido. A probabilidade de rejeição é desprezável, tornando-o altamente eficiente. **Compromissos:** Apenas aplicável a grafos densos; falha catastroficamente para grafos esparsos.

Este quadro fornece um guia prático e baseado em evidências para a seleção da metodologia mais eficiente e escalável para a geração de grafos aleatórios conexos com uma densidade alvo, garantindo tanto a correção teórica como o desempenho

prático.

Referências Bibliográficas

1. Bohman, T., Frieze, A., Krivelevich, M., & Martin, R. (2004). Adding random edges to dense graphs. *Random Structures & Algorithms*, 24(2), 105-117.

Snippet de código

```
@article{bohman2004adding,
  title={Adding random edges to dense graphs},
  author={Bohman, Tom and Frieze, Alan and Krivelevich, Michael and Martin, Ryan R.},
  journal={Random Structures \& Algorithms},
  volume={24},
  number={2},
  pages={105--117},
  year={2004},
  publisher={Wiley Online Library}
}
```

- **Relevância:** Este artigo seminal investiga o modelo de perturbação de grafos, quantificando o número de arestas aleatórias necessárias para induzir propriedades como conectividade e diâmetro pequeno, validando teoricamente a fase de aumento do método construtivo.

2. Broder, A. (1989). Generating random spanning trees. In *30th Annual Symposium on Foundations of Computer Science* (pp. 442-447). IEEE.

Snippet de código

```
@inproceedings{broder1989generating,
  title={Generating random spanning trees},
```

```
  author={Broder, Andrei},
  booktitle={30th Annual Symposium on Foundations of Computer Science},
  pages={442--447},
  year={1989},
  organization={IEEE}
}
```

- **Relevância:** Apresenta um dos primeiros e mais influentes algoritmos baseados em passeios aleatórios (o algoritmo de Aldous-Broder) para gerar árvores geradoras uniformes num grafo geral.

3. Cesa-Bianchi, N., Gentile, C., & Zappella, G. (2013). Random spanning trees and the prediction of weighted graphs. *Journal of Machine Learning Research*, 14(May), 1253-1284.

Snippet de código

```
@article{cesa2013random,
  title={Random spanning trees and the prediction of weighted graphs},
  author={Cesa-Bianchi, Nicol{\`o} and Gentile, Claudio and Zappella, Giovanni},
  journal={Journal of Machine Learning Research},
  volume={14},
  number={May},
  pages={1253--1284},
  year={2013}
}
```

- **Relevância:** Demonstra a aplicação de árvores geradoras aleatórias em problemas de aprendizagem de máquina em grafos, destacando a importância de amostrar uma árvore aleatoriamente para evitar viés adversarial.

4. Erdős, P., & Rényi, A. (1959). On random graphs I. *Publicationes Mathematicae Debrecen*, 6, 290-297.

Snippet de código

```
@article{erdos1959random,
  title={On random graphs I},
  author={Erd{\H{o}}s, P{\'a}l and R{\'e}nyi, Alfr{\'e}d},
  journal={Publicationes Mathematicae Debrecen},
  volume={6},
  pages={290--297},
  year={1959}
}
```

- **Relevância:** O artigo fundamental que introduziu o modelo de grafos aleatórios G(n,p) e lançou as bases para a teoria dos grafos aleatórios, incluindo o conceito de limiares de propriedade como a conectividade.

5. Pradhan, M., & Bhattacharya, B. B. (2022). A Prufer-Sequence Based Representation of Large Graphs for Structural Encoding of Logic Networks. *arXiv preprint arXiv:2209.01596*.

Snippet de código

```
@article{pradhan2022prufer,
  title={A Prufer-Sequence Based Representation of Large Graphs for Structural Encoding of Logic Networks},
  author={Pradhan, Manjari and Bhattacharya, Bhargab B.},
  journal={arXiv preprint arXiv:2209.01596},
  year={2022}
}
```

- **Relevância:** Descreve em detalhe a codificação e descodificação de sequências de Prüfer e discute a sua aplicação, confirmando a sua validade para gerar árvores rotuladas e a sua complexidade temporal linear.

6. Schild, A. (2017). An almost-linear time algorithm for uniform random spanning tree generation. *arXiv preprint arXiv:1711.06455*.

Snippet de código

```
@article{schild2017almost,
  title={An almost-linear time algorithm for uniform random spanning tree generation},
  author={Schild, Aaron},
  journal={arXiv preprint arXiv:1711.06455},
  year={2017}
}
```

- **Relevância:** Apresenta um algoritmo de ponta para a geração de USTs em tempo quase-linear, representando a alternativa de maior desempenho para a Fase 1 do método construtivo.

7. Wilson, D. B. (1996). Generating random spanning trees more quickly than the cover time. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing* (pp. 296-303).

Snippet de código

```
@inproceedings{wilson1996generating,
  title={Generating random spanning trees more quickly than the cover time},
  author={Wilson, David Bruce},
  booktitle={Proceedings of the twenty-eighth annual ACM symposium on Theory of computing},
  pages={296--303},
  year={1996}
}
```

- **Relevância:** Introduz o algoritmo de Wilson, um método baseado em passeios aleatórios com eliminação de laços que é geralmente muito mais eficiente do que o algoritmo de Aldous-Broder e é uma excelente escolha prática para a geração de USTs.

Referências citadas

1. Erdős–Rényi model - Wikipedia, acessado em julho 25, 2025, https://en.wikipedia.org/wiki/Erd%C5%91s%E2%80%93R%C3%A9nyi_model

2. Uniform sampling of random connected graph with given number of vertices/edges, acessado em julho 25, 2025, https://mathoverflow.net/questions/297317/uniform-sampling-of-random-connected-graph-with-given-number-of-vertices-edges
3. What is a good method to randomly generate edges between graph nodes?, acessado em julho 25, 2025, https://gamedev.stackexchange.com/questions/18351/what-is-a-good-method-to-randomly-generate-edges-between-graph-nodes
4. A Prufer-Sequence Based Representation of Large Graphs for Structural Encoding of Logic Networks - arXiv, acessado em julho 25, 2025, https://arxiv.org/pdf/2209.01596
5. Enumeration of tree-type diagrams assembled from oriented chains of edges - arXiv, acessado em julho 25, 2025, https://arxiv.org/pdf/2207.00766
6. The average number of spanning trees in sparse graphs with given degrees - arXiv, acessado em julho 25, 2025, https://arxiv.org/abs/1606.01586
7. Random Walks and Random Spanning Trees - Dartmouth College Mathematics, acessado em julho 25, 2025, https://math.dartmouth.edu/~pw/math100w13/kothari
8. Random choice spanning trees - arXiv, acessado em julho 25, 2025, https://arxiv.org/pdf/2402.05800
9. ALGORITHMS FOR SAMPLING SPANNING TREES ... - UPCommons, acessado em julho 25, 2025, https://upcommons.upc.edu/bitstreams/43163f5b-c95a-4e6e-8bb0-2a9682ae4173/download
10. Generating Random Spanning Trees More Quickly than the Cover ..., acessado em julho 25, 2025, https://www.cs.cmu.edu/~15859n/RelatedWork/RandomTrees-Wilson.pdf
11. Contents 1 Introduction and Wilson's algorithm - RIMS, Kyoto University, acessado em julho 25, 2025, https://www.kurims.kyoto-u.ac.jp/~kumagai/LN-barlow1.pdf
12. Loop Erased Walks and Uniform Spanning Trees Contents 1 Introduction - UBC Math, acessado em julho 25, 2025, https://www.math.ubc.ca/~barlow/preprints/106_ust_survey.pdf
13. A network clustering method based on intersection of random spanning trees - Annals of Computer Science and Information Systems, acessado em julho 25, 2025, https://annals-csis.org/Volume_39/drp/pdf/7644.pdf
14. An almost-linear time algorithm for uniform random spanning tree generation - arXiv, acessado em julho 25, 2025, https://arxiv.org/abs/1711.06455
15. [0908.1448] Faster generation of random spanning trees - arXiv, acessado em julho 25, 2025, https://arxiv.org/abs/0908.1448
16. Adding random edges to dense graphs - Mathematical Sciences, acessado em julho 25, 2025, https://www.math.cmu.edu/~af1p/Texfiles/denseplus2.pdf
17. Adding random edges to dense graphs, acessado em julho 25, 2025, https://arxiv.org/pdf/1605.07237
18. On smoothed analysis in dense graphs and formulas - SciSpace, acessado em julho 25, 2025,

https://scispace.com/pdf/on-smoothed-analysis-in-dense-graphs-and-formulas-4my41pfwu9.pdf
19. [1605.07237] Adding random edges to dense graphs - arXiv, acessado em julho 25, 2025, https://arxiv.org/abs/1605.07237
20. Rainbow Hamilton Cycles in Randomly Colored Randomly Perturbed Dense Graphs | SIAM Journal on Discrete Mathematics - SIAM.org, acessado em julho 25, 2025, https://epubs.siam.org/doi/10.1137/20M1332992
21. 6.207/14.15: Networks Lecture 4: Erdös-Renyi Graphs and Phase Transitions - MIT Economics, acessado em julho 25, 2025, https://economics.mit.edu/sites/default/files/inline-files/Lecture%204%20-%20Erdos-Renyi%20Graphs%20and%20Phase%20Transitions.pdf
22. Connectivity of the Erdős–Rényi random graph - MathOverflow, acessado em julho 25, 2025, https://mathoverflow.net/questions/60075/connectivity-of-the-erd%C5%91s-r%C3%A9nyi-random-graph
23. Generating connected random graphs | Journal of Complex ..., acessado em julho 25, 2025, https://academic.oup.com/comnet/advance-article/doi/10.1093/comnet/cnz011/5415736?guestAccessKey=847c8ac5-f90e-4b42-b3e1-8503de742f55