Formulações e heurísticas para os problemas *leasing k-median* e *leasing k-center*

J. M. dos Santos[1], G. P. P. Londe[1], A. Ribeiro[1], W. R. da Silva[2]

[1]Pontifícia Universidade Católica de Goiás (PUC - GOIÁS) – Goiânia, GO – Brasil

[2]Instituto de Computação
Universidade Estadual de Campinas (UNICAMP) – Campinas, SP – Brasil

{jorgemenezesad,guilherme.londe2}@gmail.com, alexribeiro@pucgoias.edu.br
welverton.silva@unicamp.br

Abstract. *In this paper, we consider the generalizations of the problems k-median and k-center, known, respectively, as leasing k-median (LKM) and leasing k-center (LKC). We present the formulations of integer linear programming and a heuristic based on the BRKGA meta-heuristic. We compare the solutions found by the heuristic to the solutions found by the solver GUROBI applied to the formulations of linear programming, setting a time limit of 10 minutes. For the tested small instances, the solutions costs found by the heuristic were close to the optimal cost (average GAP* $\leq 8\%$*). For the tested greater instances, the heuristic found better solutions than the solutions found by the solver, at least in half of the tests.*

Resumo. *Neste artigo, consideramos as generalizações dos problemas* k*-median e* k*-center, conhecidas, respectivamente, por leasing* k*-median (LKM) e leasing* k*-center (LKC). Apresentamos formulações de programação linear inteira e uma heurística baseada na meta-heurística BRKGA. Comparamos as soluções geradas pela heurística com as soluções geradas pelo resolvedor GUROBI aplicado às formulações em programação linear, estabelecendo um prazo de 10 minutos de execução para ambos. Para as instâncias pequenas testadas, os custos das soluções heurísticas foram próximos aos custos ótimos (GAP médio* $\leq 8\%$*). Para instâncias maiores testadas a heurística gerou soluções superiores às do resolvedor, em pelo menos metade dos testes.*

1. Introdução

Os problemas *leasing k-median* (LKM) e *leasing k-center* (LKC) foram propostos em [Lintzmayer and Mota 2017] como generalizações dos problemas *k-median* e *k-center*. Nessas generalizações, os clientes precisam ser servidos em instantes de tempo específicos e as facilidades são abertas por um determinado intervalo de tempo, onde clientes e facilidades são pontos no espaço métrico.

Sejam (V, d) um espaço métrico e $T = \{0, \ldots, t_{max}\}$ o conjunto dos instantes de tempo, onde t_{max} é o instante máximo considerado. Os clientes que precisam ser servidos no instante t, para t em T, são agrupados no subconjunto D_t de V. Seja, portanto, $\mathcal{D} = \{D_1, \ldots, D_{t_{max}}\}$ o conjunto de todos os clientes. Sejam $L = \{1, \ldots, l_{max}\}$ o conjunto dos tipos de facilidades, onde l_{max} é o último tipo considerado, e $\delta_1, \ldots, \delta_{l_{max}}$

as durações possíveis para a abertura de uma facilidade, de modo que uma facilidade do tipo l aberta no instante t permanece ativa durante o intervalo $[t, t + \delta_l)$.

O objetivo do problema LKM é encontrar um conjunto $\mathcal{M} \subseteq V \times L \times T$ tal que, para todo t em T, vale que $|\{(i, l, t') \in \mathcal{M} : t \in [t', t' + \delta_l)\}| \leq k$, e que minimiza a soma das distâncias de cada cliente à facilidade ativa mais próxima, dada por

$$\sum_{t \in T} \sum_{j \in D_t} \min_{\substack{(i,l,t') \in \mathcal{M} \\ t \in [t', t' + \delta_l)}} d(i, j).$$

O problema LKC se diferencia do LKM apenas na função objetivo, que consiste em encontrar um conjunto $\mathcal{M}$ que minimiza a distância máxima entre um cliente e a facilidade ativa mais próxima, dada por

$$\max_{t \in T} \max_{j \in D_t} \min_{\substack{(i,l,t') \in \mathcal{M} \\ t \in [t', t' + \delta_l)}} d(i, j).$$

Os problemas *k-median* e *k-center* são NP-difíceis [Kariv and Hakimi 1979a, Kariv and Hakimi 1979b]. Estes problemas são, respectivamente, os casos particulares do LKM e do LKC, quando $|T| = 1$. Portanto os problemas LKM e LKC também são NP-difíceis.

2. Formulações

A formulação em programação linear inteira do LKM é uma adaptação da formulação de [Ahn et al. 1988] para o *k-median*.

Na formulação a seguir, a variável x_{ij}^t indica se a facilidade i atende o cliente j no instante t. A variável y_{il}^t indica se a facilidade i do tipo l é aberta no instante t. Considere $p_l^t = t - \delta_l + 1$, se $\delta_l \leq t$; ou $p_l^t = 1$, caso contrário.

$$\text{minimizar} \sum_{t \in T} \sum_{i \in V} \sum_{j \in D_t} x_{ij}^t d(i, j) \tag{1}$$

$$\text{sujeito a:} \sum_{i \in V} x_{ij}^t = 1 \qquad \forall t \in T,\ \forall j \in D_t \tag{2}$$

$$\sum_{i \in V} \sum_{l \in L} \sum_{t' = p_l^t}^{t} y_{il}^{t'} \leq k \qquad \forall t \in T \tag{3}$$

$$\sum_{l \in L} \sum_{t' = p_l^t}^{t} y_{il}^{t'} \geq x_{ij}^t \qquad \forall t \in T,\ \forall i \in V,\ \forall j \in D_t \tag{4}$$

$$x_{ij}^t \in \{0, 1\} \qquad \forall t \in T,\ \forall i \in V,\ \forall j \in D_t \tag{5}$$

$$y_{il}^t \in \{0, 1\} \qquad \forall t \in T,\ \forall i \in V,\ \forall l \in L \tag{6}$$

A função objetivo (1) minimiza a soma das distâncias de cada cliente à facilidade ativa mais próxima, enquanto que as restrições em (2) asseguram para cada instante t que cada cliente j em D_t seja servido por uma única facilidade i. As restrições em (3) garantem para cada instante t que no máximo k facilidades estejam ativas. As restrições em (4) asseguram que um cliente j em D_t possa ser servido pela facilidade i em V, apenas se i estiver ativa no instante t.

A formulação de programação linear inteira mista do problema LKC é uma adaptação da formulação de [Daskin 1995] para o *k-center*. Essa formulação introduz

uma variável z, que representa a distância máxima entre um cliente ativo e uma facilidade ativa mais próxima.

$$\text{minimizar } z \tag{7}$$

$$\text{sujeito a: } \sum_{i \in V} x_{ij}^t d(i,j) \leq z \qquad \forall t \in T,\ \forall j \in D_t \tag{8}$$

$$(2), (3), (4), (5), (6)$$

$$z \in \mathbb{R}$$

A função objetivo (7) minimiza a distância máxima entre cada cliente e a facilidade ativa mais próxima. As restrições em (8) asseguram para cada instante t que a maior distância entre um cliente j em D_t e a facilidade ativa mais próxima seja no máximo z.

3. Heurística

Nesta seção é proposta uma heurística para os problemas LKM e LKC baseada no *Biased Random-Key Genetic Algorithm* (BRKGA), proposto por [Gonçalves and Resende 2010]. Nesta meta-heurística as soluções de um problema de otimização combinatória são obtidas a partir de um vetor de chaves aleatórias (cromossomo) pertencentes ao intervalo $[0, 1)$, por meio de um decodificador.

Para ambos os problemas, um cromossomo é codificado como uma matriz de chaves aleatórias A de dimensões $|T| \times |V|$, onde o elemento a_{ti} corresponde à possibilidade de abrir a facilidade i no instante t.

Seja $\delta_\mu = \frac{1}{l_{max}} \sum_{l \in L} \delta_l$ a duração média de uma facilidade. Seja v a probabilidade de o decodificador abrir cada uma das facilidades do cromossomo A, dado por

$$v = \frac{\frac{|T|k}{\delta_\mu}}{|T||V|} = \frac{k}{\delta_\mu |V|}.$$

O decodificador recebe um cromossomo A e retorna o custo da solução. O algoritmo percorre A em ordem prioritária de linha (*row-major order*), e caso o elemento a_{ti} seja menor que v e a abertura da facilidade i de tipo $l = 1 + \lfloor \frac{a_{ti} * l_{max}}{v} \rfloor$ não ultrapasse o limite de k facilidades ativas no instante t, então a tupla (i, l, t) é adicionada à solução $\mathcal{M}$. Ao final da análise do instante t, caso hajam clientes em D_t e nenhuma facilidade ativa neste instante, a tupla (i^t_{min}, l_{min}, t) é adicionada à $\mathcal{M}$, onde $l_{min} = arg \min_{l \in L} \delta_l$ e $i^t_{min} = arg \min_{i \in V} \sum_{j \in D_t} d(i,j)$ para o LKM, ou $i^t_{min} = arg \min_{i \in V} \max_{j \in D_t} d(i,j)$ para o LKC.

4. Experimentos computacionais

Os algoritmos foram implementados na linguagem C++ com o compilador GCC, versão 5.4.0. Para as formulações de programação linear foi utilizado o resolvedor comercial GUROBI, versão 8.1.0. Para a heurística foi utilizado o $framework$ do BRKGA proposto por [Toso and Resende 2015] e o decodificador foi implementado conforme a Seção 3.

Os testes foram realizados em um computador de 4GB de memória RAM e processador pentium dual core 2,60 GHz sob um sistema operacional Linux. Foi estabelecido um tempo limite de 10 minutos para a execução de cada teste, e sendo assim foi considerada apenas a melhor solução que cada abordagem obteve nesse tempo.

Um conjunto de instâncias foi gerado aleatoriamente para testar ambos os problemas. Essas instâncias foram divididas em três categorias. A primeira categoria reune as

instâncias em que o GUROBI encontrou a solução ótima dentro do tempo limite. A segunda categoria reune as instâncias em que o GUROBI não encontrou a solução ótima no tempo limite. A terceira categoria reune instâncias em que a execução pelo GUROBI não foi possível devido ao elevado número de restrições e de variáveis ($|V| > 200$, $|T| > 30$ e $|L| > 20$), e por isso não são consideradas nas análises dos experimentos. A Tabela 1 e a Tabela 2 apresentam os resultados encontrados.

Tabela 1. Resultados para o LKM

Instâncias da categoria 1			Instâncias da categoria 2		
Instância	Gurobi	BRKGA	Instância	Gurobi	BRKGA
1	23168	23821	11	216618	236478
2	27953	28412	12	225868	291786
3	3025	3025	13	476682	401752
4	4801	4801	14	352198	301892
5	6785	6785	15	345736	233866
6	28087	28242	16	348736	316436
7	9497	9902	17	184862	185082
8	6568	6568	18	187872	164238
9	38773	38808	19	55221	55158
10	19147	19709	20	39258	40872

Tabela 2. Resultados para o LKC

Instâncias da categoria 1			Instâncias da categoria 2		
Instância	Gurobi	BRKGA	Instância	Gurobi	BRKGA
1	116	133	11	442	248
2	127	133	12	238	220
3	111	121	13	408	256
4	112	112	14	392	260
5	115	124	15	460	236
6	98	102	16	380	276
7	96	102	17	300	240
8	95	102	18	378	252
9	98	109	19	97	77
10	98	111	20	97	70

O GAP médio para a primeira categoria de instâncias foi de $1,19\%$ para o LKM e de $7,13\%$ para o LKC. Para a segunda categoria de instâncias, 50% das soluções encontradas pelo BRKGA foram melhores do que as fornecidas pelo GUROBI para o LKM, e 100% das soluções do BRKGA foram melhores do que as do GUROBI para o LKC.

Para instâncias da terceira categoria, o BRKGA pode se tornar a única opção dentre as duas apresentadas, haja vista que o resolvedor exigiu recursos não suportados pela máquina de teste.

É possível observar que para instâncias pequenas, em que o GUROBI encontra uma solução ótima em um curto prazo, o BRKGA é capaz de prover soluções com custo próximo ao custo ótimo (GAP médio $\leq 8\%$). Para instâncias maiores, o BRKGA pode encontrar soluções melhores do que as soluções do GUROBI, quando fixa-se um tempo de execução. Com base nessas observações, o uso do BRKGA sobre os problemas LKM e LKC mostra-se interessante em relação ao uso de um método exato como o GUROBI.

Referências

Ahn, S., Cooper, C., Cornuéjols, G., and Frieze, A. (1988). Probabilistic Analysis of a Relaxation for the k-Median Problem. *Mathematics of Operations Research*.

Daskin, M. S. (1995). *Network and Discrete Locations: Models, algorithms, and applications*. John Wiley & Sons.

Gonçalves, J. F. and Resende, M. G. C. (2010). Biased random-key genetic algorithms for combinatorial optimization. *Journal of Heuristics*.

Kariv, O. and Hakimi, S. L. (1979a). An Algorithmic Approach to Network Location Problems. I: The p-Centers. *SIAM Journal on Applied Mathematics*.

Kariv, O. and Hakimi, S. L. (1979b). An Algorithmic Approach to Network Location Problems. II: The p-Medians. *SIAM Journal on Applied Mathematics*.

Lintzmayer, C. N. and Mota, G. O. (2017). Caderno de problemas. In *1º Workshop Paulista em Otimização, Combinatória e Algoritmos*.

Toso, R. and Resende, M. (2015). A C++ application programming interface for biased random-key genetic algorithms. *Optimization Methods and Software*.