Trabalho Final Teoria dos Grafos
Algoritmo Parametrizado para o problema da Árvore de Steiner

Pedro Brum; Rafael Grandsire

5 de Julho de 2018

1 Introdução

Em uma cidade, apenas alguns habitantes assinaram o serviço de internet e, interessada em minimizar a quantidade de gastos em infraestrutura, a empresa provedora quer encontrar qual a menor quantidade de pontos que deve ligar com seus cabos de rede, de modo que todos seus assinantes possam se conectar às redes sociais favoritas. Esse problema é mais comumente conhecido como o problema da Árvore de Steiner. Em um grafo não direcionado e conexo, com um conjunto de vértices especiais (chamados de terminais), pretende-se encontrar um subconjunto de suas arestas que os conecte. Certamente esse conjunto existe, pois, o conjunto de todas as arestas satisfaz o requisito. O interesse real é, então, em um conjunto que minimize uma função de custo das arestas.

Formalmente, seja $G(V, E)$ um grafo conexo não direcionado, $K \subseteq V$ um conjunto de vértices terminais e $W\colon E \mapsto \mathbb{R}^+$, uma função de peso para as arestas. $E' \subseteq E$ forma uma árvore de Steiner se existe um caminho no grafo induzido por suas arestas entre quaisquer dois vértices de K. O custo do conjunto de arestas é dado por $\sum_{e \in E'} W(e)$ e a árvore de Steiner ótima, denotada por $ST(G, K)$, é aquela que, dentre todas as outras, possui custo mínimo.

Um parâmetro natural do problema é a quantidade de vértices terminais a serem conectados. Por essa perspectiva, a árvore de Steiner é uma generalização de problemas clássicos de teoria dos grafos. Se a quantidade de vértices terminais for igual a 2, o subconjunto de arestas com custo mínimo que liga os vértices é o caminho mínimo entre eles, mas, se a quantidade de terminais for igual à quantidade de vértices totais, o conjunto ótimo é a própria árvore geradora mínima.

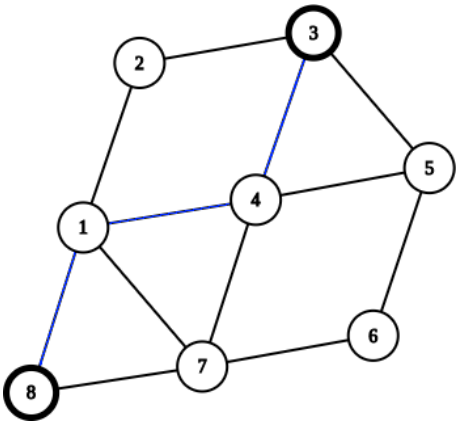

Figura 1: Caminho Mínimo

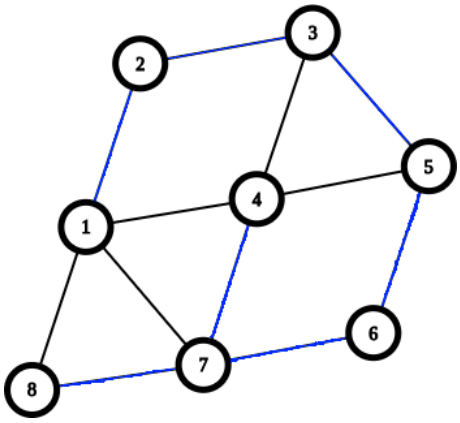

Figura 2: Árvore Geradora Mínima

Como visto, para conjuntos específicos de vértices terminais, em que $k = 2$ ou $k = |V|$, o problema pode ser resolvido em tempo polinomial. Na verdade, o resultado é mais amplo, já que, apesar de ser NP-Difícil [2], o problema é FPT [1] quando o parâmetro k é a quantidade de vértices terminais. Nesse trabalho, implementaremos um algoritmo de complexidade parametrizada para o problema da Árvore de Steiner. Aplicaremos o algoritmo em grafos gerados aleatoriamente para averiguarmos a escalabilidade do algoritmo e seu tempo de execução em casos práticos. Além disso, exploraremos as propriedades combinatoriais dos grafos que garantem que o algoritmo funcione para todos os casos e seja aplicável na prática para casos em que a quantidade de terminais é pequena.

2 Fundamentação Teórica

Para desenvolver o algoritmo parametrizado, precisaremos provar propriedades sobre os grafos e as soluções do problema. Algumas delas são triviais, como, por exemplo: o $ST(G, K)$ é sempre uma árvore. Entretanto outros serão provados nessa seção. O algoritmo em questão se baseia na modelagem do problema como uma programação dinâmica com uma relação de recorrência com parâmetros exponenciais apenas em k.

Lema 1 *Há uma transformação para qualquer grafo que faz com que o problema da árvore de Steiner seja resolvido equivalentemente com conjunto de terminais em que cada elemento possui grau igual a 1.*

Basta criarmos um novo grafo com um vértice artificial para cada vértice terminal do problema original e conectá-los a uma aresta que possua custo constante. A solução para o problema original será a mesma do novo grafo, a menos das arestas extras.

Prova. Seja $K = \{x_1, x_2, \ldots, x_k\}$ o conjunto de terminais originais, $K^* = \{y_1, y_2, \ldots, y_k\}$ os vértices artificiais e $E^* = \{\{x_1, y_1\}, \{x_2, y_2\}, \ldots, \{x_k, y_k\}\}$ as arestas adicionais. Temos que provar que para $G^* = (V \cup K^*, E \cup E^*)$, podemos construir a $ST(G, K)$ a partir de $ST(G^*, K^*)$

Primeiramente, E^* está sempre contido em $ST(G^*, K^*)$, pois são as únicas arestas que ligam aos vértices de K^*. Além disso, como cada aresta de E^* é da forma $\{x_i, y_i\}$, os vértices de K também pertencem ao grafo induzido por $ST(G^*, K^*)$. Portanto, $ST(G^*, K^*) \setminus E^*$ forma uma árvore de Steiner em G.

Finalmente, o custo de $ST(G^*, K^*) \setminus E^*$ é mínimo, pois $W(E^*) = \sum_{e \in E^*} W(e)$ é um valor constante e $W(ST(G^*, K^*) \setminus E^*) = \sum_{e \in ST(G^*, K^*)} W(e) - W(E^*)$ é mínimo para G', K'.

Sendo assim, $ST(G, K) = ST(G^*, K^*) \setminus E^*$, pois esse é um conjunto de arestas que conecta todos os vértices de K de custo mínimo. ■

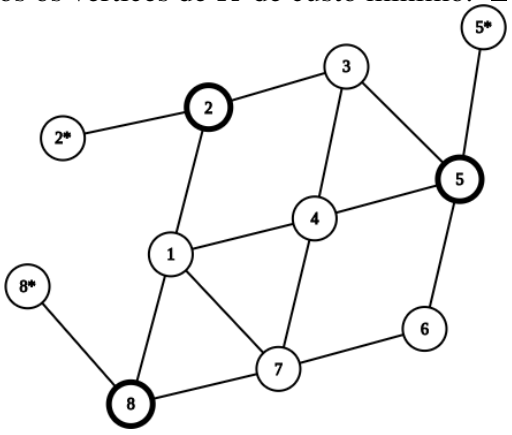

Figura 3: Grafo Modificado

No grafo apresentado, os terminais originais 2, 8 e 5 possuiam grau maior que 1. Após a inclusão dos vértices artificiais 2^*, 8^*, 5^* e arestas $\{2, 2^*\}, \{8, 8^*\}, \{5, 5^*\}$ a solução para o problema original pode ser obtida pela solução do novo grafo.

$$K = \{2, 5, 8\}$$

$$K^* = \{2^*, 5^*, 8^*\}$$

$$E^* = \{\{2, 2^*\}, \{8, 8^*\}, \{5, 5^*\}\}$$

$$ST(G, K) =$$

$$ST(G(V \cup K^*, E \cup E^*), K^*) \setminus E^*$$

Lema 2 *Se todo elemento de K possui grau 1, existe uma relação de recorrência T que calcula o custo de $ST(G, K)$ com no máximo $(|V| - |K|)2^k$ estados possíveis.*

$$T\colon \mathcal{P}(K) \times (V \setminus K) \mapsto \mathbb{R}$$

$$T(D, v) = \min_{\substack{u \in V \setminus K \\ \emptyset \neq D' \subset D}} T(D', u) + T(D \setminus D', u) + \text{dist}(u, v), \quad \text{para } |D| > 1$$

$$T(\{t\}, v) = \text{dist}(t, v), \quad \text{para } |D| = 1$$

Observação: $\mathcal{P}(K) = \{x\colon x \subseteq K\}$ é o conjunto potência de K.

Intuitivamente, a função $T(D, v)$ recebe um subconjunto D dos terminais e um vértice v não terminal e calcula o custo de formarmos duas árvore de Steiner ótimas, enraizadas em v, para dois subconjuntos próprios de D. Esses subconjuntos são denotados por D' e $D \setminus D'$ possuem cardinalidade menor que D e, por isso, o problema pode ser resolvido como uma programação dinâmica. Como não sabemos, entretanto, qual o conjunto D' ótimo, todos devem ser testados. Além disso, deve-se determinar se outros vértices não terminais também devam participar da solução e, para o conecta-lo à raiz, o caminho mínimo é a melhor opção gulosamente.

Para o caso geral, como não sabemos quais vértices não terminais devem pertencer à arvore final, o custo de uma árvore de Steiner de um grafo para um certo conjunto de terminais é dado por

$$W(ST(G, K)) = \min_{v \in V \setminus K} T(K, v)$$

Prova. A demonstração da otimalidade da relação de recorrência será dividida em duas partes. Queremos provar que, se uma árvore de Steiner para os terminais D contém o vértice v, seu custo é limitado por $T(D, v)$. Denotando por $ST_v(G, D)$ a árvore de Steiner ótima que possui o vértice v, provaremos que $T(D, v) \leq W(ST_v(G, D)) \leq T(D, v)$ e depois isso basta para que $W(ST_v(G, D)) = T(D, v)$.

- **Limite Superior:** É óbvio, inicialmente, que para qualquer vértice não terminal u, cujo caminho mínimo até v é dado por $H(v \sim u)$, e subconjunto de terminais D', $ST_u(G, D') \cup H(v \sim u) \cup ST_u(G, D \setminus D')$ forma uma árvore de Steiner. Isso significa diretamente que, para toda árvore de Steiner que contenha o vértice v, seu custo é limitado da forma

  $$\forall D' \subseteq D, \forall u \in V \setminus K, \quad W(ST_v(G, D^{)}) \leq W(ST_u(G, D')) + \text{dist}(v, u) + W(ST_u(G, D \setminus D'))$$

  Como o resultado anterior é independente do subconjunto D' e vértice u, diz-se, sem perda de generalidade, que

  $$\forall D' \subseteq D, \forall u \in V \setminus K, \quad W(ST_v(G, D)) \leq W(ST_u(G, D')) + \text{dist}(v, u) + W(ST_u(G, D \setminus D'))$$

  Por indução, $W(ST_u(G, X)) \leq T(X, u)$, para todo $X \subset D$ e $u \in V \setminus K$ (o caso base é trivial). Segue imediatamente o limite superior.

  $$\forall D' \subseteq D, \forall u \in V \setminus K, \quad W(ST_v(G, D)) \leq T(D', u) + T(D \setminus D', u) + \text{dist}(v, u)$$

  $$\implies W(ST_v(G, D)) \leq T(D, v)$$

- **Limite Inferior** Para $ST_v(G, D \cup \{v\})$ uma árvore de Steiner ótima enraizada em v, temos as seguintes garantias.

  Seja u_0 o vértice mais próxima de v que possua mais de um vizinho. Certamente, esse vértice existe, já que $|D| > 1$. Como nenhum dos terminais possui mais de 1 vizinho, $u_0 \notin K$. Isso implica que u_0 possui pelo menos um filho, que denotaremos por u_1. Seja $D' \subseteq D$ os terminais filhos de u_1.

$$ST_v(G, D \cup \{v\}) = H(v \sim v_0) \cup ST_{u_0}(G, D \setminus D') \cup ST_{v_1}(G, D') \cup \{\{u_0, v_1\}\}$$
$$\implies W(ST_v(G, D)) = \text{dist}(v, v_0) + W(ST_{u_0}(G, D \setminus D')) + ST_{u_1}(G, D') + \text{dist}(u_0, u_1)$$

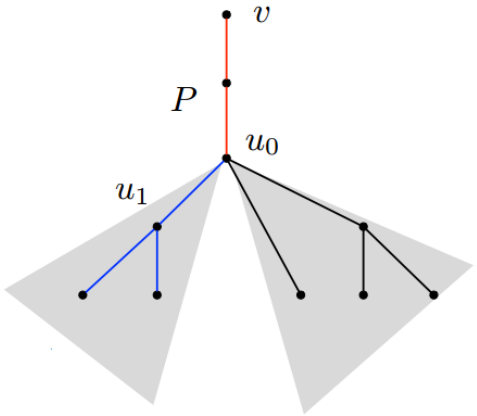

Figura 4: Decomposição da Árvore de Steiner

Como $\text{dist}(v, u_0)$ é mínimo, utilizando o mesmo raciocínio indutivo, temos que

$$\forall D' \subseteq D, \forall u \in V \setminus K$$
$$W(ST_v(G, D)) \geq T(D', u) + T(D \setminus D', u) + \text{dist}(v, u)$$
$$\implies W(ST_v(G, D)) \geq T(D, v)$$

■

2.1 Análise de Complexidade

A quantidade de operações realizadas para calcular $T(D, v)$ é $\mathcal{O}(|V|^{\mathcal{O}(1)}2^{|D|})$. Como no pior casos todos os estados da programação dinâmica podem ser calculados, a complexidade do programa é dada por

$$\sum_{v \in V \setminus K} \sum_{D \subseteq K} |V|^{\mathcal{O}(1)} 2^{|D|} \leq |V| \sum_{j=2}^{|K|} \binom{|K|}{j} 2^j |V|^{\mathcal{O}(1)} = 3^{|K|} |V|^{\mathcal{O}(1)}$$

Teorema 3 *O problema da árvore de Steiner, parametrizado pela quantidade de terminais, é FPT e pode ser resolvido em $\mathcal{O}(n^{\mathcal{O}(1)}3^k)$ de tempo e $\mathcal{O}(n2^k)$ de memória.*

3 Análise experimental

3.1 Implementação

O algoritmo foi implementado em **C++** com o auxílio da sua biblioteca padrão. O código fonte está (disponível aqui). O programa recebe como entrada o número de vértices n, o número de arestas m, o número de vértices terminais k, a lista de m arestas e o conjunto K de terminais. Cada aresta é uma lista de dois vértices, seguida do seu peso. Um exemplo de entrada é:

```
5 4 2
0 1 2
0 2 3
1 3 5
1 4 7
0 4 11
4 2 13
3 4
```

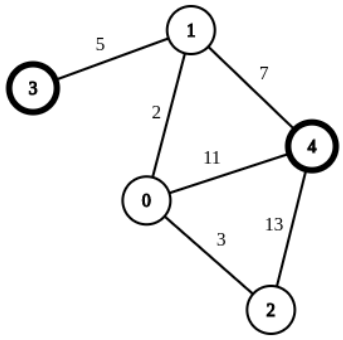

Figura 5: Exemplo de entrada.

3.2 Modelo de Grafo Aleatório

O algoritmo possui natureza exponencial e polinomial, visto que possui tempo de execução $\mathcal{O}(3^{|K|}n^{\mathcal{O}(1)})$. Para averiguarmos esse comportamento, construímos um gerador de grafos aleatórios, que recebe a quantidade de vértices e um parâmetro $p \in [0, 1]$ como entrada, e gera um grafo aleatório com probabilidade associada às suas arestas.

Grafos gerados pelo modelo seguem a seguinte única regra: $\forall i, \forall j, \mathbb{P}(\{i, j\} \in E(G)) = p$

Para entradas específicas como $p = 1$, por exemplo, sempre o grafo completo é gerado, ou, para $p = 0$, sempre o grafo vazio é gerado.

3.3 Testes Experimentais

É esperado que, variando-se n, a quantidade de vértices, ou m, a quantidade de arestas, o crescimento do tempo de execução seja limitado por um polinômio. Para o aumento da quantidade de vértices terminais, o crescimento do tempo de execução será exponencial. Nos três gráficos seguintes, avaliamos empiricamente o comportamento do algoritmo em três situações distintas.

Todos os dados usados para construção dos gráficos podem ser obtidos aqui.

Variação na quantidade de vértices

Para esse experimento, variamos o número de vértices do grafo, mantendo a proporção de arestas existentes e quantidade de terminais constantes. A Figura 6 apresenta a variação do tempo de execução em função apenas do número de vértices n.

Para esse caso em específico, fixamos p = 0.5 e k = 5. Podemos observar que o crescimento pode ser ajustado por função polinomial com o $R^2 \approx 1$. Com esse método, pudemos aproximar qual o possível expoente de n para esse tipo de grafo aleatório, visto a dificuldade analítica. O resultado obtido foi um polinômio de terceiro grau.

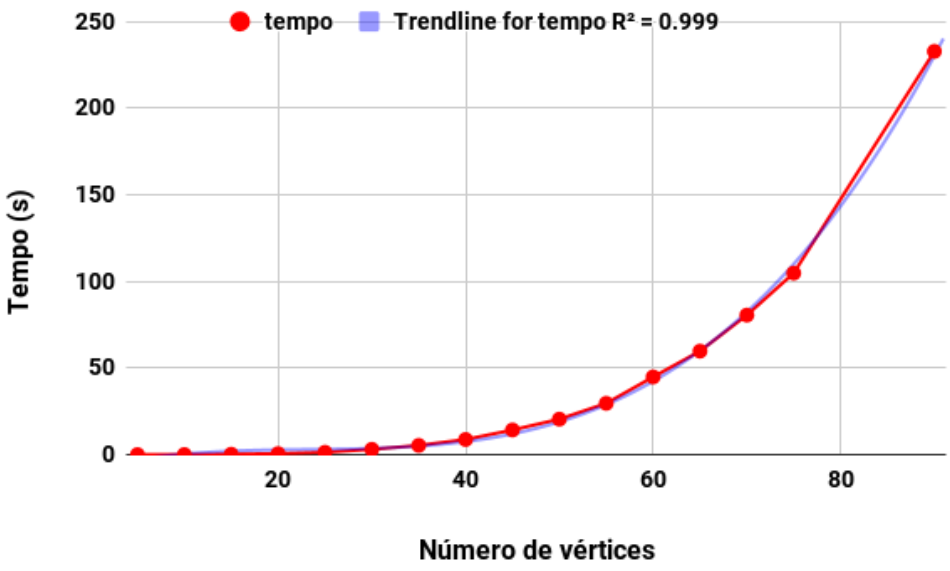

Figura 6: Variação do tempo de execução com o aumento do número de vértices.

Variação na quantidade de arestas

Para esse experimento, variamos o número de arestas do grafo, mantendo quantidade de vértices e de terminais constantes.

Para esse caso, fixamos $n = 40$ e $k = 5$. Podemos observar que o crescimento é dado por uma função aproximadamente linear. O único momento em que a quantidade de arestas é relevante para o problema é no calculo da distância mínima. Implementamos o Algoritmo Dijkstra mas, por termos uma quantidade muito pequena de vértices, seu impacto é aproximadamente linear.

A Figura 7 apresenta a variação do tempo de execução com o aumento do número de arestas m.

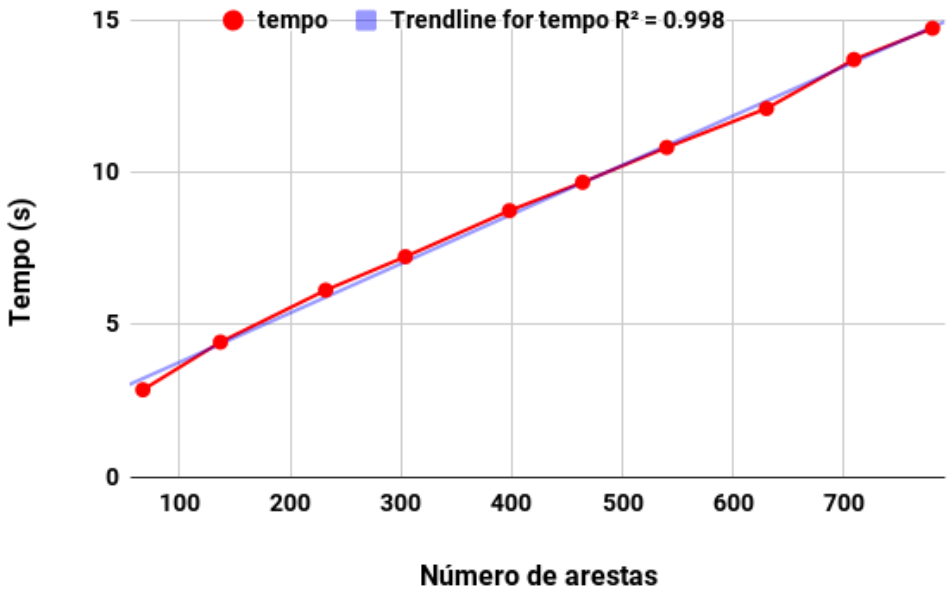

Figura 7: Variação do tempo de execução com o aumento do número de arestas.

Variação na quantidade de terminais

Para esse experimento, variamos o número de terminais, mantendo a proporção de arestas existentes e quantidade de vértices constantes.

Para esse caso, fixamos $n = 20$ e $p = 0.5$. Podemos observar que o crescimento é bem ajustado por uma função exponencial assim como previsto. Infelizmente, a não ser que P seja igual a NP, não há nenhum algoritmo que seja polinomial no número de terminais que resolva o problema da árvore de Steiner.

Figura 8 apresenta a variação do tempo de execução com o aumento do número de vértices terminais k.

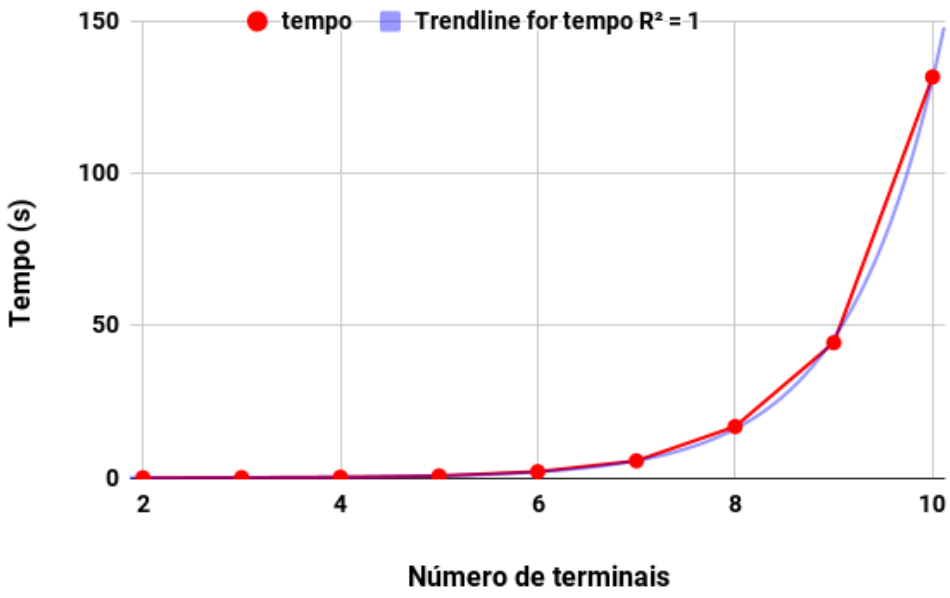

Figura 8: Variação do tempo de execução com o aumento do número de vértices terminais.

4 Conclusão

Neste trabalho, implementamos um algoritmo para encontrar Árvores Steiner em grafos ponderados. O algoritmo se baseia na modelagem do problema como um programação dinâmica e possui tempo de execução $\mathcal{O}(3^{|K|}n^{\mathcal{O}(1)})$. Além disso, realizamos experimentos para avaliarmos a proximidade entre a complexidade teórica do algoritmo e seu desempenho prático.

Apesar de existirem algoritmos mais eficientes para o problema, a experiência nos permitiu lidar com problemas práticos e teóricos interessantes. O estudo da teoria de grafos fundamental permitiu o desenvolvimento de propriedades e lemas para a construção do teorema principal. Finalmente, por ser um problema NP-Difícil, a dificuldade de lidar com grandes instâncias do problema foi muito maior, mas os resultados obtidos foram suficientes.

Referências

[1] Lukasz Kowalik Daniel Lokshtanov Dániel Marx Marcin Pilipczuk Michal Pilipczuk Marek Cygan, Fedor V. Fomin and Saket Saurabh. *Parameterized Algorithms*. Springer, 2016.

[2] Alessandro Santuari. *Steiner Tree NP-completeness Proof*, 2003.