LIII Simpósio Brasileiro de Pesquisa Operacional
João Pessoa - PB, 3 a 5 de novembro de 2021

UMA GENERALIZAÇÃO DO PROBLEMA DO CAIXEIRO VIAJANTE COM COBERTURA

Davi Rodrigues
Instituto de Computação - Universidade Estadual de Campinas
Av. Albert Einstein, 1251 - Cidade Universitária, Campinas - SP
davirdgs@gmail.com

Prof. Dr. Fábio Luiz Usberti
Instituto de Computação - Universidade Estadual de Campinas
Av. Albert Einstein, 1251 - Cidade Universitária, Campinas - SP
fusberti@gmail.com

Prof. Dr. Celso Cavellucci
Instituto de Computação - Universidade Estadual de Campinas
Av. Albert Einstein, 1251 - Cidade Universitária, Campinas - SP
ccavellucci@gmail.com

RESUMO

O problema do caixeiro viajante (travelling salesman problem - TSP) tem por objetivo encontrar um ciclo hamiltoniano de custo mínimo em um grafo não-direcionado completo $G = (V, E)$. Uma variação do TSP, o problema do caixeiro viajante com cobertura (Covering Salesman Problem - CSP), visa obter um ciclo de custo mínimo que cobre todos os nós do grafo G, onde a cada vértice v é associado um conjunto de cobertura. Este trabalho apresenta uma generalização do CSP, o problema das p-medianas com cobertura (Covering Hamiltonian p-Medians Problem - CHpMP), cujo objetivo é encontrar em G um subgrafo de custo mínimo composto por p ciclos hamiltonianos disjuntos que cubram todos os nós de G. Uma formulação para este novo problema é apresentada, assim como ensaios computacionais para avaliar a solução exata do CHpMP.

ABSTRACT

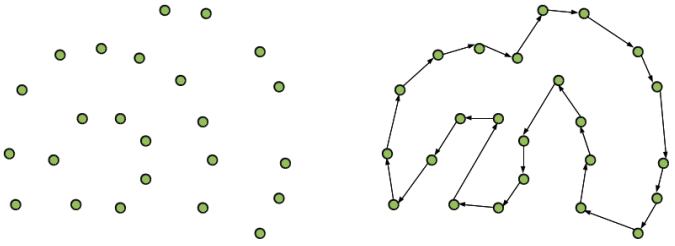
The Traveling Salesman Problem (TSP) aims to find a minimal cost Hamiltonian cycle in a complete non-directed graph $G = (V, E)$. A variation of the TSP, the Covering Salesman Problem (CSP), searches for the minimum cost cycle that covers all nodes of the graph G, where each vertex v is associated with a set $C(v)$, containing all nodes that cover v. This work presents a generalization of the CSP, the problem of the Covering Hamiltonian p-Medians Problem (CHpMP), whose objective is to find in G a subgraph of minimum cost composed of disjoint p Hamilton cycles that cover all the G nodes . A formulation for this new problem is presented, as well as computational tests to evaluate the exact solution of CHpMP.

KEYWORDS. Combinatorial Optimization, Travelling Salesman Problem, Integer Linear Programming

1. Introdução

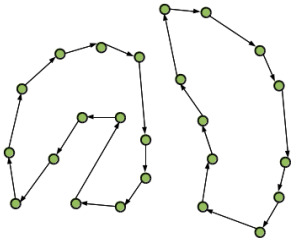
O TSP é um problema NP-difícil [Applegate, 2007] que consiste na busca de um ciclo hamiltoniano de menor custo em um grafo completo não-direcionado G. Nesse problema, o espaço de busca aumenta de maneira fatorial em relação ao número de nós, tornando a busca exaustiva por uma solução exata impraticável para instâncias suficientemente grandes, uma vez que um grafo com n nós possui $\frac{(n-1)!}{2}$ ciclos hamiltonianos [Agarwal, 2009].

O problema das p-medianas hamiltonianas (em inglês *hamiltonian p-median problem* - HpMP) é uma generalização do problema do caixeiro viajante (em inglês *travelling salesman problem* - TSP), onde é dado um grafo completo não-direcionado G, e busca-se um subgrafo hamiltoniano formado por p ciclos disjuntos. Portanto, o TSP é um caso particular do HpMP quando $p = 1$, o que implica no HpMP também ser NP-difícil [Garey, 1990]. A Figura 1(a) exemplifica uma instância do HpMP e possíveis soluções para $p = 1$, $p = 2$ e $p = 3$.

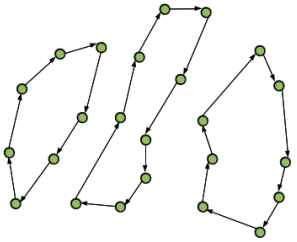


(a) Instância.

(b) Solução com um ciclo.



(c) Solução com dois ciclos.



(d) Solução com três ciclos.

Figura 1: Instância e soluções do HpMP.

Assim como o TSP, o HPMP também pode ser usado como um modelo para muitos problemas práticos, envolvendo cadeia de suprimentos, serviços de entregas, localização de centros de distribuição, serviços de mobilidade urbana, correios [Hongtao Lei, 2016; Groër C, 2008; Applegate, 2007].

Outra generalização do TSP consiste no problema do caixeiro viajante com cobertura (Covering Salesman Problem - CSP), proposto por Current e Schilling [1989]. Seja um grafo não direcionado $G(V, E)$, onde V é o conjunto de vértices, E o conjunto de arestas, e para cada aresta $e \in E$ é associado um custo não-negativo c_e. Para cada vértice $v \in V$, $C(v)$ é o conjunto de vértices que cobrem v e $D(v)$ é o conjunto de vértices que são cobertos por v, sendo que $v \in C(v)$ e $v \in D(V), \forall v \in V$. Uma solução ótima para o CSP é um ciclo hamiltoniano de custo mínimo sobre um subconjunto dos vértices que cobre todos os vértices de G.

Maziero [2016] apresenta um modelo em programação linear inteira para resolver o CSP, além da implementação de uma metaheurística GRASP para o problema. A cobertura dos nós é

definida pela distância euclidiana entre eles, sendo controlada por um parametro k. Neste caso, para um determinado nó i, todo nó com distância de i menor do que k será considerado coberto por i.

Outra possível definição de cobertura adotada neste trabalho é que, para cada vértice v, considera-se que os k nós com menor distância euclidiana de v, v incluso, são cobertos por ele.

No Problema das p-Medianas Hamiltonianas com Cobertura (Covering Hamiltonian p-Medians Problem CHpMP), dado um grafo completo não-direcionado G, busca-se um subgrafo formado por exatamente p ciclos disjuntos, onde todos os vértices de G estão cobertos por pelo menos um vértice pertencente a um dos ciclos. Neste caso, o CSP é um caso particular do CHpMP quando $p = 1$.

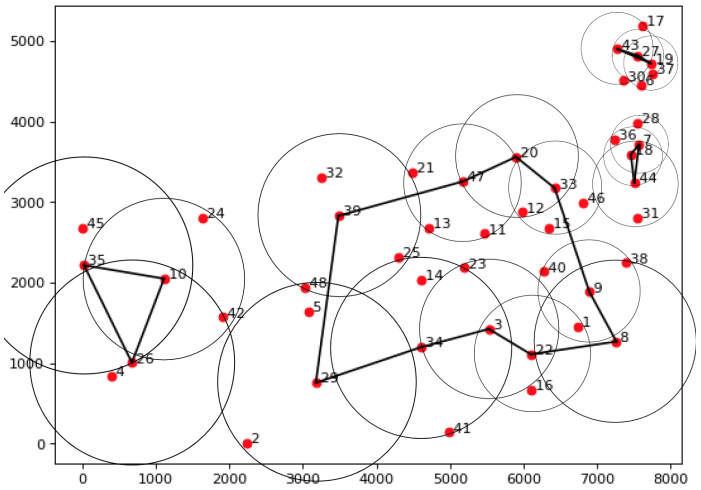


Figura 2: Exemplo de solução para o CHpMP

Contribuições. Neste trabalho introduzimos um novo problema, generalizando o CSP pela possibilidade de existir mais de um ciclo na solução final. Assim sendo, uma solução ótima para o problema das p-medianas hamiltonianas com cobertura (CHpMP) é um conjunto de p ciclos que cobre todos os vértices de G e a soma dos custos das arestas dos ciclos seja mínimo.

1.1. Conceitos preliminares

Um *grafo direcionado* $G = (V, E)$ consiste em um conjunto de vértices (nós) V, um conjunto de arestas (arcos) E, e uma função $(s, t) : E \to V$, onde $s(e)$ é a fonte e $t(e)$ é o destino da aresta direcionada e. Um grafo é dito *não-direcionado* quando, ao invés da função $(s, t) : E \to V$, existe uma função $w : E \to V^2$ que associa cada aresta a um par de vértices, sendo estes os *terminais* da aresta. Um *grafo com pesos* possui uma função adicional $C : E \to \mathbb{R}$ que associa um valor (custo) a cada aresta.

O *grau* de um vértice é definido como o número de arestas incidentes ao mesmo vértice. Um *laço* é definido como uma aresta cujas duas extremidades incidem no mesmo vértice. Caso haja um laço em um vértice, são consideradas as duas extremidades no cálculo do grau.

Um *passeio* é uma sequência $P =< v_0, a_1, v_1, a_2, ..., v_{k-1}, a_k, v_k >$ onde $v_0, v_1, ..., v_k$ são vértices e $a_1, ..., a_k$ são arestas e, para cada i, a_i é um arco tal que $s(a_i) = v_{i-1}$ e $t(a_i) = v_i$. Um *caminho* é um passeio onde não existem vértices repetidos. Um passeio é dito um *ciclo* quando

$v_0 = v_k$. Um caminho em G é dito *hamiltoniano* se contém exatamente uma cópia de cada vértice de G. O *custo de um caminho* ou ciclo é definido como $\sum_{i=1}^{i=k} c(a_i)$, onde $c(a_i)$ é o custo associado à aresta a_i.

2. Modelo de programação linear inteira

Nesta seção será apresentada uma formulação em programação linear inteira para o CHpMP. São dadas as seguintes variáveis de decisão:

$$x_{i,j} = \begin{cases} 1, & \text{se aresta } (i,j) \text{ pertence à solução.} \\ 0, & \text{caso contrário.} \end{cases} \tag{1}$$

$$y_i^k = \begin{cases} 1, & \text{se nó } i \text{ pertence ao ciclo } k. \\ 0, & \text{caso contrário.} \end{cases} \tag{2}$$

Função objetivo:

$$\text{Min} \sum_{(i,j)\in E} c_{i,j} x_{i,j} \tag{3}$$

Dados a quantidade p de ciclos desejados na solução, e os conjuntos $C(v)$ para cada vértice v, o modelo é sujeito às seguintes restrições:

$$\sum_{\delta(i)} x_{i,j} = 2 \sum_{k}^{p} y_i^k \;\; \forall \;\; i \in V \tag{4}$$

A restrição 4 garante que todo vértice terá grau 2 dentro de um determinado ciclo, onde $\delta(i)$ são as arestas no corte do vértice i.

$$\sum_{k}^{p} y_i^k \leq 1 \;\; \forall i \in V \tag{5}$$

A restrição 5 garante que cada vértice i pertença a no máximo um ciclo.

$$\sum_{k}^{p} \sum_{C(i)} y_i^k \geq 1 \;\; \forall i \in V \tag{6}$$

A restrição 6 garante que sempre haverá um nó que cobre i na solução, para todo $i \in V$.

$$y_i^k + x_{i,j} \leq 1 + y_j^k \quad (i,j) \in E, \forall k \in \{1, ..., p\} \tag{7}$$

$$y_j^k + x_{i,j} \leq 1 + y_i^k, \quad (i,j) \in E, \forall k \in \{1, ..., p\} \tag{8}$$

As restrições (7) e (8) garantem que, se dois vértices i e j estão conectados, eles pertencem ao mesmo ciclo.

$$\sum_{i} y_i^k \geq 3 \;\; \forall k \in p \tag{9}$$

A equação 9 garante que haverá pelo menos 3 vértices em cada ciclo.

$$\sum_{(i,j)\in\delta(W)} x_{i,j} \geq 2(y_i^k + y_j^k - 1), \qquad \forall W \subset V,\ \forall i \in W,\ \forall j \in V \setminus W \tag{10}$$

A restrição (10) garante que não haverá formação de sub-ciclos ilegais. Se i e j pertencem ao mesmo ciclo k, o lado direito da restrição será 2, logo deve haver pelo menos duas arestas em qualquer corte de (i, j).

2.1. Algoritmo branch-and-cut

Nesta seção será apresentado um algoritmo branch-and-cut para o CHpMP. Para cada par de vértices do grafo induzido pela solução fracionária, mapeamos os valores dos cortes mínimos. Note que o menor valor do lado esquerdo da restrição 10 corresponde ao corte mínimo entre i e j, sendo assim, ao se obter o corte mínimo para todos os pares de vértices, busca-se minimizar o lado esquerdo da restrição (10) com o objetivo de encontrar alguma violação.

Foi utilizado o algoritmo de Gomory-Hu [Gomory e Hu, 1961] para o cálculo dos cortes mínimos. O algoritmo de Gomory-Hu recebe como entrada um grafo não-direcionado com pesos nas arestas e retorna uma árvore de cortes mínimos. Cada nó i do grafo original é mapeado em um nó i^* da árvore de Gomory-Hu e o corte mínimo entre um determinado par de nós (i, j) no grafo pode ser obtido pelo peso da aresta de menor valor no caminho dos pares equivalentes (i^*, j^*) na árvore (Figura 3).

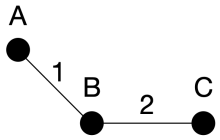


Figura 3: Árvore de Gomory-Hu. Os cortes mínimos que separam os nós 'A' e 'B', 'A' e 'C', e 'B' e 'C' possuem valores 1, 2 e 2, respectivamente.

Para reduzir o espaço de busca, ao invés de explorar os cortes para todos os pares de nós, considera-se apenas os pares de nós adjacentes na árvore de Gomory-Hu, restringindo assim a busca somente às arestas da árvore de Gomory-Hu.

Para encontrar uma restrição (10) violada, para cada aresta da árvore de Gomory-Hu que está associada a um corte mínimo $\delta(W^*)$, deseja-se maximizar o lado direito da restrição (10). Para tal, busca-se por dois vértices i e j, $i \in W^*$ e $j \in V \setminus W^*$, cuja soma $y_i^k + y_j^k$ seja máxima para um determinado k. Com este par de vértices e o corte mínimo obtido pela árvore de Gomory-Hu, se verifica a violação da restrição (10) correspondente e, em caso afirmativo, essa restrição é

incorporada ao modelo. Um pseudo-código dessa separação é apresentado no Algoritmo (1).

Algoritmo 1: Algoritmo de separação das desigualdades (10)

```
início
    Calcular árvore de Gomory-Hu
    para cada ciclo k, k ∈ {1, ..., p} faça
        para cada aresta (u, v) da árvore de Gomory-Hu faça
            Encontrar vértice i do lado esquerdo do corte que maximiza y_i^k
            Encontrar vértice j do lado direito do corte que maximiza y_j^k
            se peso(u, v) < 2(y_i^k + y_j^k − 1) então
                Adiciona restrição (10) correspondente
```

É possível generalizar a restrição (10) para considerar múltiplos ciclos em cada corte, conforme mostra a desigualdade (11) a seguir:

$$\sum_{(i,j)\in\delta(W)} X_{i,j} \geq 2\sum_{k\in\kappa} Y^k_{I(k)} + 2\sum_{k\in\kappa} Y^k_{J(k)} - 2|\kappa| \qquad \forall W \subset V, \forall \kappa \subset \{1, ..., p\} \qquad (11)$$

Onde $I(k) : \kappa \to W$ e $J(k) : \kappa \to W \setminus V$. Para cada par (i, j) de vértices, soma-se os valores parciais de cada componente y_i^k e y_j^k para todos os subciclos k. O conjunto κ é o subconjunto de ciclos cujo valor das componentes parciais y_i^k ou y_j^k é não-nulo.

O Algoritmo (2) mostra como realizar a separação da nova família de desigualdades (11). Com esta separação, agrega-se em uma única restrição todos os pares de vértices que estão em um mesmo ciclo mas que se encontram em lados opostos do corte.

Algoritmo 2: Algoritmo de separação das desigualdades (11).

```
início
    Pré-calcular árvore de Gomory-Hu
    para cada aresta (u, v) da árvore de Gomory-Hu faça
        Y_i ← 0
        Y_j ← 0
        κ ← 0
        para cada ciclo k, k ∈ {1, ..., p} faça
            Encontrar vértice i do lado esquerdo do corte que maximiza y_i^k
            Encontrar vértice j do lado direito do corte que maximiza y_j^k
            se y_i^k + y_j^k > 1 então
                Y_i ← Y_i + y_i^k
                Y_j ← Y_j + y_j^k
                κ ← κ + 1
        se peso(u, v) < 2(Y_i + Y_j − κ) então
            Adiciona restrição (11) correspondente
```

3. Meta-heurística para busca de solução inicial

Nota-se que uma solução para o HpMP também é uma solução para o CHpMP pois, como todos os vértices pertencem à solução, todos os vértices estão cobertos. Uma heurística apresentada por Erdogan et al Günes Erdogan [2016] para gerar soluções de partida para o HpMP é o *Giant Tour Heuristic* (GTH).

A partir de um ciclo hamiltoniano, busca-se particioná-lo de maneira ótima em p sub-ciclos disjuntos através da seguinte formulação de programação dinâmica:

$$g_k(i) = \begin{cases} min\{c_{i,j} + g_{k+1}(j+1), j \in \{i+2, ..., n-3(p-k)\}\}, & k < p, 3k-2 \leq i \leq \\ & n-3(p-k)-2 \\ c_{in} & , k = p, 3k-2 \leq i \leq n-2 \\ \inf & , c.c. \end{cases} \tag{12}$$

Seja $g_k(i)$ a soma dos custos ótimos dos ciclos $k, k+1, ..., p$, onde o *k*-ésimo ciclo inicia no *i*-ésimo elemento, e $c_{i,j}$ o custo do ciclo que inicia no *i*-ésimo elemento e termina no *j*-ésimo elemento, onde a ordem dos vértices é dada pela ordem em que eles aparecem no ciclo hamiltoniano inicial.

Note que o primeiro e o último elemento da lista de vértices nunca estarão no mesmo ciclo. Para contornar este problema, o modelo de programação dinâmica é executado para cada uma das $|V|$ rotações dos nós. Sendo assim, o *Giant Tour* possui complexidade $O(|V|^4)$.

O ciclo hamiltoniano utilizado como ponto de partida para o *Giant Tour* é construído a partir da heurística gulosa descrita em Brassard [1988], sendo escolhido o nó de menor índice como ponto de partida para a heurística gulosa.

Apesar de ser válida, uma solução gerada pelo *Giant Tour* pode conter muitos nós para a solução do CHpMP. Para aprimorar essa solução inicial obtida através do GTH, percorremos todos os vértices v da solução. Se todos os nós do conjunto de nós cobertos por v, $D(v)$, são cobertos por algum outro nó diferente de v, então v pode ser removido da solução. Note que a remoção de um vértice não pode violar a restrição de pelo menos três vértices em cada sub-ciclo.

Um pseudo-código para a heurística de geração da solução inicial é apresentado no Algo-

ritmo (3).

Algoritmo 3: Meta-heurística de geração de solução inicial

```
início
    Gerar ciclo hamiltoniano S a partir de algoritmo guloso.
    Particionar ciclo hamiltoniano S de maneira ótima em p ciclos utilizando a
     formulação (12)
    para cada vértice v, v ∈ S faça
        remover = true;
        para cada vértice d, v ∈ D(v) faça
            se Vértice d não é coberto por outro vértice diferente de v então
                remover = false;
                break;
        se remover = true e ciclo que o vértice v pertence tem mais do que três
         vértices então
            remove v da solução;
```

4. Experimentos Computacionais

Nesta seção são apresentados os resultados computacionais para instâncias do TSPLib Reinelt [2016] com dimensões que variam de 22 a 100 nós. Nos ensaios computacionais foi utilizado $k = 4$ para definição da cobertura conforme descrito na seção 1. A solução inicial utilizada como *warm-start* pelo solver foi gerada através do Algoritmo (3).

A formulação PLI foi implementada utilizando o solver Gurobi, com uma hora de tempo de processamento para cada ensaio. Os ensaios foram conduzidos em um MacBook Pro 2019, SO MacOS BigSur com processador Intel i7 2,6GHz e 16 GB de RAM. O algoritmo de Gomory-Hu utilizado encontra-se na biblioteca Lemon em C++.

Na Tabela 1 emcontram-se as seguintes informações:

- Instância: Nome da instância.
- p: Quantidade de ciclos na solução final.
- BS: Best solution - melhor solução encontrada.
- gap_0: Gap inicial gerado pela solução de partida em porcentagem. Foi utilizado o resultado da relaxação linear gerada pelo Gurobi como limitante dual para determinar o gap inicial.
- gap: Gap de otimalidade em porcentagem.
- tempo: Tempo utilizado pelo solver em segundos.

Vale destacar que se uma instância do CHpMP possui um número de ciclos p igual a exatamente um terço da quantidade de nós total, ela também é uma instância do HpMP pois, como o número mínimo de vértices por ciclo é três, todos os vértices estarão presentes na solução final.

Através dos ensaios pode-se notar que o desempenho da meta-heurística apresentada na Seção 2.1 utilizada na geração de soluções iniciais apresentou maior desempenho quando valores maiores de p foram empregados. Por exemplo, o ensaio com a instância att48 teve um gap inicial de 75.2% com $p = 4$ e 42% com $p = 16$. Este padrão pode ser observado na maioria dos ensaios,

com raras excessões, como é o caso do ensaio com a instância berlin52, que possui gap inicial de 74.4% para $p = 5$ e 85.2% para $p = 7$. Apesar disto, os gaps gerados pelas soluções iniciais ainda são considerados altos, existindo espaço para melhoria neste aspecto.

Dos 59 ensaios, 18 geraram uma solução ótima e apenas 14 geraram gaps maiores do que 10%. Para a maior parte das instâncias, os ensaios com os maiores gaps foram aqueles com o maior valor de p, com excessão de st70, pr76, rat99, kroB, kroD.

5. Considerações finais

Este trabalho apresentou um novo problema combinatório, o problema das p-medianas hamiltonianas com cobertura (CHpMP), sendo este uma generalização do CSP. Também foi apresentado uma formulação em programação linear inteira para o problema, e um algoritmo de separação utilizando a árvore de Gomory-Hu. Uma série de ensaios foi realizado utilizando instâncias da biblioteca TSPLib. Como trabalhos futuros pretende-se investigar a adoção de meta-heurísticas para o tratamento de instâncias de grande porte.

Tabela 1: Resultados dos experimentos computacionais.

Instância	p	BS	gap_0	gap	tempo	Instância	p	BS	gap_0	gap	tempo
att48	4	17226.2	75.2	0,00	750.39	pr76	7	69389.6	81.4	10.2	3655.76
att48	6	16274.5	72.2	0,00	38.56	pr76	10	63630.1	78.8	0.08	3634.98
att48	9	18190.8	69.2	0,00	130.0	pr76	15	66285	71.7	0.00	3440.52
att48	12	23142.1	52.4	6.51	3608.0	pr76	19	77145.6	64.6	4.56	3605.36
att48	16	37700.8	42.0	12.95	3600.18	pr76	25	108990.6	49.2	9.79	3600.10
ulysses22	2	55.3	48.3	0.00	2.03	rat99	9	841.6	89.0	32.82	3650.09
ulysses22	3	54.3	50.4	0.00	1.09	rat99	14	622.8	89.2	4.65	3614.9
ulysses22	4	53.2	46.1	0.00	0.96	rat99	19	721.4	87.3	4.49	3600.22
ulysses22	5	52.7	43.9	0.00	0.22	rat99	24	901.3	58.2	8.27	3600.77
ulysses22	7	62.3	36.9	0,00	1.43	kroA100	10	12172.1	91.3	9.09	3617.1
eil51	5	208.1	83.3	0.00	697.55	kroA100	14	11876.8	93.5	7.61	3605.58
eil51	7	203.1	84.8	0,00	217.7	kroA100	20	11518.9	90.3	0.00	2650.97
eil51	10	244.5	60.1	0.92	3601.07	kroA100	25	13476.9	61.8	3.61	3657.59
eil51	12	291.5	53.9	4.17	3603.67	kroA100	33	22253.4	42.9	14.24	3600.32
eil51	17	489.5	35.7	13.7	3619.4	kroB100	10	12525.9	78.3	3.77	3600.70
berlin52	5	4432.8	74.4	0,00	616.59	kroB100	14	19724.7	77.1	39.52	3600.76
berlin52	7	4473.0	85.2	2.79	3616.8	kroB100	20	12628.0	70.4	0.00	432.86
berlin52	10	4462.0	61.2	0,00	39.69	kroB100	25	14698.1	61.9	1.98	3623.64
berlin52	13	5053.5	57.5	0.29	3607.2	kroB100	33	22921.2	28.9	11.79	3600.2
berlin52	17	7449.8	37.7	7.53	3604.32	kroC100	10	12484.1	91.6	12.2	3602.87
st70	7	438.2	87.7	18.8	3729.64	kroC100	14	11179.9	93.3	0.30	3600.58

Continua na próxima página

Continuação da página anterior

Instância	p	BS	gap_0	gap	tempo	Instância	p	BS	gap_0	gap	tempo
st70	10	363.8	88.1	2.2	3641.91	kroC100	20	12008.6	68.2	0.00	1724.06
st70	14	379.2	87.8	0.00	22.1	kroC100	25	14024.6	58.8	2.03	3626.63
st70	17	441.6	52.4	0.93	3602.87	kroC100	33	32432.2	43.3	40.0	3601.0
st70	23	685.4	30.4	10.67	3607.28	kroD100	10	12281.4	77.3	3.52	3605.45
eil76	7	280.4	76.1	4.57	3604.73	kroD100	14	12484.1	74.3	12.20	3602.87
eil76	10	279.1	71.2	3.23	3607.56	kroD100	20	12785.4	64.6	0.00	252.93
eil76	15	334.3	61.6	5.0	3603.16	kroD100	25	14883.3	56.8	2.30	3647.97
eil76	19	410.8	52.4	6.67	3600.66	kroD100	33	22093	29.6	10.07	3600.36
eil76	25	605.7	38.4	13.11	3600.12						

Referências

Agarwal, S. U., U. (2009). *Graph Theory*. Laxmi Publications.

Applegate, R. E. C. V. C. W. J., David L.; Bixby (2007). *The Traveling Salesman Problem: A Computational Study*. Princeton University Press.

Brassard, P., Gilles; Bratley (1988). *Algorithmics theory and practice*. Prentice-Hall.

Current, J. R. e Schilling, D. A. (1989). The covering salesman problem. *Transportation science*, 23(3):208–213.

Garey, J. D. S., Michael R. (1990). *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York.

Gomory, R. E. e Hu, T. C. (1961). Multi-terminal network flows. *Journal of the Society for Industrial and Applied Mathematics*, 9(4):551–570.

Groër C, W. E., Golden BL (2008). The consistent vehicle routing problem. *Manufacturing & Service Operations Management*, 11(4):630–643. URL `http://dx.doi.org/10.1287/msom.1080.0243`.

Günes Erdogan, A. M. R. C., Gilbert Laporte (2016). Exact and heuristic algorithms for the hamiltonian p-median problem. *European Journal of Operational Research*, 253:280–289.

Hongtao Lei, G. L., Rui Wang (2016). Solving a multi-objective dynamic stochastic districting and routing problem with a co-evolutionary algorithm. *Computers & Operations Research*, 67: 12–24.

Maziero, L. P. (2016). Algoritmos heurísticos e exatos para o problema do ciclo dominante. *Dissertação (mestrado) – Universidade Estadual de Campinas, Instituto de Computação.*, p. 29–32.

Reinelt, G. (2016). Tsp lib. `http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsplib.html`. Acessado: 2020-06-21.