Uma Aproximação Ótima para o Problema do Caixeiro Alugador*

Lehilton L. C. Pedrosa[1], **Rafael C. S. Schouery**[1]

[1]Instituto de Computação – Universidade Estadual de Campinas (UNICAMP)
Av. Albert Einstein, 1251 – 13083-852 – Campinas – SP – Brasil

{lehilton,rafael}@ic.unicamp.br

Abstract. *In the classical Traveling Salesman Problem (TSP), we want to find a route that visits each of n cities, minimizing the total traveled distance. An often considered generalization is the Traveling Car Renter Problem, in which the route is traveled by renting a set of cars. Every car has a distinct distance function, but each rented car incurs the payment of a return fee $g \geq 0$. The objective is to minimize the sum of traveled distances plus return fees. In this work, we present a $O(\log n)$-approximation for the problem. This algorithm is asymptotically optimal, since we show that there is no approximation with factor $o(\log n)$, unless P = NP.*

Resumo. *No clássico Problema do Caixeiro Viajante (TSP), queremos encontrar uma rota que passa por um conjunto de n cidades e que minimize a distância total percorrida. Uma generalização conhecida é o chamada Problema do Caixeiro Alugador, em que a rota pode ser percorrida alugando-se um subconjunto de veículos. Cada veículo tem uma função de distância distinta, mas para cada veículo alugado, paga-se uma taxa de retorno $g \geq 0$. O objetivo é minimizar a soma das distâncias percorridas mais as taxas de retorno. Neste trabalho, apresentamos uma $O(\log n)$-aproximação para o problema. Esse algoritmo é assintoticamente ótimo já que também mostramos que não há aproximação com fator $o(\log n)$, a não ser que P = NP.*

1. Introdução

Problemas de transporte aparecem no contexto de logística ou transporte urbano. Particularmente, serviços de transporte individual são oferecidos nas mais diversas formas, como compartilhamento e aluguel de veículos. Na maior parte das vezes, os problemas de aluguel de veículos são discutidos do ponto de vista de uma companhia, que detém uma frota de veículos e cujo objetivo é maximizar o lucro total ou o número de clientes atendidos. Alguns trabalhos consideram também o problema do ponto de vista de um usuário (Goldbarg et al., 2012; da Silva and Ochi, 2016; Rios et al., 2017), que tem um conjunto de opções de aluguel disponíveis e deseja escolher quais serviços contratar para minimizar o seu custo total.

Um problema que visa minimizar os custos de um usuário aparece no setor de turismo (da Silva and Ochi, 2016). Goldbarg et al. (2012) introduziram o *Problema do*

*Este trabalho foi parcialmente financiado pelo processo nº 2015/11937-9, Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP) e pelos processos nºs 425340/2016-3, 308689/2017-8 e 313026/2017-3, Conselho Nacional de Desenvolvimento Científico e Tecnológico.

Caixeiro Alugador (CARS), ilustrado a seguir. Um turista deseja visitar um conjunto de cidades e, para isso, tem um conjunto de veículos disponíveis para alugar. O custo de alugar cada veículo é dado por uma função de distância do trajeto percorrido, além do custo de retornar o veículo até o local alugado. Tanto a função de distância quanto a taxa de retorno dependem do veículo escolhido. O objetivo é alugar veículos para visitar todas as cidades e retornar ao local de origem pagando o mínimo possível.

Seguindo Goldbarg et al. (2012), que desenvolveram algoritmos meméticos, outros trabalhos consideraram o problema (da Silva and Ochi, 2016; Rios et al., 2017; Dias et al., 2016). Observamos que, se exigirmos que a solução seja um ciclo sem repetição de cidades, ou permitirmos funções de custo arbitrárias, então o problema não admite aproximação por qualquer função computável, a não ser que o Problema do Caminho Hamiltoniano possa ser resolvido em tempo polinomial. Assim, consideramos a versão do problema em que as funções são métricas e a taxa de retorno é a mesma para todo veículo e independe do local de devolução. Formalmente, no *Problema do Caixeiro Alugador Uniforme* (UCARS), dado um conjunto V de n cidades, r funções de distância $d_i : V \times V \to \mathbb{R}_{\geq 0}$ e uma taxa de retorno $g \geq 0$, o objetivo é encontrar um conjunto de passeios $P_1, \ldots, P_s$ associados a índices $e_1, e_2, \ldots, e_s$, tais que $(P_1 \ldots P_s)$ é um passeio fechado que passa por todas cidades e que minimiza a soma

$$\sum_{i=0}^{s} \left(g + \sum_{(u,v) \in E(P_i)} d_{e_i}(u, v) \right).$$

Contribuições Neste trabalho, mostramos que UCARS é tão difícil quanto o Problema da Cobertura de Conjuntos (SC), o que implica que UCARS não tem aproximação com razão $o(\log n)$, a não ser que P = NP (Guha and Khuller (1999)). Em seguida, observamos que UCARS pode ser reduzido para o Group-TSP (versão de TSP em que se deseja percorrer um elemento de cada grupo dado), derivando diretamente em uma aproximação de razão $O(\log^2 n \log \log n \log r)$. Finalmente, obtemos uma aproximação melhor, apresentando um algoritmo de arredondamento probabilístico com razão de aproximação $O(\log n)$. Esse algoritmo é assintoticamente ótimo por causa do limite inferior.

2. Inaproximabilidade

No Problema da Cobertura de Conjuntos, dados conjuntos $S_1, S_2, \ldots, S_m$ cuja união é U, queremos encontrar $S_{e_1}, S_{e_2}, \ldots, S_{e_s}$ cuja união é U e s é mínimo.

Teorema 2.1. Se UCARS admite α-aproximação, então SC admite α-aproximação.

Corolário 2.2. Se P $\neq$ NP, então UCARS não admite $(c \log n)$-aproximação com $c < 1$.

Por outro lado, reduzimos UCARS para Group-TSP. Uma instância de Group-TSP é formada por um conjunto V', uma função de distância d' e conjuntos $S_1, S_2, \ldots, S_l \subseteq V'$ e uma solução é uma passeio fechado P tal que $S_i \cap V(P) \neq \emptyset$ para $1 \leq i \leq l$. Construímos uma instância de Group-TSP. Primeiro, criamos um grafo $G = (V', E)$ em que, para cada $v \in V$, adicionamos um grupo $\{v_1, v_2, \ldots, v_r\}$ formando uma clique com arestas de peso g. Depois, para cada $u, v \in V$ e $1 \leq i \leq r$, adicionamos arestas (u_i, v_i) com custo $d_i(u, v)$. Observe que, dada uma solução para a instância de Group-TSP com custo SOL, podemos criar uma solução para a instância de UCARS de custo O(SOL) e vice-versa. Portanto, obtemos uma aproximação para UCARS com fator $O(\log^2 n \log \log n \log r)$ (Garg et al. (2000)). Na seção seguinte, mostramos que há uma $O(\log n)$-aproximação para UCARS.

3. Algoritmo de arredondamento de PL

A ideia principal do algoritmo é resolver o problema em duas fases. Primeiro, obtemos uma cobertura de todos os vértices de V utilizando subárvores. Assim, ao invés de procurar um conjunto de segmentos que forme um passeio fechado, nos preocupamos apenas em encontrar um conjunto de componentes conexas que cubram o conjunto de vértices. Na segunda fase, transformamos a cobertura em um passeio fechado.

3.1. Uma formulação de cobertura

No seguinte programa linear inteiro, denotamos por $\mathcal{S}$ o conjunto de todos os pares (i, S), com $S \subseteq V$ e $1 \leq i \leq r$. Para cada $(i, S) \in \mathcal{S}$, $\text{MST}_i(S)$ denota o custo de uma árvore geradora mínima de S com respeito a d_i e $x_{(i,S)}$ indica se (i, S) faz parte da solução.

$$\begin{array}{lll} \text{minimizar} & \displaystyle\sum_{(i,S)\in\mathcal{S}} x_{(i,S)}(\text{MST}_i(S) + g) & \\ \text{sujeito a} & \sum_{(i,S)\in\mathcal{S}: v\in V} x_{(i,S)} \geq 1 & \forall\ v \in V, \\ & x_{(i,S)} \in \{0, 1\} & \forall\ (i, S) \in \mathcal{S}. \end{array} \tag{PLI}$$

Observe que uma solução de uma instância de UCARS induz uma solução para esse programa, então o valor ótimo do (PLI) é um limitante inferior para o valor ótimo dessa instância. Seja (PL) a relaxação linear de (PLI). Observe que (PL) tem um número exponencial de variáveis e, portanto, não sabemos resolver esse problema diretamente. No entanto, podemos obter uma solução aproximada com um número polinomial de variáveis não nulas, conforme o lema a seguir, que é um resultado central para o algoritmo. Para um programa linear (Q), denote por $\text{OPT}(Q)$ o valor ótimo de (Q).

Lema 3.1. É possível, em tempo polinomial, encontrar uma solução viável para (PL) de valor $O(\text{OPT}(\text{PLI}))$.

A prova desse lema pode ser resumida do seguinte modo. Obtemos a formulação dual de (PL), que tem restrições $\sum_{v\in V} y_v - \text{MST}_i(S) \leq g$ para cada $(i, S) \in \mathcal{S}$ e executamos o método dos elipsoides. Para cada solução candidata y dada pelo método, queremos resolver o problema da separação correspondente. Se pudéssemos encontrar (i, S) que maximiza o valor do lado esquerdo, então poderíamos ou encontrar uma restrição violada, ou concluir que y é viável. Infelizmente, esse problema é NP-difícil e sequer tem aproximação constante. Para contornar essa dificuldade, utilizamos uma estratégia mais elaborada. Observe que se $\text{MST}_i(S)$ for pequeno quando comparado com g, então maximizar $\sum_{v\in V} y_v$ é uma boa aproximação para esse valor e esse problema pode ser bem resolvido a menos de um fator constante. Assim, a principal observação para derivar o lema é que o valor de PLI quando restrito a componentes com $\text{MST}_i(S) \leq g$ está a um múltiplo constante de $\text{OPT}(\text{PLI})$.

3.2. Arredondando uma solução fracionária

Dada uma solução x de (PL) de valor $O(\text{OPT}(\text{PLI}))$, o algoritmo obtém uma solução para a instância de UCARS arredondando o valor das variáveis $x_{(i,S)}$ para cada $(i, S) \in \mathcal{S}$. Observe que, como podemos supor que $0 \leq x_{(i,S)} \leq 1$, podemos incluir (i, S) em uma solução integral com probabilidade $x_{(i,S)}$. Fazendo isso para todas as componentes, o custo esperado é igual $O(\text{OPT}(\text{PLI}))$. A ideia do algoritmo é que, se repetirmos o procedimento acima um número suficiente de vezes, então a probabilidade de um elemento continuar descoberto tende a zero. Todos os passos são descritos em Algoritmo 1.

1. Obtenha uma solução x de (PL) usando o Lema 3.1.
2. Faça $\mathcal{C} \leftarrow \emptyset$ e repita $\lceil \log n \rceil$ vezes:
   - Para cada (i, S), inclua (i, S) em $\mathcal{C}$ com probabilidade $x_{(i,S)}$.
3. Para cada $v \in V$ tal que $v \notin S$ para todo $(i, S) \in \mathcal{C}$, adicione $(1, \{v\})$ a $\mathcal{C}$.
4. Faça $F \leftarrow \emptyset$ e para cada $(i, S) \in \mathcal{C}$:
   - Construa uma árvore geradora mínima H de $S \setminus V(F)$ com respeito a d_i.
   - Construa uma rota P duplicando as arestas de H.
   - Rotule cada aresta de H com índice i e adicione H a F.
5. Defina $d_{\min} : V \times V \to \mathbb{R}$ de forma que $d_{\min}(u, v) = \min_{1 \leq i \leq r} d_i(u, v)$.
6. Contraia cada componente conexa de F e obtenha F'.
7. Construa uma árvore geradora mínima T' de F' com respeito a $d_{\min}$.
8. Para cada aresta (u, v) de T', adicione duas arestas paralelas (u, v) a F rotuladas com índice i tal que $d_{\min}(u, v) = d_i(u, v)$.
9. Construa um ciclo euleriano C de F e devolva C.

Algoritmo 1: Aproximação para UCARS.

Nos passos de 1 a 3, obtemos uma cobertura de V por componentes de $\mathcal{C}$. Após o passo 4, F é formado por um conjunto de rotas que cobrem todos os vértices V, mas é possivelmente desconexo. Assim, queremos adicionar arestas para conectar essas rotas. Para cada aresta, escolhemos o veículo de menor custo correspondente e obtemos um conjunto de arestas que conecta F com menor custo. Daí, após o passo 8, F é um grafo conexo em que todo vértice tem grau par e, portanto, tem um ciclo euleriano C. Observe que C pode ser decomposto em uma sequência de passeios maximais de arestas com o mesmo rótulo e, portanto, induz uma solução viável do problema.

O custo da solução devolvida é composto das distâncias percorridas e das taxas de retorno. Observe que ambos custos podem ser limitados pelos custos das componentes em $\mathcal{C}$. Como o custo esperado das componentes sorteadas em cada iteração do passo 2 é de no máximo o valor de x, que é $O(\text{OPT}(\text{PLI}))$, e o passo 2 é executado $O(\log n)$ vezes, obtemos o seguinte teorema.

Teorema 3.2. O Algoritmo 1 é uma $O(\log n)$-aproximação para UCARS.

Referências

da Silva, A. R. V. and Ochi, L. S. (2016). An efficient hybrid algorithm for the traveling car renter problem. *Expert Systems with Applications*, 64:132 – 140.

Dias, S. S., Ochi, L. S., Machado, V. M. C., Simonetti, L. G., and da Silva, A. R. V. (2016). Uma heurística baseada em ils para o problema do caixeiro alugador. In *Anais do XLVIII SBPO Simpósio Brasileiro de Pesquisa Operacional*.

Garg, N., Konjevod, G., and Ravi, R. (2000). A polylogarithmic approximation algorithm for the group steiner tree problem. *Journal of Algorithms*, 37(1):66 – 84.

Goldbarg, M. C., Asconavieta, P. H., and Goldbarg, E. F. G. (2012). Memetic algorithm for the traveling car renter problem: an experimental investigation. *Memetic Computing*, 4(2):89–108.

Guha, S. and Khuller, S. (1999). Greedy Strikes Back: Improved Facility Location Algorithms. *Journal of Algorithms*, 31(1):228–248.

Rios, B. H. O., Goldbarg, E. F. G., and Quesquén, G. Y. O. (2017). A hybrid metaheuristic using a corrected formulation for the traveling car renter salesman problem. In *2017 IEEE Congress on Evolutionary Computation (CEC)*, pages 2308–2314.