Algoritmos de Aproximação e Problemas com Seqüências

IME - Instituto de Matemática e Estatística

Rafael Crivellari Saliba Schouery

Orientadora: Cristina Gomes Fernandes

Instituto de Matemática e Estatística
Universidade de São Paulo

FAPESP

MOTIVAÇÃO

Os métodos químicos conhecidos para seqüenciar DNA são eficazes apenas para uma pequena quantidade de bases, mas em geral desejamos seqüenciar uma quantidade muito maior de bases do que os métodos permitem. Para resolver este problema, quebramos aleatoriamente a mesma seqüência várias vezes em seqüências curtas o suficiente. Considere, por exemplo, a seqüência de DNA abaixo e três divisões da mesma:

AGCATGCTGCAGTCATGCTTAGGCTATTGGA
AGCATGCTGCAGTCATGCTTAGGCTATTGGA
AGCATGCTGCAGTCATGCTTAGGCTATTGGA
AGCATGCTGCAGTCATGCTTAGGCTATTGGA

Podemos utilizar as informações de redundância para tentar encontrar a seqüência original. Perceba que AGCATGC e TGCTGCA representam o trecho **AGCATGCTGCA** da seqüência original.
Desejamos, portanto, encontrar a menor seqüência de bases que contenha todos os pedaços seqüenciados.

FORMULAÇÃO DO PROBLEMA

Apresentamos agora algumas definições para formalizar o problema descrito acima.

Definição 1 (Superseqüência comum)**.**
*Uma seqüência σ é uma **superseqüência comum** para um conjunto de seqüências S se toda seqüência $s \in S$, s é uma subseqüência (contígua) de σ.*

O problema da superseqüência comum mínima, denotado por SCS, consiste em encontrar uma superseqüência comum para S de comprimento mínimo.

Definição 2 (Sobreposição)**.**
*Para duas seqüências s e t, a **sobreposição entre** s **e** t, denotada por $over(s,t)$ é a maior seqüência v tal que $s = uv$ e $t = vw$.*

Definição 3 (Composição)**.**
*Para duas seqüências s e t, a **composição de** s **e** t, denotada por $comp(s,t)$, é a seqüência uvw onde $s = uv$, $t = vw$ e $v = over(s,t)$.*

Definição 4 (Prefixo)**.**
*Para duas seqüências s e t, o **prefixo de** s **em relação a** t, denotada por $pref(s,t)$ é a seqüência u tal que $s = uv$ e $t = vw$ onde $v = over(s,t)$.*

Por exemplo, a sobreposição entre **AGCATGCTGC** e **TGCTGCAG** é TGCTGC, o prefixo é AGCA e a composição delas é AGCATGCTGCAG.

Um resultado que motiva o estudo deste problema foi apresentado por Gallant et al. [2]:

Teorema 1.
O SCS é NP-difícil.

Portanto, não existe um algoritmo polinomial para o SCS, a menos que P $=$ NP.

UM ALGORITMO GULOSO

Consideramos o algoritmo GREEDY que procede da seguinte forma: para um conjunto S de seqüências, escolha duas seqüências s e t tais que a sobreposição entre elas seja máxima. Faça a composição c entre as duas e então remova s e t e insira c em S. Repita o processo até restar uma única seqüência em S, esta é uma superseqüência comum para o conjunto original. Escrevendo formalmente temos:

GREEDY (S)
1 **enquanto** $|S| \neq 1$ **faça**
2 Seja (s,t) tal que $|over(s,t)|$ seja máximo
3 $S \leftarrow S \setminus \{s,t\} \cup \{comp(s,t)\}$
4 seja σ o único elemento de S
5 **devolva** σ

Exemplificamos abaixo a execução do algoritmo no conjunto $\mathbf{S} = \{$**AACGTC, ACTGAA, TCTGAC, GTCACT, GTCAGG**$\}$.

- Compomos GTCACT com ACTGAA.
- Compomos AACGTC com ACTGAA.
  Temos neste ponto o conjunto $T = \{$AACGTCAGG, TCTGAC, GTCACTGAA$\}$.
- Compomos GTCACTGAA com AACGTCAGG.
- Compomos TCTGAC a GTCACTGAACGTCAGG.
- Temos que $\sigma =$ TCTGACGTCACTGAACGTCAGG.

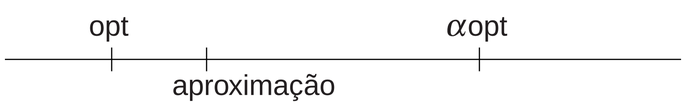
Este algoritmo nem sempre encontra uma solução ótima para o problema. Trata-se, na verdade, de um algoritmo de aproximação.

ALGORITMOS DE APROXIMAÇÃO

Algoritmos de aproximação são uma alternativa natural quando o problema é NP-difícil. Eles apresentam duas características fundamentais:

- São **eficientes**.
- Garantem a **qualidade da aproximação**.

Dizemos que um algoritmo é uma α-aproximação quando o valor da solução encontrada sempre se encontra entre o valor ótimo e α vezes o valor ótimo. A figura abaixo esquematiza este fato:



O teorema abaixo, provado por Blum et al. [1], mostra a razão de aproximação do GREEDY.

Teorema 2.
O algoritmo GREEDY é uma **4-aproximação** para o SCS.

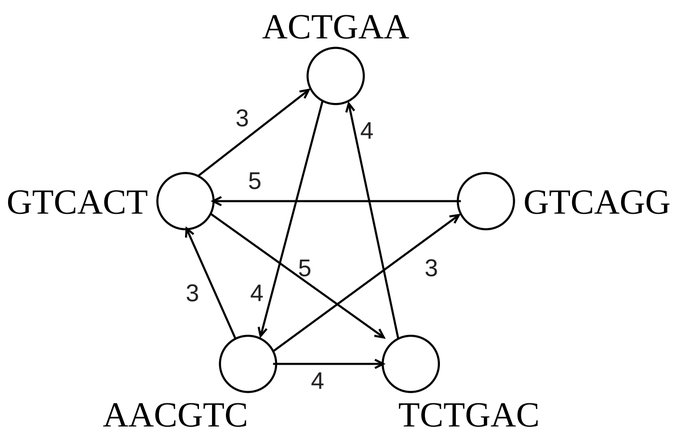
Portanto, a seqüência encontrada pelo GREEDY para um conjunto S nunca é maior do que 4 vezes o tamanho de uma superseqüência comum mínima para S.

REPRESENTAÇÃO ATRAVÉS DE GRAFOS

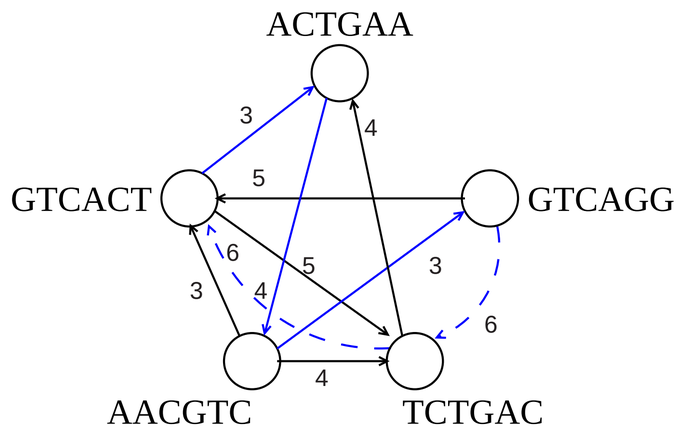
Existe uma forte relação entre o problema e grafos completos orientados com custos.

Definição 5 (Digrafo de prefixo)**.**
*Para um conjunto S de seqüências, definimos o **digrafo de prefixo** de S como o digrafo completo onde os vértices são as seqüências de S. Um arco (u,v) deste digrafo tem custo $|pref(u,v)|$.*

Para o nosso exemplo, temos o seguinte digrafo, onde os arcos de custo 6 foram omitidos.



Podemos perceber uma relação entre o SCS e o TSP, o problema do caixeiro viajante. No TSP desejamos encontrar um circuito que passe por todos os vértices (chamado de circuito hamiltoniano) e tenha custo mínimo. A figura abaixo mostra o caminho hamiltoniano encontrado para pela execução do GREEDY no nosso exemplo.



Esta relação entre os problemas é utilizada para provar diversos resultados para o SCS.

CONCLUSÃO

Existe uma conjectura de que o GREEDY é uma 2-aproximação para o SCS. De fato, o GREEDY não é melhor do que uma 2-aproximação, como pode ser verificado através da instância $S = \{c(ab)^k, (ba)^k, (ab)^k c\}$. O GREEDY encontra a superseqüência comum $c(ab)^k c(ba)^k$, mas a superseqüência comum mínima σ^* para S é da forma $c(ab)^{k+1}c$. Portanto temos que:

$$\frac{\sigma}{\sigma^*} = \frac{4k+2}{2k+4} \approx 2$$

Kaplan et al. [3] mostraram que o GREEDY é melhor do que uma 4-aproximação.

Teorema 3. O algoritmo GREEDY é uma $3\frac{1}{2}$-aproximação para o SCS.

Sweedyk [4] criou a melhor aproximação conhecida para o SCS.

Teorema 4. Existe uma $2\frac{1}{2}$-aproximação para o SCS.

A conjectura sobre o GREEDY é importante, pois se for provada teremos um algoritmo fácil de implementar e rápido, mas que também seria a melhor aproximação conhecida para o SCS.

REFERÊNCIAS

[1] A. Blum, T. Jiang, M. Li, J. Tromp, and M. Yannakakis. Linear approximation of shortest superstrings. *Journal of the Association for Computing Machinery*, 41(4):630–647, 1994.

[2] J. Gallant, D. Maier, and J.A. Storer. On finding minimal length superstrings. *Journal of Computer and System Sciences*, 20(1):50–58, 1980.

[3] H. Kaplan and N. Shafrir. The greedy algorithm for shortest superstrings. *Information Processing Letters*, 93(1):13–17, 2005.

[4] Z. Sweedyk. A $2\frac{1}{2}$-approximation algorithm for shortest superstring. *SIAM Journal on Computing*, 29(3):954–986 (electronic), 2000.