

Algoritmos de Aproximação e Problemas de *Clustering*

Samuel Praça de Paula, Orientadora: Professora Cristina Gomes Fernandes
Departamento de Ciência da Computação, Instituto de Matemática e Estatística, Universidade de São Paulo

Introdução

Estamos acostumados à ideia de que há muitos problemas práticos que podem ser modelados e resolvidos como problemas de computação. Há uma imensa variedade de problemas reais que podem ser modelados como problemas de *otimização combinatória*. No entanto, muitos dos problemas mais interessantes de otimização combinatória são *NP-difíceis*, o que significa que, a menos que $P = NP$, não existem algoritmos *eficientes*, isto é, de complexidade de tempo polinomial, para resolvê-los.

Se estamos interessados em resolver um problema real, geralmente é importante que consigamos garantir a obtenção de uma resposta em tempo razoável. Uma das maneiras de enfrentar tal barreira são os *algoritmos de aproximação*: algoritmos eficientes que encontram soluções aproximadas para esses problemas. Ou seja, tentamos encontrar rapidamente uma “boa” solução, ainda que não necessariamente uma solução *ótima*.

Algoritmos de aproximação garantem sempre encontrar soluções a um determinado fator do ótimo. É importante notar que, para alguns problemas, podemos não conseguir algo do tipo. Para o clássico problema do Caixeiro Viajante, por exemplo, encontrar uma solução a um fator constante do ótimo é tão difícil quanto resolver o problema de maneira exata.

Objetivos

Neste trabalho nos dedicamos a estudar a área de Algoritmos de Aproximação, buscando conhecer tanto o conteúdo mais clássico da literatura quanto resultados novos, obtidos em pesquisas recentes. Os resultados referem-se tanto a algoritmos para aproximar problemas quanto demonstrações sobre a *difículdade* de problemas, isto é, resultados de *inaproximabilidade*.

Em especial, nos concentramos em resultados referentes a problemas de *clustering* (agrupamento). Trata-se de uma categoria que engloba muitos problemas diferentes, úteis para modelar problemas como categorização automática e *facility location* (localização de instalações). Em nosso trabalho, têm especial destaque o *k-center* e algumas de suas generalizações.

Todo o conteúdo estudado é apresentado em nosso trabalho de maneira adaptada, de modo a tentar reunir resultados afins de maneira mais confortável à leitura — por exemplo, através de eventuais modificações de notação e definições, fornecimento de demonstrações ausentes nos artigos originais, etc., além da tradução para o português.

Definições

Um problema de otimização combinatória é um conjunto de instâncias. A cada instância I , está associado um conjunto $\text{Sol}(I)$ de *soluções viáveis* e uma função **val** que, a cada $S \in \text{Sol}(I)$, associa um *valor*, $\text{val}(S)$.

Dada uma instância I , desejamos encontrar uma solução $S^* \in \text{Sol}(I)$ tal que $\text{val}(S^*)$ seja

- mínimo, caso o problema seja de minimização
- máximo, caso o problema seja de maximização

O valor $\text{val}(S^*)$ de uma tal solução é chamado de *valor ótimo* da instância, e denotado por $\text{OPT}(I)$.

Um *algoritmo de aproximação com garantia de desempenho* α para um dado problema de otimização combinatória, ou simplesmente α -*aproximação*, é um algoritmo polinomial que, dada uma instância I do problema, devolve uma solução S com

- $\text{val}(S) \leq \alpha \cdot \text{OPT}(I)$, se o problema é de minimização (e $\alpha > 1$)
- $\text{val}(S) \geq \alpha \cdot \text{OPT}(I)$, se o problema é de maximização (e $\alpha < 1$)

Um dado problema A é dito *NP-difícil* se existe uma redução polinomial de um problema NP-completo a A . Em outros termos, se há algoritmo eficiente para A , então há também para todo problema na classe NP (inclusive os NP-completos). Isso implica que encontrar um algoritmo eficiente para resolver A equivale a provar que $P = NP$.

Informações e contato

Para mais informações, acesse a página do trabalho:
<http://www.linux.ime.usp.br/~samuel/mac499>

Endereço para contato: samuel@linux.ime.usp.br

Problemas de clustering

Problemas de *clustering* são problemas em que temos alguns elementos comparáveis entre si e desejamos particioná-los em grupos (*clusters*) de acordo com suas semelhanças. Um bom particionamento nos dá grupos homogêneos, isto é, em que os elementos colocados em um mesmo *cluster* são parecidos entre si.

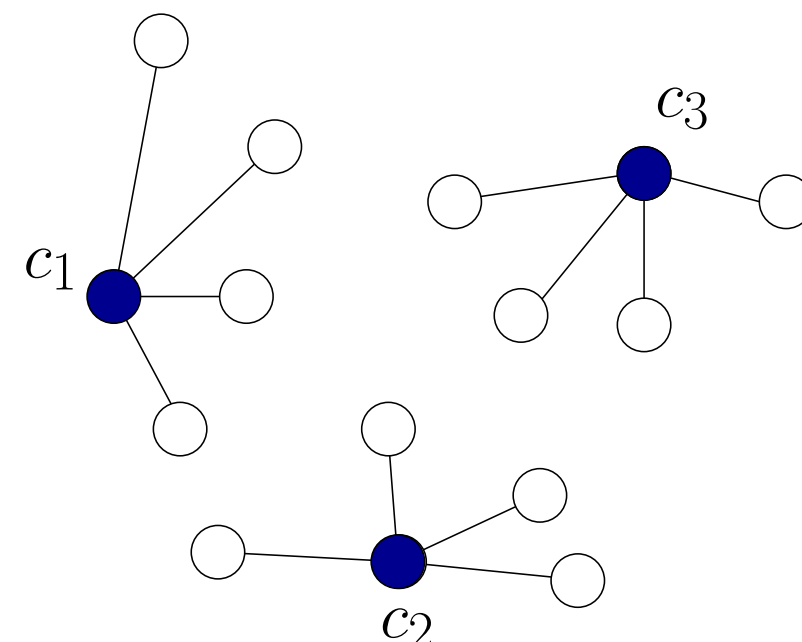


Figura: *k-center*: partição resultante da escolha de 3 centros.

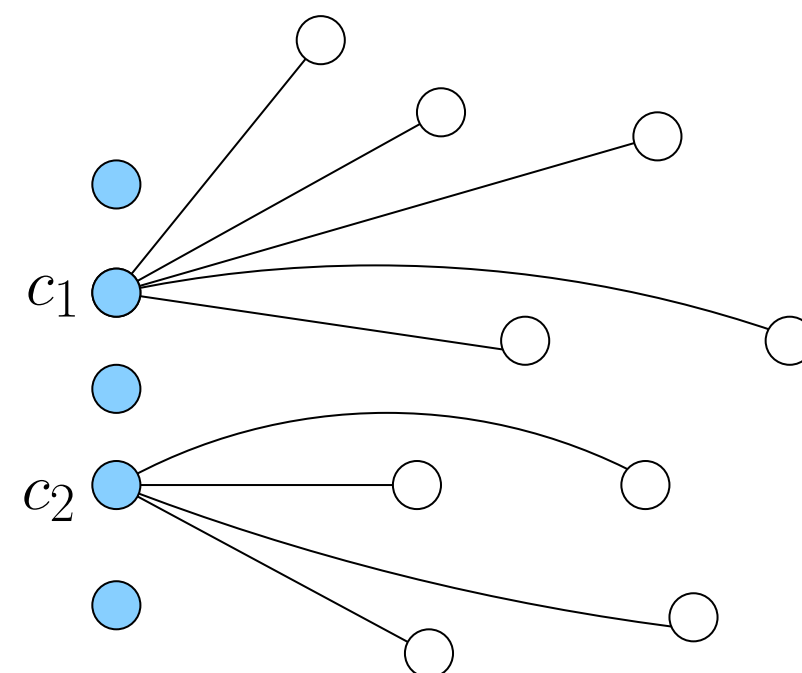


Figura: *k-supplier*: apenas os vértices azuis podem ser centros.

k-center: Dados um grafo $G = (V, E)$, completo, custos $d(i, j) \geq 0$ para cada aresta ij e um inteiro k , desejamos escolher k vértices para serem os *centros* de maneira que **a maior distância de um vértice ao centro mais próximo seja mínima**.

k-supplier: Semelhante ao *k-center*, com a diferença de que temos um grafo $\{U, W\}$ -bipartido e os centros podem ser escolhidos apenas dentre os vértices que estejam em U . Além disso, os vértices em U que não são escolhidos como centros são ignorados: eles não precisam estar próximos de nenhum centro.

Podemos interpretar um *cluster* como o conjunto dos vértices que se conectam a um dado centro. Os vértices sempre “escolhem” se conectar ao centro mais próximo, como nas figuras ao lado.

O *k-center* e o *k-supplier* são problemas de *clustering* bastante básicos, e são ambos **NP-difíceis**, assim como suas generalizações.

Algoritmos e resultados

Para aproximar os problemas acima, consideramos o caso métrico, isto é, em que o grafo $G = (V, E)$ dado tem custos que respeitam a **desigualdade triangular**:

$$d(i, j) \leq d(i, k) + d(k, j), \quad \forall i, j, k \in V$$

Para cada $e = ij \in E$, defina $c_e = d(i, j)$. Ordene as arestas do grafo de modo que $c_1 \leq c_2 \leq \dots \leq c_m$. Defina o grafo G_i como o subgrafo de G em que preservamos todos os vértices, e retiramos as arestas de custo maior que c_i . Defina, por fim, $G_i^2 = (V, E')$, em que dois vértices estão conectados por uma aresta caso estejam a distância no máximo 2 em G_i .

Para $i = 1, 2, \dots, m$, construímos um conjunto $I \subseteq V$ independente maximal em G_i^2 . Ou seja, um conjunto de vértices que estão a distância maior que 2 no grafo G_i .

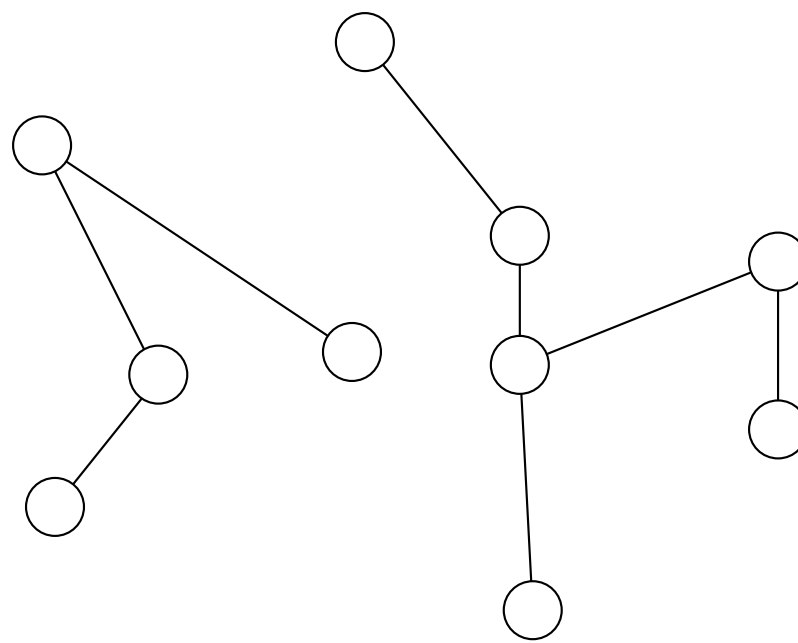


Figura: Exemplo de grafo G_i .

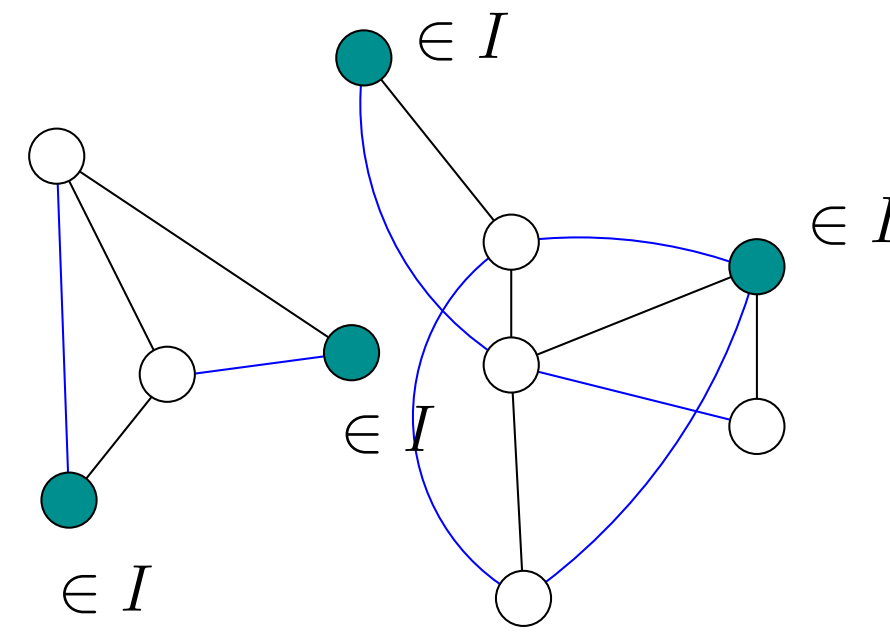


Figura: G_i^2 correspondente, e um conjunto independente maximal é indicado.

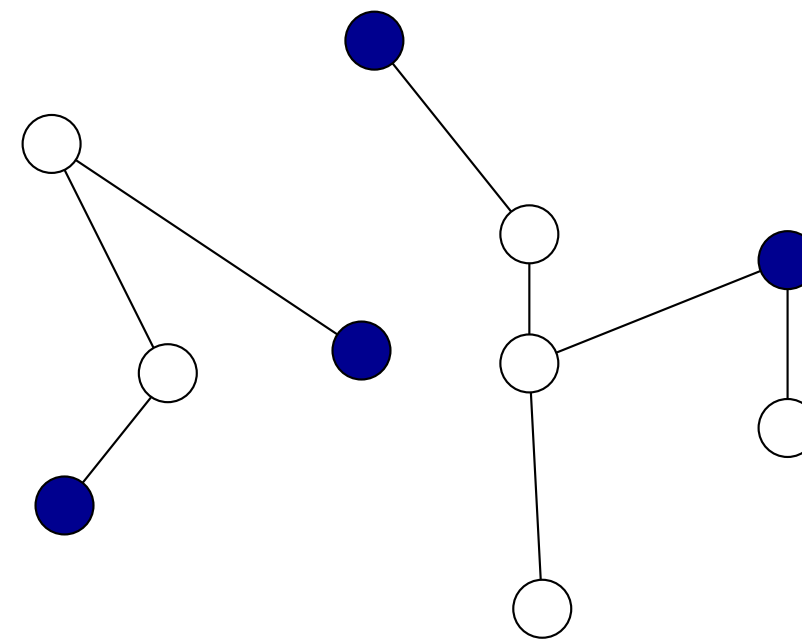


Figura: Os centros escolhidos a partir de I .

Para o *k-center*, se $|I| > k$, o valor ótimo OPT é estritamente maior que c_i . O algoritmo vai para o próximo i e tenta de novo.

Se, no entanto, $|I| \leq k$, escolha cada $v \in I$ como um centro.

Para o primeiro i para o qual a iteração seja bem-sucedida, vale que todo vértice está a distância no máximo 2, em G_i , de algum centro.

Pela desigualdade triangular, o custo de uma aresta entre um vértice qualquer e seu centro mais próximo não pode ser superior a $2 \cdot c_i$. O valor da solução, assim, é

$$\text{val}(S) \leq 2 \cdot c_i \leq 2 \cdot \text{OPT}.$$

Esse algoritmo, portanto, é uma 2-aproximação. Sabe-se que esse fator é o melhor possível para o *k-center* a menos que $P = NP$!

Hochbaum e Shmoys [2] descrevem explicitamente essa técnica de percorrer os custos construindo os grafos G_i para buscar uma solução. Os autores a aplicam a diversos problemas, obtendo a melhor razão possível (a menos que $P = NP$) para a maioria destes, incluindo o *k-supplier*.

Referências

- [1] D.P. Williamson, and D.B. Shmoys, “**The Design of Approximation Algorithms**,” *Cambridge University Press*, 2011.
- [2] D.Hochbaum, and D.B.Shmoys, “**A unified approach to approximation algorithms for bottleneck problems**,” in *Journal of the ACM*, 1986, volume 33, pp. 533-550.
- [3] T.H.Cormen, C.E.Leiserson, R.L.Rivest, and C. Stein, “**Introduction to algorithms**,” *The MIT Press*, ed. 3, 2009.