Facility location problems: A parameterized view

Michael R. Fellows e Henning Fernau

Seminário por

Marcelo Pinheiro Leite Benedito

Instituto de Computação
UNICAMP

13 de dezembro de 2016

Introdução

- Inicia estudo parametrizado do *Facility Location Problem* (FLP)
- Dos autores Michael R. Fellows e Henning Fernau, no ano de 2011
- Apresenta vários resultados de complexidade parametrizada:
  - Diversos algoritmos de kernelização sobre diferentes parâmetros
  - Resultados relacionados com o limite inferior do tamanho de kernel
  - Exemplos de variantes e aplicações do FLP

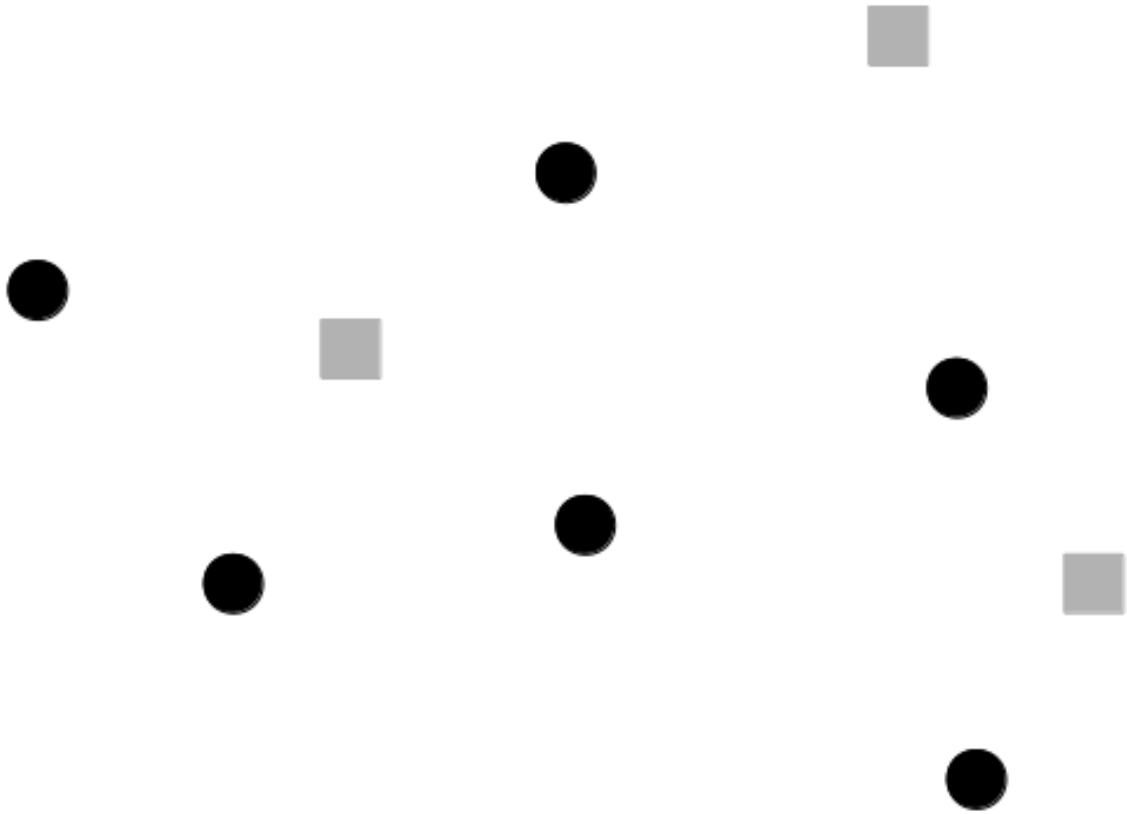
Facility Location Problem

Introdução

- Modela o seguinte problema:
  - Uma empresa deseja abrir instalações para atender clientes em uma região
  - Todos os clientes devem ser atendidos por alguma instalação
  - Existe um custo para a abertura de instalações e atendimento de clientes
  - Deseja-se minimizar o custo total

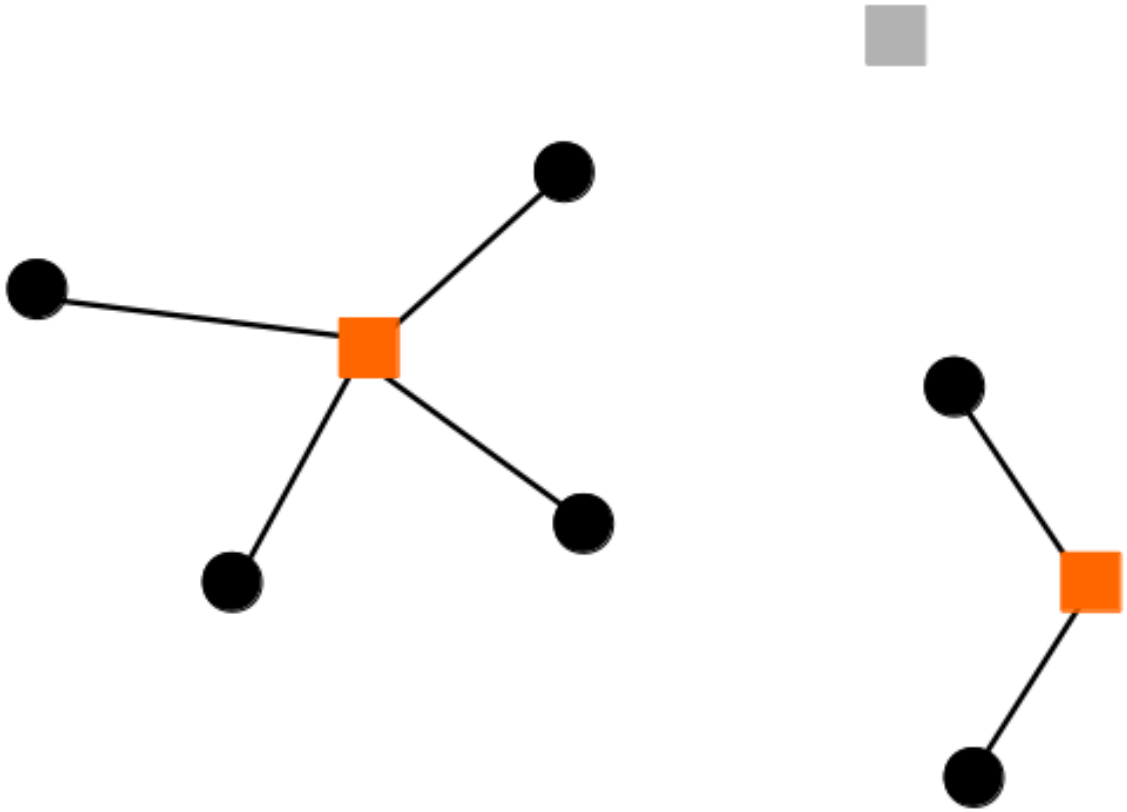
Facility Location Problem

Introdução



Facility Location Problem

Introdução



Facility Location Problem

Histórico

- Primeiros estudos na década de 60
- Algoritmos de aproximação na década de 80
- Atualmente, temos:
  - Algoritmo 1.488-aproximado [3]
  - Limite de aproximabilidade de 1.463 [2]
- FLP é NP-difícil [1]

Facility Location Problem

Histórico

- Problema de localização amplamente estudado:
  - Heurísticas
  - Algoritmos de aproximação
  - Técnicas envolvendo programação linear inteira

Facility Location Problem

Definições

- Entrada:
  - Grafo bipartido completo $B = (F \cup C, E)$
    - Conjunto F de potenciais instalações, com $|F| = m$
    - Conjunto C de clientes, com $|C| = n$
  - Funções de custo $\omega_F : F \to \mathbb{N}_{\geq 1}$ e $\omega_E : E \to \mathbb{N}_{\geq 1}$
  - Custo máximo $k \in \mathbb{N}$

Facility Location Problem

Definições

- Saída:
  - Conjunto $F' \subseteq F$ de instalações abertas
  - Conjunto $E' \subseteq E$ que indica qual instalação atende cada cliente
- Objetivo: minimizar o custo de abertura de instalações e de atendimento a clientes

Facility Location Problem

Definições

- Forma alternativa de representar os custos: $M \in \mathbb{N}_{\geq 1}^{(n+1)\times m}$
  - Linha 0: custo de abertura das instalações
  - Demais linhas: custo de conexão
- Com função $s : C \to F'$, pergunta-se:

$$\sum_{f \in F'} M[0, f] + \sum_{c \in C} M[c, s(c)] \leq k?$$

Abordagens iniciais

- Parâmetros naturais:
  - Número n de clientes
  - Número m de instalações
  - Limite superior do custo k
  - Limite superior de instalações abertas

Abordagens iniciais

- Parâmetros naturais:
  - Número n de clientes
  - **Número m de instalações**
  - Limite superior do custo k
  - Limite superior de instalações abertas

Abordagens iniciais

Teorema 1

O FLP pode ser resolvido em tempo $\mathcal{O}^*(2^m)$.

Demonstração.

Para cada $F' \subseteq F$, avalia-se custo de solução, em tempo polinomial. Observe que, dado F', podemos encontrar E' facilmente. O tempo de execução do algoritmo é dado pelo número de subconjuntos de F: $2^{|F|} = 2^m$. □

Abordagens iniciais

- Parâmetros naturais:
  - Número n de clientes
  - Número m de instalações
  - **Limite superior do custo** k
  - Limite superior de instalações abertas

Abordagens iniciais

Regras de redução

Regra de Redução 1

Se dada uma instância (M, k) *com* $M \in \mathbb{N}_{\geq 1}^{(n+1)\times m}$ *e vale que* $n > k$*, então retorne* **não**.

Lema 1

A Regra 1 é válida.

Abordagens iniciais

Regras de redução

Demonstração.

Se cada cliente possui custo de atendimento de pelo menos uma unidade e todos devem ser atendidos, então não é possível atender a todos quando $n > k$. □

Abordagens iniciais

Regras de redução

Lema 2

Quando não for mais possível a aplicação da Regra 1, então a instância reduzida (M, k) não tem mais que $(k+1)$ linhas. □

- Dada instalação f, define-se o vetor $v_f = M[0 \ldots n][f]$
- Se é válido que $v_f \leq v_g$, então v_f é menor ou igual a v_g posição a posição

Abordagens iniciais

Regras de redução

Regra de Redução 2

Se existem instalações f e g tal que $v_f \leq v_g$, então remova g da instância sem mudança em k.

Lema 3

A Regra 2 é segura.

Abordagens iniciais

Regras de redução

Demonstração.

Se temos uma solução S com instalações f e g e vale que $v_f \leq v_g$, então uma solução S' obtida removendo-se g de F' e adicionando f (consequentemente servindo os clientes antes atendidos por g) não pode ter valor de solução maior do que S. □

Abordagens iniciais

Regras de redução

Regra de Redução 3

Em uma instância $((B, \omega_E, \omega_F), k)$, as modificações abaixo não mudam o parâmetro:

Algoritmo 1

1: **para cada** instalação f **faça**
2: **se** $\omega_F(f) \geq k$ **então** remova f
3: **fim se**
4: **para cada** cliente c **faça**
5: **se** $\omega_F(f) + \omega_E(c, f) > k + 1$ **então**
6: $\omega_E(c, f) := k + 1 - \omega_F(f)$
7: **fim se**
8: **fim para**
9: **fim para**

Abordagens iniciais

Regras de redução

Lema 4

A Regra 3 é segura.

Demonstração.

Uma instalação com custo de abertura k não pode servir nenhum cliente, porque isso implicaria em uma solução de custo pelo menos $k+1$. Se uma instalação não é tão cara mas somada com o custo de atendimento passa o limite superior do parâmetro, então podemos diminuir este valor para $k+1$, que ainda é inviável. □

Abordagens iniciais

Kernelização com Lema de Dickson

- Tentaremos limitar o número de instalações após a aplicação das regras com o seguinte lema:

> **Lema de Dickson**: dada constante n, o conjunto $(\mathbb{N}^n, \leq)$ possui tamanho finito.

Abordagens iniciais

Kernelização com Lema de Dickson

Teorema 2

Facility Location é $\mathcal{FPT}$ quando parametrizado com o custo k.

Demonstração.

Seja (M, k) uma instância já reduzida com as regras 1-3. Pelo Lema 2, M não tem mais que $k + 1$ linhas, logo, cada instalação possui um vetor de custos deste tamanho. Também sabemos que os vetores das instalações não são comparáveis, pela redução 2.

Abordagens iniciais

Kernelização com Lema de Dickson

Demonstração (Cont.)

Pelo Lema de Dickson, o número de vetores é limitado por uma função $g(k)$, já que usamos a redução 3. Então, a matriz M tem, no máximo, $(k+1)g(k)$ elementos. □

- Com isso, temos $g(k)$ vetores de tamanho $k+1$ não comparáveis
- Cada posição desse vetor tem valor variando de 0 a $k+1$ (regra 3)

Abordagens iniciais

Kernelização com Lema de Dickson

- Logo, existem $(k+2)^{k+1}$ vetores deste tipo, um para cada instalação possível

Corolário 1

Facility Location pode ser resolvido em tempo $\mathcal{O}^*(2^{(k+2)^{k+1}})$.

Refinamentos

- Podemos ver uma solução do FLP como uma partição do conjunto de clientes
- Os clientes de uma partição são atendidos pela mesma instalação
- Dados F' e E', obtemos as partições de C polinomialmente

Refinamentos

Lema 5

Dada instância (M, k), Facility Location pode ser resolvido em tempo $\mathcal{O}(k^k \, poli(g(k)) + nm)$, onde $poli(\cdot)$ é um polinômio e $g(k)$ limita o número de instalações.

Refinamentos

Demonstração.

As regras 1 e 2 são executadas em tempo $\mathcal{O}(nm)$. Sabe-se que o número de partições é, no máximo, k^k, que é um limitante do Número de Bell. Cada partição é servida por uma única instalação, então calculamos o seu custo em tempo polinomial em $\mathcal{O}(\text{poli}(g(k)))$. □

Refinamentos

Lema 6

Se tivermos instância (M, k) *reduzida com as regras 1-3, então* $nm \leq (k+1)^{k+2}$.

Demonstração.

Pela regra 1, temos, no máximo, k clientes; pela regra 3, existem $(k+1)^{k+1}$ vetores distintos referentes a instalações. Pela regra 2, cada instalação deve possuir vetor diferente, logo é possível a existência de $(k+1)^{k+1}$ instalações. Como cada vetor de instalação tem $k+1$ elementos, M tem, no máximo, $(k+1)^{k+2}$ elementos. □

Refinamentos

- Até agora, obtemos kernels grandes
- Tentaremos aprimorar com programação dinâmica

Programação Dinâmica

Pré-processamento

- Para cada $X \subseteq C$, defina $OS(X)$ como o custo de atender os clientes de X e abrir a instalação que resulta em custo total mínimo
- Sendo $F_X(f)$ o custo de abrir f e servir X através dela, temos:

$$OS(X) := \min_{f \in F}(F_X(f))$$

Programação Dinâmica

- Defina $s(X)$: custo de atender X com pelo menos uma instalação
- Caso base:

$$s(\emptyset) = 0$$

$$s(X) = OS(X), \text{ se } |X| = 1$$

- Caso geral:

$$s(X) := \min_{\emptyset \subseteq Y \subseteq X}(OS(Y) + s(X \setminus Y))$$

Programação Dinâmica

Algoritmo 2

1: **se** $|C| > k$ **então** retorne não
2: **fim se**
3: $s(\emptyset) := 0$
4: **para cada** $X \subseteq C$ **faça**
5: calcule $OS(X)$ em tempo $\mathcal{O}(|X||F|)$
6: **fim para**
7: **para** $i := 1 \ldots |C|$ **faça**
8: **para cada** $X \subseteq C$ com $|X| = i$ **faça**
9: $s(X) := \min_{\emptyset \subseteq Y \subseteq X}(OS(Y) + s(X \setminus Y))$
10: **fim para**
11: **fim para**

Programação Dinâmica

Tempo de execução

- Pré-processamento: $\mathcal{O}(2^{|C|}|F|) \subseteq \mathcal{O}(2^k m)$
- Algoritmo: $\mathcal{O}(3^n) \subseteq \mathcal{O}(3^k)$
- Tempo final: $\mathcal{O}(2^k m + 3^k)$

Teorema 3

Facility Location pode ser resolvido em tempo $\mathcal{O}(2^k m + 3^k)$ em uma instância (M, k). □

Programação Dinâmica

Tempo de execução

- Aplicando o Lema 6, o algoritmo possui tempo:

$$\mathcal{O}^*(2^k(k+1)^{k+2}) \subseteq \mathcal{O}^*((2k)^k)$$

Conclusões

- Primeiro (e único) estudo parametrizado do FLP
- Fornece diversos resultados iniciais
- Exemplos de aplicação e variantes do FLP

Conclusões

- Algumas possíveis melhorias:
  - É possível melhorar os algoritmos e kernels fornecidos para o FLP e variantes?
  - Existem parâmetros mais adequados para se utilizar?
  - Talvez seja possível diminuir o tempo de execução dos algoritmos introduzindo uma noção de vizinhança

Exercício

Durante todo o trabalho, consideramos que os custos de abertura de instalações e de conexão entre clientes e instalações são números naturais. Se esses valores fossem números racionais, explique o motivo das abordagens abaixo funcionarem ou não com essa mudança:

(a) Kernelização com Lema de Dickson em tempo $\mathcal{O}^*(2^{(k+2)^{k+1}})$;

(b) Programação dinâmica em tempo $\mathcal{O}(2^k m + 3^k)$.

Referências

Cornuéjols, G., Nemhauser, G. L., and Wolsey, L. A.
The uncapacitated facility location problem.
Tech. rep., DTIC Document, 1983.

Guha, S., and Khuller, S.
Greedy Strikes Back: Improved Facility Location Algorithms.
In *Proc. of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms* (Philadelphia, PA, USA, 1998), SODA '98, Society for Industrial and Applied Mathematics, pp. 649–657.

Li, S.
A 1.488 Approximation Algorithm for the Uncapacitated Facility Location Problem.
In *Automata, Languages and Programming* (2011), L. Aceto, M. Henzinger, and J. Sgall, Eds., vol. 6756 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, pp. 77–88.