Facility location problems: A parameterized view

Michael R. Fellows e Henning Fernau (2011)

Resumo para Algoritmos Parametrizados

Marcelo Pinheiro Leite Benedito

13 de dezembro de 2016

Resumo

Neste trabalho, descrevemos brevemente alguns resultados obtidos em *Facility location problems: A parameterized view* [1], por Michael R. Fellows e Henning Fernau, onde são introduzidas as primeiras tentativas de resolução do problema a partir do ponto de vista parametrizado.

1 Introdução

O *Facility Location Problem* (FLP) é amplamente estudado na literatura de diversas áreas da computação e busca modelar o seguinte cenário: uma empresa deseja abrir uma quantidade de instalações para atender seus clientes. Há custo tanto para abrir uma nova instalação quanto para servir um cliente através de uma instalação específica. O objetivo é servir todos os clientes minimizando o custo total. Diversos outros problemas NP-difíceis podem ser modelados como o FLP, o que torna ainda mais útil o estudo de sua complexidade parametrizada. Abaixo temos as principais formas de se tratar um problema NP-difícil:

A. **heurísticas**: espera-se que produzam boas soluções em pouco tempo, porém não garantem a qualidade dessas soluções;

B. **algoritmos de aproximação**: executam em tempo polinomial e garantem soluções que possuem valores aproximados do ótimo, mas nem sempre essa aproximação é boa, isso é, com fator pequeno;

C. **programação linear inteira (PLI)**: não possuem garantia de tempo de execução polinomial, mas se o algoritmo termina, produz solução ótima.

Este trabalho se alinha com a última estratégia, pois o problema é formulado seguindo a estratégia de PLI e é investigado considerando algumas parametrizações naturais, como custo e número de instalações. Por ser o primeiro trabalho que trata o FLP de forma parametrizada, apresentamos resultados "triviais", mas que são importantes para se dar início ao estudo.

Sobre o ponto de vista de aproximação, o FLP é um dos problemas mais estudados da área, com algoritmos criados a partir de diversas técnicas. Atualmente, o melhor algoritmo de aproximação para este problema, no caso métrico, possui fator 1.488 [3] e limite de aproximabilidade de 1.463 [2].

1.1 Definições

Dado um grafo bipartido $B = (F \cup C, E)$, sendo F o conjunto de instalações, C o de clientes e E o conjunto de arestas do tipo (f, c), que indica que o cliente c pode ser atendido por uma instalação f; funções de custo $\omega_F : F \to \mathbb{N}_{\geq 1}$ e $\omega_E : E \to \mathbb{N}_{\geq 1}$, representando o custo de abertura de instalações e o custo de atendimento a clientes por instalações, respectivamente. Além disso, é dado um custo máximo $k \in \mathbb{N}$. Devemos determinar um conjunto de instalações a serem abertas $F' \subseteq F$ e arestas $E' \subseteq E$ tal que:

A. $\forall f \in F(f \in F' \iff \exists e \in E'(f \in e))$

B. $\forall c \in C, \exists e \in E'(c \in e)$

C. $\sum_{f \in F'} \omega_F(f) + \sum_{e \in E'} \omega_E(e) \leq k$

Podemos formular o problema em termos da representação de matrizes dos custos de abertura de instalações e custos de atendimento. Dados $M \in \mathbb{N}_{\geq 1}^{(n+1)\times m}$, indexada de forma $M[0 \ldots n][1 \ldots m]$, e $k \in \mathbb{N}$, existe $F' \subseteq \{1, \ldots, m\}$ e função $s : C \to F'$ tal que $\sum_{f \in F'} M[0, f] + \sum_{c \in C} M[c, s(c)] \leq k$?

Nesta formulação, as colunas são as potenciais instalações e as linhas são os clientes. Na matriz M, a linha de índice 0 possui os custos de abertura de cada instalação, enquanto as restantes indicam o custo de atendimento do cliente (índice da linha) por uma instalação específica (índice da coluna). Eventuais arestas faltantes no grafo de entrada B podem ter valor maior que k na matriz M, indicando inviabilidade.

2 Abordagens iniciais

Existem alguns parâmetros naturais para o problema, que são: o número n de clientes, o número m de potenciais instalações, o limite superior k do custo e o limite superior l de instalações que podem ser abertas. A primeira abordagem, com força bruta, nos fornece o seguinte resultado:

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

Demonstração. Para cada $F' \subseteq F$, avalia-se o valor obtido (em tempo polinomial) e devolve o subconjunto que leva a uma melhor solução. Observe que, dado F', podemos encontrar em tempo polinomial o conjunto E' de custo mínimo, bastando conectar cada cliente a uma instalação de F' □

Em relação aos outros parâmetros, as abordagens são menos claras. Neste trabalhos, exploraremos o parâmetro k, que é uma escolha natural para problemas de minimização.

2.1 Regras de Redução

Um algoritmo de kernelização transforma uma instância (I, k) em uma (I', k') em tempo polinomial, sendo que o tamanho da nova instância é limitado por uma função de k. Sabe-se que um problema é $\mathcal{FPT}$ se, e somente se, ele admite um algoritmo de kernelização. Esses algoritmos geralmente são baseados em regras de redução, por isso, elas são importantes nesse tipo de estudo.

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.*

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 $n > k$. □

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.*

Note que este Lema nos dá forte ligação entre os parâmetros k e n. Podemos definir os custos de uma instalação f como o vetor $v_f = M[0\ldots n][f]$; quando aplicamos um operador de comparação entre vetores, entende-se que estão sendo comparados elementos de uma mesma posição.

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.*

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. □

Regra de Redução 3. *Em uma instância $((B,\omega_E,\omega_F),k)$, as modificações abaixo não mudam o parâmetro:*

Algoritmo 1

para cada instalação f **faça**
  se $\omega_F(f) \geq k$ **então** remova f
  fim se
  para cada cliente c **faça**
    se $\omega_F(f) + \omega_E(c,f) > k+1$ **então** $\omega_E(c,f) \coloneqq k+1-\omega_F(f)$
    fim se
  fim para
fim para

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. □

2.2 Kernelização com Lema de Dickson

Nesta seção, usaremos o Lema de Dickson para limitar o número de instalações que podem existir após a aplicação da regra de redução 3, ou seja, o número de colunas de M.

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

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. 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 $f(k) = (k+1)g(k)$, o tamanho do kernel que foi conseguido pelo último Teorema. Mais precisamente, $g(k)$ limita a quantidade de vetores (de tamanho $k+1$) de números naturais não comparáveis, sendo que cada elemento destes varia de 0 a $k+1$, pela regra 3. Logo, temos, no máximo, $(k+2)^{k+1}$ vetores deste tipo. Com o Teorema 2 e a argumentação anterior, chegamos no seguinte resultado:

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

2.3 Refinamentos

Observa-se que uma solução para o FLP pode ser vista como uma partição no conjunto de clientes, onde clientes em um mesmo grupo são atendidos pela mesma instalação. Podemos obter esses grupos polinomialmente dados o conjunto F' de instalações abertas e as conexões ativas entre instalações e clientes. O contrário também é facilmente obtido. Com isso, obtemos o seguinte resultado:

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.*

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. Para cada partição, assumimos que é servida por uma única instalação, então calculamos o seu custo em tempo polinomial em $\mathcal{O}(\text{poli}(g(k)))$. □

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. □

Até agora, obtivemos kernels não muito pequenos que podem não ser úteis na prática. Adiante, tentaremos responder se existem kernels menores do que os que obtemos e se há alternativa de algoritmos parametrizados além da força bruta no kernel obtido.

2.4 Algoritmo usando programação dinâmica

A ideia de programação dinâmica em subconjuntos melhora o tempo de execução.

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

Demonstração. Começamos pré-processando o vetor OS (de tamanho $2^n \leq 2^k$), fixando um subconjunto $X \subseteq C$ de clientes atendidos por uma mesma instalação. A célula $OS(X)$ possui o custo de atender todos os clientes em X e abrir a instalação que resulta no custo mínimo. Então, $OS(X) \coloneqq \min_{f \in F}(F_X(f))$, onde $F_X(f)$ é o custo de abrir f e servir X através dela. Com isso, podemos calcular o custo $s(X)$ de servir um certo grupo de clientes através de algumas instalações combinando dois subconjuntos, com o uso de programação dinâmica:

$$s(X) \coloneqq \min_{\emptyset \subseteq Y \subseteq X}(OS(Y) + s(X \backslash Y))$$

Tal fórmula resulta em $\mathcal{O}(3^n) \leq \mathcal{O}(3^k)$ operações. Cada um dos clientes pertence a Y, $X \backslash Y$ ou $C \backslash X$. Note que quando $|X| = 1$, então $Y = X$ e $s(X) = OS(X) + s(\emptyset) = OS(X)$. Pode acontecer de existirem clientes atendidos pela mesma instalação nos conjuntos Y e $X \backslash Y$, seja Y' estes clientes. Mesmo com essa possibilidade, ainda vale a fórmula apresentada, pois:

$$s(X) \leq OS(Y \cup Y') + s(X \backslash (Y \cup Y')) \leq OS(Y) + s(X \backslash Y)$$

O tempo de execução do algoritmo se dá pelo cálculo de cada $OS(X)$ em tempo $\mathcal{O}(|X||F|) \subseteq \mathcal{O}(2^k m)$ e do custo da fórmula da programação dinâmica em si, o que resulta em um algoritmo $\mathcal{O}(2^k m + 3^k)$. Abaixo temos o algoritmo:

Algoritmo 2

se $|C| > k$ **então** retorne não
fim se
$s(\emptyset) \coloneqq 0$
para cada $X \subseteq C$ **faça**
    calcule $OS(X)$ em tempo $\mathcal{O}(|X||F|)$
fim para
para $i \coloneqq 1 \ldots |C|$ **faça**
    para cada $X \subseteq C$ com $|X| = i$ **faça**
        $s(X) \coloneqq \min_{\emptyset \subseteq Y \subseteq X}(OS(Y) + s(X \backslash Y))$
    fim para
fim para

□

Aplicando-se o Lema 6 antes de executar a programação dinâmica, obtemos um algoritmo que possui melhor tempo de execução, dado por $\mathcal{O}^*(2^k(k+1)^{k+2}) \subseteq \mathcal{O}^*((2k)^k)$.

3 Conclusões

Este trabalho foi o responsável por iniciar o estudo parametrizado do FLP e fornecer alguns pontos importantes sobre o problema. Até hoje, não foi feito outro trabalho parametrizado para o FLP, portanto, temos possibilidades que podem ser exploradas:

- Ainda está em aberto se é possível dar melhores algoritmos e kernels menores para este problema e variantes;

- Talvez existam outros parâmetros que se mostrem mais adequados para serem aplicados em casos específicos do problema, como o que admite clientes não atendidos (*outliers*) ou um problema que já possua um certo número de instalações abertas;
- No caso geral, nem toda instalação estará perto de todos os clientes, então pode-se usar este fato para diminuir ainda mais a complexidade dos algoritmos dando uma noção de vizinhança ou restrição de grau à clientes.

4 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

[1] M. R. Fellows and H. Fernau. Facility location problems: A parameterized view. *Discrete Applied Mathematics*, 159(11):1118–1130, 2011.

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

[3] S. Li. A 1.488 Approximation Algorithm for the Uncapacitated Facility Location Problem. In L. Aceto, M. Henzinger, and J. Sgall, editors, *Automata, Languages and Programming*, volume 6756 of *Lecture Notes in Computer Science*, pages 77–88. Springer Berlin Heidelberg, 2011.