# O PROBLEMA DO CICLO DOMINANTE

**Lucas Porto Maziero**
Instituto de Computação
Universidade Estadual de Campinas (UNICAMP) Campinas - SP - Brasil
lucasporto1992@gmail.com

**Fábio Luiz Usberti**
Instituto de Computação
Universidade Estadual de Campinas (UNICAMP) Campinas - SP - Brasil
fusberti@ic.unicamp.br

**Celso Cavellucci**
Instituto de Computação
Universidade Estadual de Campinas (UNICAMP) Campinas - SP - Brasil
ccavellucci@gmail.com

## RESUMO

Este artigo apresenta uma generalização do Problema do Caixeiro Viajante (TSP), denominado Problema do Ciclo Dominante (DCP). Essa generalização consiste na composição de dois problemas NP-difíceis: o TSP e o Problema do Conjunto Dominante em Grafos. Brevemente, o objetivo do DCP consiste em encontrar um ciclo de custo mínimo em um grafo não-direcionado, partindo de um nó origem. O ciclo é trafegado por um viajante que necessita visitar um conjunto de clientes (nós dominantes). A motivação do DCP consiste em sua aplicação prática para empresas de distribuição de energia, gás ou água; em particular, no processo de definição das rotas para a leitura dos consumos desses serviços. Neste artigo, são apresentadas metodologias de solução para o DCP, que consistem de formulações de programação linear inteira para os dois problemas que o compõe. Também é apresentado a solução do problema por meio de uma metaheurística, considerando a complexidade da solução exata do DCP.

**PALAVRAS CHAVE. Caixeiro Viajante, Conjunto Dominante em Grafos, Otimização Combinatória, Metaheurística.**

**Área Principal: Otimização Combinatória, Teoria da Computação e Metaheurística.**

## ABSTRACT

This paper presents a generalization of the Traveling Salesman Problem (TSP), called Dominant Cycle Problem (DCP). This generalization is the composition of two NP-hard problems: the TSP and the Dominant Set Problem in Graphs. Briefly, the aim of the DCP is to find a minimum cost cycle in an undirected graph, from a source node. The cycle is trafficked by a traveler who needs to visit a set of customers (we dominant). The motivation of the DCP is in its practical application to power distribution companies, gas or water; in particular, the process of defining routes for reading the consumption of such services. In this article, solution methodologies are presented for the DCP, which consist of integer linear programming formulations for the two problems that compose. It also presents the solution of the problem through a metaheuristic, considering the complexity of the exact solution of the DCP.

**KEYWORDS. Traveling Salesman, Dominant Set in Graphs, Combinatorial Optimization, Metaheuristic.**

**Main Area: Combinatorial Optimization, Theory of Computation and Metaheuristic**

## 1. Introdução

Existem diversos tipos de aplicações no mundo real referente a problemas de distribuição, como coleta de lixo, rotas de ônibus escolares, serviços de correio postal, operações de frete, entrega de mercadorias, coleta de consumo de energia residencial, dentre outros. As soluções para esses problemas são focadas em diminuir os custos da distribuição, o que possibilita economia significativa para as entidades provedoras desses serviços. Porém, grande parte desses problemas são de difícil solução, o que motiva o interesse na pesquisa dessa classe de problemas conhecidos como problemas de roteamento.

No ramo de otimização combinatória, em particular dos problemas de roteamento, um problema representativo na literatura que já foi amplamente investigado é o Problema do Caixeiro Viajante, do inglês *Traveling Salesman Problem* (TSP).

O TSP é um dos problemas mais investigados na literatura em otimização combinatória. Chatterjee et al. (1996), Laporte et al. (2000) e Applegate (2006) mostram que o TSP pode ser aplicado em muitos problemas reais de diversas áreas do conhecimento, como logística, genética, manufatura, telecomunicações e até mesmo neurociência.

Neste trabalho, é investigado o Problema do Ciclo Dominante, do inglês *Dominant Cycle Problem* (DCP), um problema inédito na literatura que generaliza o TSP e o Problema do Conjunto Dominante em Grafos, do inglês *Dominating Set Problem* (DSP).

O TSP é definido do seguinte modo: dado um grafo completo e não-direcionado $G(V, E)$, onde $V$ representa o conjunto de vértices e $E$ o conjunto de arestas. Cada aresta $(i, j) \in E$ possui um custo não-negativo $c_{ij} \geqslant 0$ (Papadimitriou and Steiglitz (1998)). Para este trabalho, serão assumidos custos simétricos nas arestas, $c_{ij} = c_{ji}$ . Seja a definição de ciclo hamiltoniano em $G$ um ciclo que contém todos os vértices de $G$ (Bondy and Murty (1976)). Então como solução o TSP busca encontrar, partindo de um vértice de origem, um ciclo hamiltoniano de custo mínimo que visita todos os vértices do grafo.

No DSP, é dado um grafo não-direcionado $G = (V, E)$ onde $V$ é o conjunto vértices e $E$ é o conjunto das arestas. Um subconjunto de vértices $D \subseteq V$ é um conjunto dominante de $G$ se para cada vértice $v \in V \setminus D$ existe um vértice $u \in D$ tal que $(u, v) \in E$ (Chen et al. (2012)). Uma extensão do DSP, denominado $k$-DSP, generaliza a definição de conjunto dominante a partir do conceito de *vizinhança* entre nós. No DSP um nó $i$ é vizinho de um nó $j$ se e somente se $i$ é adjacente a $j$. Já no $k$-DSP, um nó $i$ é vizinho de um nó $j$ se e somente se $d(i, j) \leqslant k$, onde $d(i, j)$ é a distância (ou custo) de um caminho mínimo entre os nós $i$ e $j$, enquanto $k$ é denominado como o *raio de vizinhança*.

Uma motivação para o estudo do DCP consiste em sua aplicação para um problema real de roteamento de leituristas. Empresas que atuam como fornecedores de energia elétrica, água e gás, possuem colaboradores responsáveis por registrar o consumo dos clientes dispersos geograficamente nas ruas e avenidas dos bairros de um centro urbano. De acordo com Usberti et al. (2008, 2011b,a), o roteamento desses funcionários, denominados leituristas, gera um problema de difícil solução, que consiste na elaboração de rotas nas quais os leituristas devem percorrer para realizar a leitura de consumo dos clientes.

Brasek (2004) e Jamil (2008) mostram que na última década, o avanço tecnológico introduziu os aparelhos de leitura remota , o que permite que um leiturista possa realizar um registro de consumo dentro de um raio de alcance máximo do aparelho. Nesse sentido, as rotas dos leituristas não precisam necessariamente visitar todos os clientes, contanto que todos os clientes estejam dentro do raio de alcance da rota. Uma solução ótima para esse problema real minimiza os tempos de percurso dos leituristas, de modo que todos os clientes do centro urbano tenham seus consumos registrados.

Este trabalho está organizado da seguinte maneira. Na Seção 2 é apresentado formalmente o DCP. Na Seção 3 são apresentadas formulações matemáticas de Programação Linear Inteira (PLI) para o TSP e $k$-DSP. A Seção 4 descreve as metodologias propostas para a solução do DCP. Na

Seção 5 são mostrados resultados de experimentos computacionais realizados com instâncias geradas aleatoriamente. Finalmente a Seção 6 apresenta os comentários finais.

## 2. Problema do Ciclo Dominante

A descrição do DCP é fornecida a seguir: dado um grafo completo e não-direcionado $G(V, E)$, onde $V$ representa o conjunto de vértices e $E$ o conjunto de arestas. Cada aresta $(i, j) \in E$ possui um custo não-negativo $c_{ij} \geqslant 0$. Para este trabalho, serão assumidos custos simétricos nas arestas, $c_{ij} = c_{ji}$. No conjunto de vértices $V$, o vértice $0$ corresponde um nó especial, denominado raiz. Um subconjunto de vértices $D \subseteq V$ é um $k$-conjunto dominante de $G$ se para cada vértice $v \in V \setminus D$ existe um vértice $u \in D$ tal que $d(v, u) \leqslant k$, onde $d(v, u)$ é a distância (ou custo) de um caminho mínimo entre os nós $v$ e $u$, enquanto $k$ é o raio de vizinhança. Como solução, o DCP busca encontrar, partindo do vértice raiz um ciclo de custo mínimo que visita todos os vértices de $D$.

A Figura 1 ilustra uma solução viável para uma instância do DCP. Os pontos em cor vermelha representam o subconjunto de nós que dominam o grafo; o ponto em cor azul corresponde ao nó raiz; o traçado da rota encontra-se em cor preta; a vizinhança de um nó do conjunto dominante é demonstrada pelas circunferências em verde.

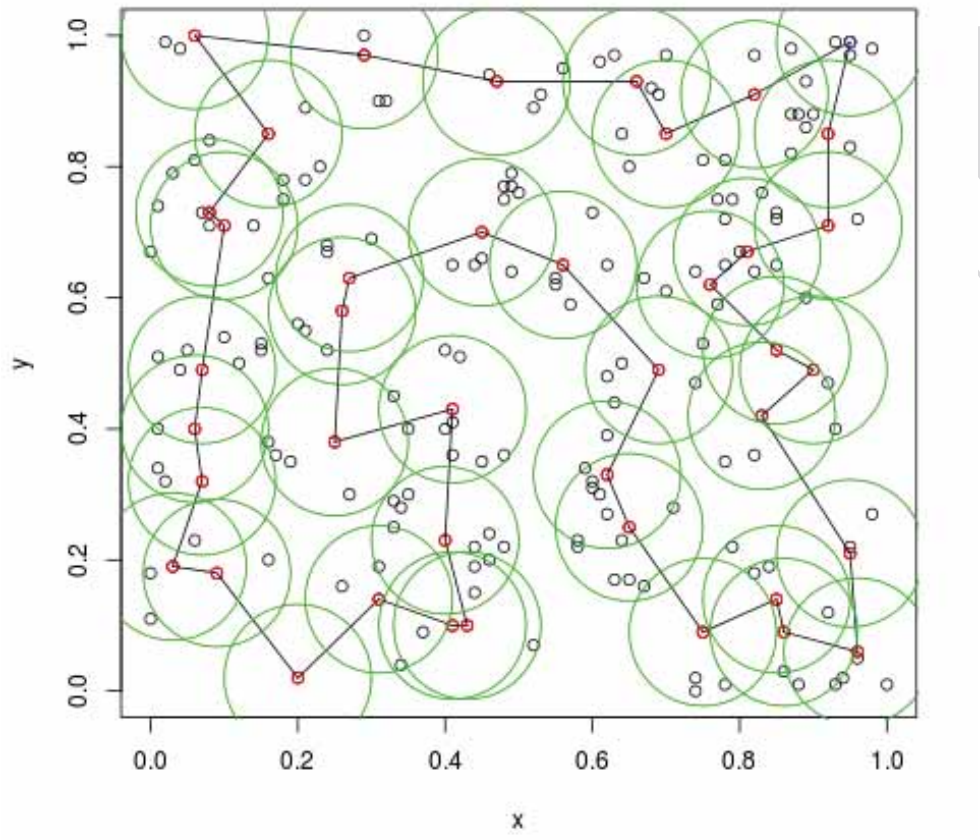


Figura 1: Solução factível para o DCP, com 200 nós e raio de vizinhança $k = 0.10$.

## 3. Formulações de PLI

Nesta seção são apresentadas duas formulações de PLI para os problemas que compõe o DCP, a primeira para o TSP e a segunda para o $k$-DSP.

### 3.1. Formulação PLI para o TSP

Uma formulação de PLI para o TSP é apresentada abaixo (Goldbarg and Luna (2005)), onde a variável binária $x_{ij}$ representa se um aresta $(i, j) \in E$ pertente (1) ou não (0) ao ciclo de custo mínimo. $S$ é um subconjunto de vértices de $G$ e $\delta(v)$ corresponde ao conjunto de nós adjacentes ao vértice $v$.

$$MIN \quad \sum_{(i,j)\in E} c_{ij}x_{ij} \tag{1}$$

s.a.

$$\sum_{i\in\delta(j)} x_{ij} = 1, \qquad \forall j \in V \tag{2}$$

$$\sum_{j\in\delta(i)} x_{ij} = 1, \qquad \forall i \in V \tag{3}$$

$$\sum_{i,j\in S} x_{ij} \leq |S| - 1, \qquad \forall S \subset V \tag{4}$$

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

A função objetivo (1) busca por um ciclo de custo mínimo, onde o custo de uma solução é dado pela somatória dos custos associados das arestas que pertencem ao ciclo. A restrição (2) determina os fluxos de chegada dos vértices. A restrição (3) determina os fluxos de saída dos vértices. O conjunto de restrições (2) e (3) garantem que qualquer vértice poderá ser visitado pelo ciclo uma única vez. A restrição (4) garante a não ocorrência de ciclos desconexos entre os nós, ou seja, evita os circuitos pré-hamiltonianos.

### 3.2. Formulação PLI para o *k*-DSP

A seguir é apresentada uma formulação PLI para $k$-DSP (Chen et al. (2012)), onde a variável binária $y_i$ revela quando um nó pertence (1) ou não (0) ao conjunto dominante. O conjunto de nós $N_k(i)$ corresponde à vizinhança do nó $i$, considerando o raio de vizinhança $k$.

$$MIN \quad \sum_{i\in V} y_i \tag{6}$$

s.a.

$$\sum_{j\in N_k(i)} y_j \geqslant 1 \qquad \forall i \in V \tag{7}$$

$$y_i \in \{0, 1\} \qquad \forall i \in V \tag{8}$$

A função objetivo (6) visa um conjunto dominante de cardinalidade mínima. A restrição (7) garante que todo vértice deve ser vizinho de pelo menos um elemento do conjunto dominante.

## 4. Metodologias

Nesta seção, são apresentadas três metodologias de solução para o DCP. A primeira é constituída basicamente por duas fases, uma para resolver o $k$-DSP e outra para o TSP. A segunda metodologia proposta é a implementação de uma metaheurística, quando uma formulação exata para o DCP ser computacionalmente inviável para determinadas instâncias. Já a terceira metodologia consiste de uma abordagem híbrida, onde brevemente a solução obtida pela metaheurística será utilizada na primeira metodologia. As subseções seguintes apresentam detalhadamente todas as metodologias.

### 4.1. Metodologia de Duas Fases

A primeira metodologia de solução proposta para o DCP consiste em um método de duas fases: a primeira fase resolve $k$-DSP através do modelo de PLI apresentado para o problema na seção anterior, resultando em um subconjunto de nós que dominam o grafo. A segunda fase busca traçar um ciclo sobre os nós pertencentes ao conjunto dominante obtido na primeira fase, sendo assim, esses nós pertencentes ao conjunto dominante são utilizados do modelo de PLI apresentado na seção anterior para o TSP. O ciclo da segunda fase trata-se portanto de uma solução viável (não necessariamente ótima) para o DCP, uma vez que seus vértices dominam o grafo.

### 4.2. Metodologia GRASP

Considerando que a solução exata do DCP requer um tempo exponencial, exceto se P=NP, este trabalho computacional propõe uma segunda metodologia de solução para o DCP através da implementação de uma metaheurística. Nesse sentido, para as instâncias onde a aplicação de uma metodologia exata se tornar inviável, uma metaheurística é proposta como alternativa para obter soluções de boa qualidade (sem garantias de otimalidade) em um tempo computacional adequado.

O GRASP (*Greedy Randomized Adaptive Search Procedure*) é a metaheurística adotada para a solução do DCP neste trabalho. Resende (2001) apresenta o GRASP em composição de dois estágios por iteração: o primeiro é responsável pela construção de uma solução inicial a partir de uma heurística construtiva aleatória gulosa. O segundo estágio realiza uma busca local no intuito de explorar a vizinhança da solução inicial até atingir um mínimo local do espaço de soluções. Após um certo número de iterações, a melhor solução encontrada é retornada como solução final da metaheurística, conforme o pseudocódigo abaixo.

**Algorithm 1:** Metaheurística GRASP

```
Entrada: Grafo completo e não-direcionado G(V, E)
Saída: Ciclo Dominante
1 inicio
2     enquanto condiçãoTérminoNãoEncontrada faça
3         solução = constróiSoluçãoInicial();
4         solução = buscaLocal(solução);
5         atualizaSolução(solução, melhorSolução);
6     retorna melhorSolução;
```

No estágio de construção da solução inicial, os elementos candidatos a participarem da solução inicial são escolhidos de modo aleatório e guloso. Esse processo se realiza a partir de uma lista restrita de candidatos (*restricted candidate list*, RCL) que recebe os melhores candidatos a entrar na solução. Por exemplo, no caso do DSP os melhores candidatos poderiam ser os nós de maior grau por serem mais promissores a fazerem parte de um conjunto dominante ótimo. A partir da lista RCL, os candidatos que efetivamente entram na solução são escolhidos aleatoriamente com distribuição uniforme.

As soluções encontradas no primeiro estágio do GRASP não são necessariamente ótimos locais, logo o segundo estágio passa a explorar a vizinhança dessas soluções. Uma vizinhança é definida como um conjunto de soluções que são alcançáveis a partir de uma solução inicial após a aplicação de um operador de busca local. Uma solução é ótima local quando na sua vizinhança não existe uma solução de melhor custo.

Para a metaheurística GRASP proposta, as variáveis de decisão representam o conjunto de arestas das instâncias. O custo final da solução é dado pela somatória de todos os pesos das arestas que estão sendo utilizadas na rota dominante (ciclo).

#### 4.2.1. Heurística Construtiva

A heurística construtiva é dividida em duas etapas: na primeira etapa, é definido um $k$-conjunto dominante de vértices $D \subseteq V$, garantindo que para cada vértice $v \in V \setminus D$ existe um

vértice $u \in D$ tal que $d(v,u) \leqslant k$, onde $d(v,u)$ é a distância do caminho mínimo entre $v$ e $u$ e $k$ é o raio de vizinhança; na segunda etapa é definido uma rota que se inicia no vértice raiz (0), utilizando o algoritmo do vizinho mais próximo sobre o $k$-conjunto dominante definido na etapa anterior.

Na primeira etapa, a descrição da lista restrita de candidatos 1 (RCL 1) é dada a seguir. A cada iteração da primeira etapa da heurística construtiva, seja $S$ o conjunto de vértices já adicionados ao $k$-conjunto dominante $D$, o conjunto $C$ de candidatos é constituído da seguinte maneira: para todo vértice $v_i \in V \setminus S$, o conjunto $A_i$ representa todos os vértices que estão a uma distância $d$ de $v_i$ cujo $d \leqslant k$ e que não estão à uma distância $d$ de um vértice $u \in S$ cujo $d \leqslant k$. Então, $c_{max}$ e $c_{min}$ denotam conjuntos $A_i$ de maior e menor cardinalidade, respectivamente, sendo a RCL 1 formada pelos elementos de $C$ que satisfazem $c_{v_i} \geqslant c_{min} + \alpha(c_{max} - c_{min})$, onde $\alpha$ foi definido como um valor randomizado dentre as possibilidades $\{0.70, 0.75, 0.80, 0.85, 0.90\}$ com iguais chances. Esses valores foram definidos de maneira empírica.

Durante a primeira etapa, a escolha de um vértice $v \in V \setminus D$ a ser adicionado no $k$-conjunto dominante $D$ é dado pela RCL 1 através de um critério guloso, escolhendo um vértice que esteja na RCL 1 randomicamente. Após ter um $k$-conjunto dominante definido, passa-se para a segunda etapa.

Na segunda etapa, a descrição da lista restrita de candidatos 2 (RCL 2) é dada a seguir. A cada iteração da segunda etapa da heurística construtiva, seja $S$ o conjunto de vértices dominantes já adicionados à solução, seja $v_0$ o vértice corrente e seja $C$ o conjunto de candidatos $\{(v_0, v_i), v_i \in D \setminus S\}$, onde $c_{max}$ e $c_{min}$ denotam o maior e menor custo deste conjunto, a RCL 2 é então formada pelos vizinhos elementos de $C$ que satisfaçam à regra $c_{v_0 v_i} \leq c_{min} + \alpha(c_{max} - c_{min})$, sendo $\alpha$ definido como um valor randomizado dentre as possibilidades $\{0.01, 0.02, 0.03, 0.04, 0.05\}$ com iguais chances. Esses valores foram definidos de maneira empírica.

A segunda etapa é dada através da implementação do algoritmo do vizinho mais próximo sobre o $k$-conjunto dominante de vértices $D$ definido na primeira etapa, onde a construção da rota se inicia no vértice raiz (0) e repetidamente, são verificados os custos das arestas dos vértices vizinhos mais próximos de acordo com a regra da RCL 2, considerando apenas os vértices que ainda não foram visitados. Então é escolhido um vértice vizinho que esteja na RCL 2 randomicamente. O algoritmo repete o procedimento agora com este vértice vizinho recentemente adicionado até que haja os $|D|$ (tamanho do conjunto dominante) vértices na solução.

O pseudocódigo abaixo representa o estágio de construção da solução inicial do GRASP.

**Algorithm 2:** Metaheurística GRASP - Estágio de Construção da Solução Inicial

```
1  inicio
2      conjuntoDominante = {};
3      enquanto conjuntoDominante não é um conjunto dominante válido faça
4          Construa a lista restrita de candidatos (RCL 1);
5          Selecione um elemento s da RCL 1 aleatoriamente;
6          conjuntoDominante = conjuntoDominante ∪{s};
7      cicloDominante = {} ;
8      enquanto cicloDominante não estiver completo faça
9          Construa a lista restrita de candidatos de acordo com conjuntoDominante
           (RCL 2);
10         Selecione um elemento s da RCL 2 aleatoriamente;
11         cicloDominante = cicloDominante ∪{s};
12     retorna cicloDominante;
```

#### 4.2.2. Operadores de Buscas Locais e o Método de busca

O estágio de busca local é composto por dois operadores de busca. O primeiro operador de busca (vizinhança) escolhido é o 2-OPT (Croes (1958)). A vizinhança 2-OPT é dada da

seguinte maneira: é vizinho da solução corrente todo o conjunto de soluções possíveis encontradas invertendo uma sequência de cidades consecutivas de uma rota. Desse modo, duas cidades são selecionadas da solução corrente e a sequência de cidades da rota que se encontram entre as cidades selecionadas são rotacionadas. Como exemplo, dado uma solução corrente $\{1, 2, 3, 4, 5, 6\}$ e as cidades 2 e 5 selecionadas, a proposta de vizinhança é gerada com a rotação feita sobre as cidades $2, 3, 4, 5$, gerando então a proposta de vizinhança $\{1, 5, 4, 3, 2, 6\}$. Com a rotação realizada, as arestas utilizadas na solução corrente $(1, 2)$ e $(5, 6)$ são retiradas da rota e as arestas $(1, 5)$ e $(2, 6)$ são inseridas.

O segundo operador de busca é denominado *swap(*1, 1*)*. Esse operador efetua movimentos sobre o ciclo dominante da seguinte maneira: para cada vértice $u \in D$, onde $D \subseteq V$ é o conjunto de vértices dominantes que formam o ciclo, é avaliado uma possível troca desse vértice $u$ com outro vértice qualquer $v \in V \setminus D$ se e somente se o vértice $v$ dominar todos os vértices dominados exclusivamente por $u$. Essa avaliação é dada pelo custo da função objetivo quando o vértice $v$ é inserido na mesma posição no ciclo que vértice $u$ ocupava. Se o custo da função objetivo for minimizado, a troca é efetivada e o vértice $v$ será inserido em $D$, enquanto o vértice $u$ será excluído de $D$. Todas as trocas são possíveis, exceto quando o vértice $u = 0$, impedindo que o nó raiz seja retirado do conjunto $D$.

O método de busca utilizado para as os dois operadores foi o *first-improving*, onde basicamente, a solução corrente desloca-se para o seu primeiro vizinho cujo o custo da função objetivo é menor do que o custo da função objetivo da solução corrente.

Vale ressaltar que o operador 2-OPT é executado antes e pós o operador *swap(*1, 1*)*. Essa sequência foi determinada pelo critério de realizar primeiro movimentos intra-ciclo, buscando por ótimos locais que não alterem a composição dos nós que fazem parte do ciclo. Após movimentos intra-ciclo, são efetuados movimentos sobre o ciclo dominante em busca de possíveis trocas de nós utilizando o operador *swap(*1, 1*)*, surgindo então a necessidade de novos movimentos intra-ciclo realizados pelo operador 2-OPT, já que a composição dos nós que fazem parte do ciclo foi alterada pelo *swap(*1, 1*)* e ocasionalmente, um novo ciclo dominante é encontrado. Após a execução dos dois operadores de busca, a solução é retornada com suas devidas otimizações locais feitas, conforme o pseudocódigo abaixo.

**Algorithm 3:** Metaheurística GRASP - Estágio de Busca Local

```
1 inicio
2     aplica2OPT(solução);
3     swap(solução);
4     aplica2OPT(solução);
```

#### 4.2.3. Critérios de Parada

O algoritmo se encerra após o tempo decorrido de execução de 60 minutos.

### 4.3. Metodologia Híbrida

A terceira metodologia de solução proposta para o DCP é semelhante à primeira. A metodologia híbrida também é composta por duas fases: a primeira fase resolve o DCP através da metaheurística GRASP apresentada na subseção anterior, resultando em um ciclo cujo seus nós dominam o grafo. A segunda fase busca traçar um novo ciclo sobre os mesmos nós pertencentes ao ciclo dominante alcançado pela metaheurística na primeira fase. Essa segunda fase é executada através do modelo de PLI para o TSP apresentado na seção anterior, construindo um novo ciclo. Esse novo ciclo trata-se portanto de uma solução viável (não necessariamente ótima) para o DCP.

## 5. Experimentos Computacionais

Os resultados consistem em experimentos computacionais sobre os modelos matemáticos apresentados na Seção 3, que serviram de base para a metodologia de duas fases apresentada na Seção 4. Também faz parte dos resultados experimentos computacionais sobre a metaheurística

GRASP e a metodologia híbrida, ambas apresentadas na Seção 4. As instâncias utilizadas nos ensaios foram geradas distribuindo-se pontos (nós) em um plano de dimensões $1 \times 1$ com coordenadas $(x, y)$ geradas aleatoriamente no intervalo $[0, 1]$ com distribuição uniforme. As instâncias são categorizadas pelo par $(n, k)$, tal que $n$ é o número de nós e $k$ é o raio de vizinhança. Foram geradas instâncias com valores de $n = \{200, 400, 600, 800\}$ e $k = \{0; 0.02; 0.04; 0.06; 0.08; 0.1\}$.

O *solver* utilizado para a solução dos modelos $k$-DSP e TSP foi o Gurobi[1], configurado com um tempo de execução máximo de 60 minutos. A metaheurística GRASP foi implementada na linguagem de programação C++ e compilada com g++ versão 4.8.2, configurada também com um tempo de execução máximo de 60 minutos. Os experimentos computacionais foram executados em um computador com processador Intel Core i3 2,53 GHz, com 4GB de memória RAM.

A Tabela 1 apresenta os resultados dos experimentos computacionais, organizando-os por cada tipo de instâncias, definido pelo par $(n, k)$. Com relação à metodologia de duas fases, são apresentados os tamanhos dos conjuntos dominantes ($|Dm1|$), tempo de execução ($tempom1f1$) do modelo $k$-DSP; os limitantes primal ($lpm1$) e dual ($ldm1$), o gap de otimalidade ($gap(\%)m1$) correspondente e o tempo de execução ($tempom1f2$) são obtidos pela solução do modelo TSP. Para a metodologia que implementa a metaheurística GRASP, são apresentados os tamanhos dos conjuntos dominantes ($|Dm2|$) obtidos, o limitante primal ($lpm2$) alcançado e o tempo de execução ($tempom2$) decorrido. Na metodologia híbrida, encontram-se os limitantes primal ($lpm3$) e dual ($ldm3$), o gap de otimalidade ($gap(\%)m3$) correspondente e o tempo de execução ($tempom3$) obtidos pela solução do modelo TSP, já que o conjunto dominate dessa metodologia é o mesmo conjunto dominante extraído da solução da metaheurística GRASP. Todos os tempos de execuções apresentados na Tabela 1 registram o tempo decorrido de uma determinada metodologia quando a melhor solução é encontrada. Como exemplo, na metodologia que implementa a metaheurística GRASP, o tempo de execução ($tempom2$) representa o instante quando a melhor solução (limitante primal $lpm2$) foi encontrada, dentre o tempo máximo de execução de 3600 segundos.

Os resultados da Tabela 1 demonstram que o solver atingiu soluções ótimas em 21 dos 24 tipos de $(n, k)$ na metodologia de duas fases e na metodologia híbrida. Mesmo nesses casos não é possível afirmar que as soluções obtidas são ótimas para o DCP, pois uma vez que o problema é decomposto em duas fases, perde-se qualquer garantia de otimalidade. De todo modo, o TSP é por si só um problema NP-difícil, logo mesmo a solução individualizada desse problema não retornou soluções ótimas para todas as instâncias. Em geral, foi possível observar que as soluções dos modelos tornam-se mais difíceis para as instâncias cujo raio de vizinhança é baixo (próximo de 0.02).

A metodologia por metaheurística apresentou boas soluções, sendo 11 dos 24 tipos de $(n, k)$ melhores quando comparados com a metodologia de duas fases. Em geral, foi possível observar que as soluções da metaheurística tornam-se mais difíceis para as instâncias onde o raio de vizinhança é baixo (próximo de 0.04). São nessas instâncias que a metodologia híbrida apresenta soluções melhores, quando comparadas com as outras duas metodologias (Figuras 2, 3, 4 e 5).

[1]O Gurobi é um solver para problemas de programação matemática. Para otimização de problemas de programação inteira e programação inteira mista, o Gurobi utiliza o algoritmo branch-and-bound em conjunto com técnicas de plano de corte e heurísticas (Gurobi Optimization Inc., 2014).

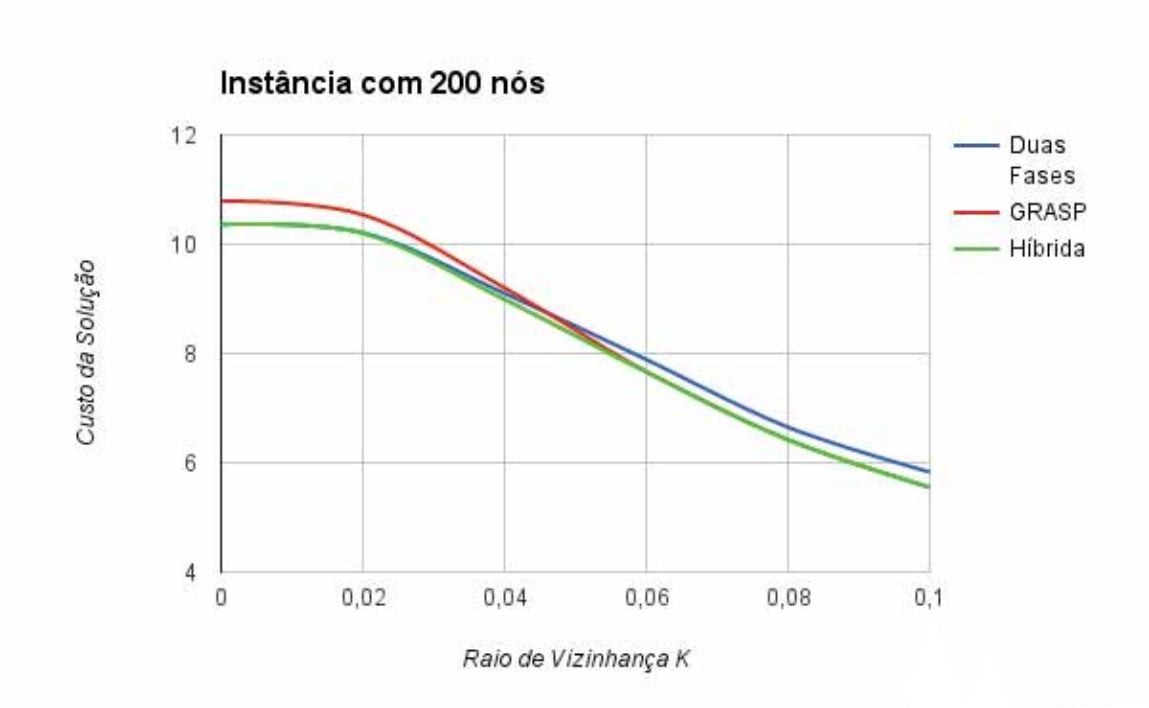


Figura 2: Custo da Solução em função da metodologia utilizada e raio de vizinhança ($k$) para instâncias com 200 nós.

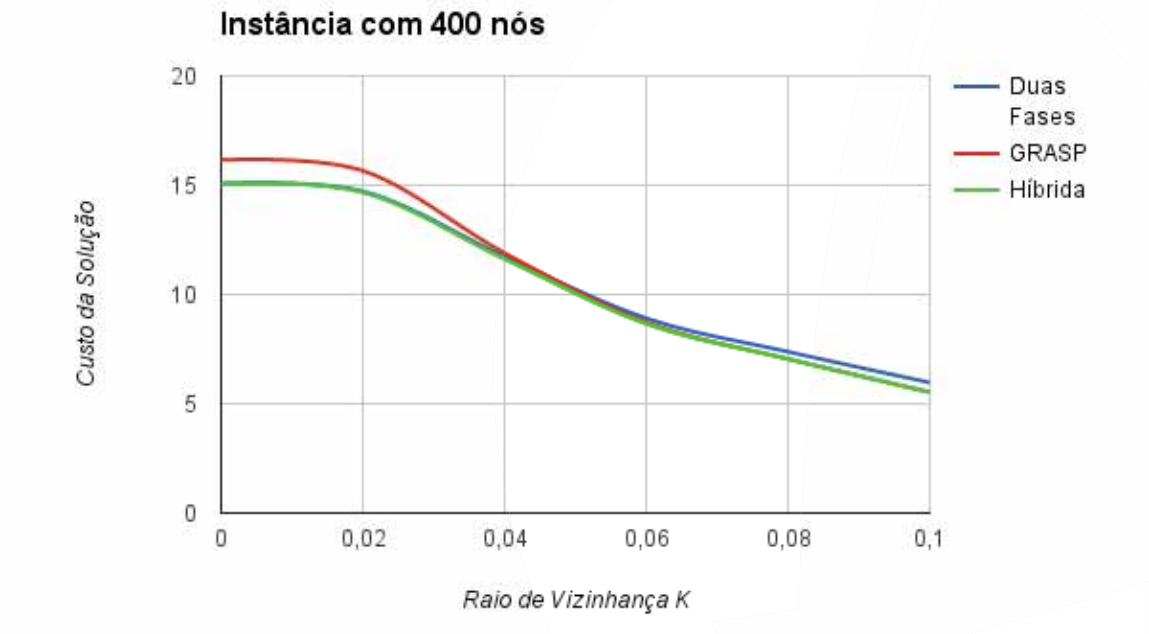


Figura 3: Custo da Solução em função da metodologia utilizada e raio de vizinhança ($k$) para instâncias com 400 nós.

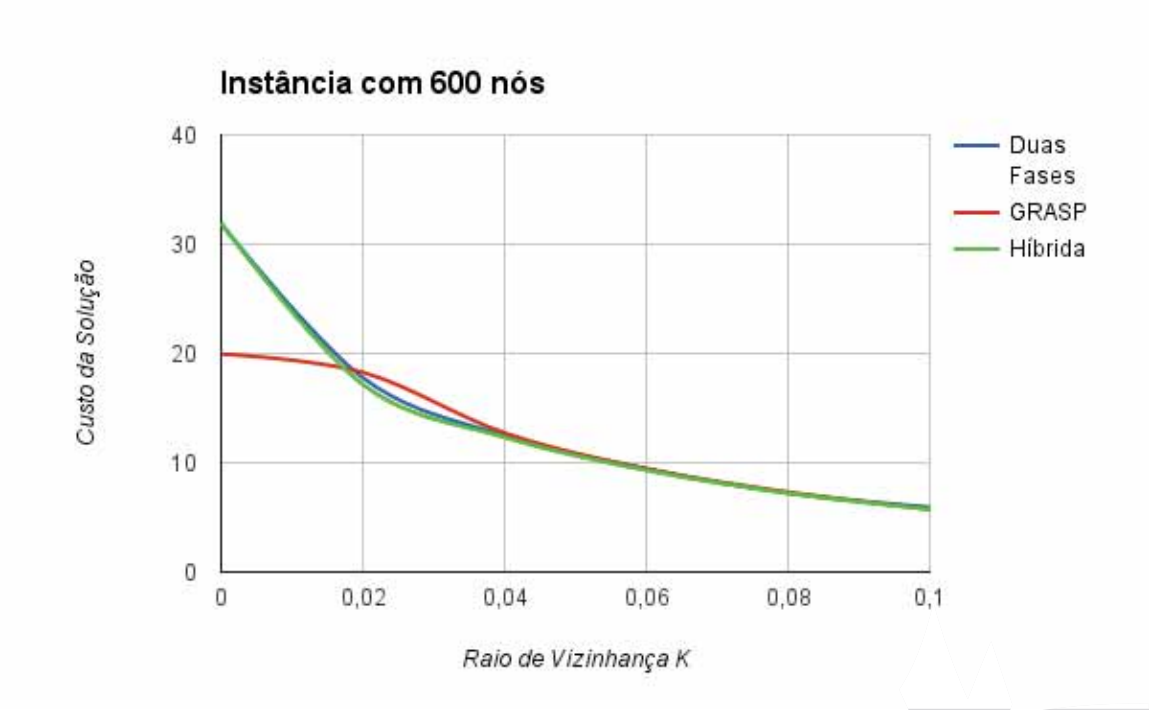


Figura 4: Custo da Solução em função da metodologia utilizada e raio de vizinhança ($k$) para instâncias com 600 nós.

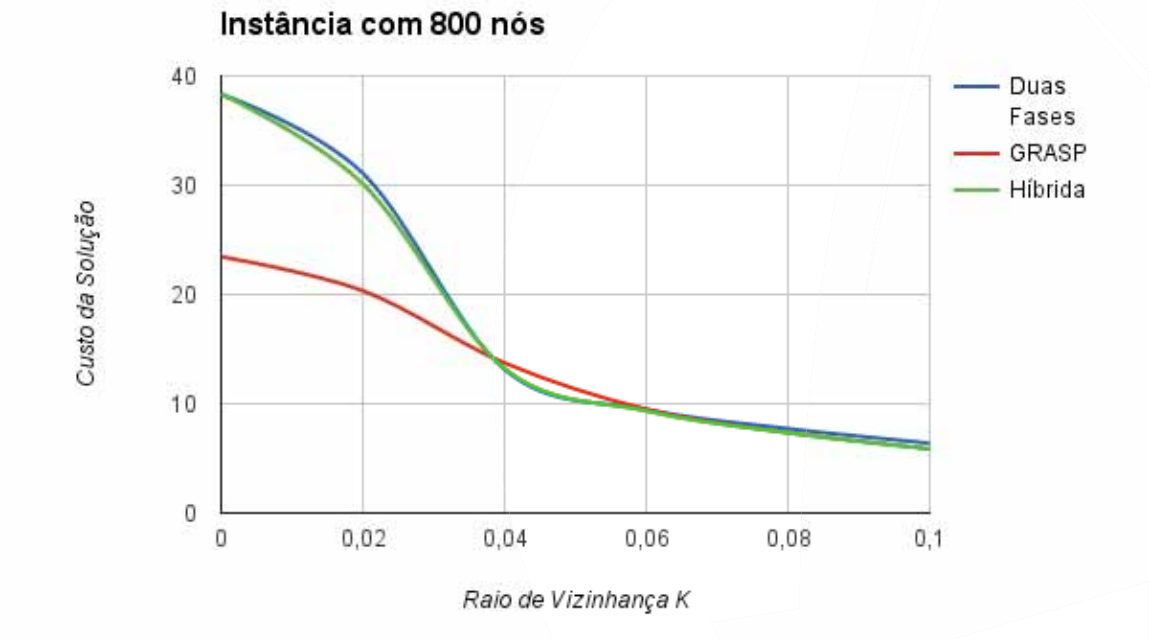


Figura 5: Custo da Solução em função da metodologia utilizada e raio de vizinhança ($k$) para instâncias com 800 nós.

Tabela 1: Resultados dos experimentos computacionais.

| | | Método de Duas Fases | | | | | | Metaheurística | | | Método Híbrido | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | k-DSP | | TSP | | | | DCP | | | TSP | | | |
| $n$ | $k$ | $\|Dm1\|$ | $tempom1f1$ | $lpm1$ | $ldm1$ | $gap(\%)m1$ | $tempom1f2$ | $\|Dm2\|$ | $lpm2$ | $tempom2$ | $lpm3$ | $ldm3$ | $gap(\%)m3$ | $tempom3$ |
| | 0,1 | 37 | 0 | 5,83 | 5,83 | 0 | 0 | 41 | 5,55 | 73 | 5,55 | 5,55 | 0 | 0 |
| | 0,08 | 50 | 0 | 6,65 | 6,65 | 0 | 0 | 55 | 6,42 | 1154 | 6,42 | 6,42 | 0 | 0 |
| 200 | 0,06 | 75 | 0 | 7,89 | 7,89 | 0 | 0 | 81 | 7,67 | 652 | 7,66 | 7,66 | 0 | 0 |
| | 0,04 | 116 | 0 | 9,1 | 9,1 | 0 | 3 | 117 | 9,2 | 221 | 8,99 | 8,99 | 0 | 0 |
| | 0,02 | 176 | 0 | 10,21 | 10,21 | 0 | 14 | 176 | 10,54 | 3250 | 10,2 | 10,2 | 0 | 20 |
| | 0 | 200 | 0 | 10,36 | 10,36 | 0 | 100 | 200 | 10,79 | 1479 | 10,36 | 10,36 | 0 | 75 |
| | 0,1 | 38 | 0 | 5,97 | 5,97 | 0 | 0 | 49 | 5,53 | 2922 | 5,53 | 5,53 | 0 | 0 |
| | 0,08 | 59 | 0 | 7,38 | 7,38 | 0 | 0 | 69 | 7,05 | 1736 | 7,05 | 7,05 | 0 | 0 |
| 400 | 0,06 | 96 | 0 | 8,91 | 8,91 | 0 | 0 | 108 | 8,73 | 1721 | 8,66 | 8,66 | 0 | 0 |
| | 0,04 | 188 | 0 | 11,74 | 11,74 | 0 | 16 | 192 | 11,9 | 535 | 11,63 | 11,63 | 0 | 23 |
| | 0,02 | 345 | 0 | 14,72 | 14,72 | 0 | 3600 | 345 | 15,66 | 606 | 14,68 | 14,68 | 0 | 2468 |
| | 0 | 400 | 0 | 15,07 | 15,07 | 0 | 1031 | 400 | 16,15 | 2180 | 15,07 | 15,07 | 0 | 660 |
| | 0,1 | 39 | 0 | 5,94 | 5,94 | 0 | 0 | 57 | 5,74 | 753 | 5,74 | 5,74 | 0 | 0 |
| | 0,08 | 60 | 0 | 7,23 | 7,23 | 0 | 0 | 80 | 7,32 | 2608 | 7,21 | 7,21 | 0 | 0 |
| 600 | 0,06 | 100 | 0 | 9,46 | 9,46 | 0 | 2 | 117 | 9,45 | 1436 | 9,32 | 9,32 | 0 | 0 |
| | 0,04 | 202 | 0 | 12,48 | 12,48 | 0 | 20 | 207 | 12,73 | 111 | 12,33 | 12,33 | 0 | 78 |
| | 0,02 | 452 | 0 | 17,78 | 17,21 | 3,31 | 3600 | 454 | 18,27 | 3115 | 17,16 | 17,16 | 0 | 1720 |
| | 0 | 600 | 0 | 31,89 | 18,05 | 76,67 | 3600 | 600 | 19,95 | 3541 | 31,89 | 18,05 | 76,67 | 3600 |
| | 0,1 | 41 | 30 | 6,39 | 6,39 | 0 | 0 | 61 | 5,88 | 3191 | 5,88 | 5,88 | 0 | 0 |
| | 0,08 | 62 | 0 | 7,72 | 7,72 | 0 | 0 | 83 | 7,36 | 1612 | 7,34 | 7,34 | 0 | 0 |
| 800 | 0,06 | 100 | 0 | 9,48 | 9,48 | 0 | 1 | 125 | 9,5 | 2207 | 9,34 | 9,34 | 0 | 1 |
| | 0,04 | 215 | 0 | 13,15 | 13,15 | 0 | 28 | 242 | 13,74 | 362 | 13,22 | 13,22 | 0 | 42 |
| | 0,02 | 559 | 0 | 31,09 | 19,18 | 62,09 | 3600 | 561 | 20,34 | 3600 | 30,13 | 19,04 | 58,24 | 3600 |
| | 0 | 800 | 0 | 38,33 | 20,63 | 85,79 | 3600 | 800 | 23,45 | 3600 | 38,33 | 20,63 | 85,79 | 3600 |

$n$ – número de nós. $k$ – raio de vizinhança.

## 6. Conclusões

Os experimentos computacionais deste trabalho foram realizados através de três metodologias: a primeira consiste de um método de solução de duas fases, quando na primeira fase é resolvido o modelo para o $k$-DSP, tal que a solução encontrada será usada na segunda fase, que consiste na solução do TSP, quando as rotas são construídas para os nós do conjunto dominante; a segunda metodologia é a proposta de uma metaheurística GRASP, quando são aplicadas heurísticas adequadas para o DCP; e a terceira metodologia denominada híbrida é dada pela solução do TSP sobre os elementos do conjunto dominante encontrados pela metaheurística GRASP. Essas metodologias retornam portanto soluções factíveis para o DCP.

Os resultados obtidos pelos experimentos computacionais demonstraram o impacto da metodologia empregada na qualidade das soluções obtidas. Nos resultados obtidos pela metodologia de duas fases, o modelo $k$-DSP encontrou soluções ótimas para todas as instâncias. Com relação a metodologia híbrida é possível afirmar que suas rotas possuem menor custo quando comparadas às rotas da metodologia de duas fases. Isso pode ser explicado pelo fato de que a metodologia de duas fases utiliza em sua solução um conjunto dominante de cardinalidade mínima, o que não é necessariamente uma boa solução para construir uma rota de custo mínimo. Por sua vez, o método híbrido utiliza a metaheurística GRASP para explorar conjuntos dominantes que se adequam melhor ao DCP. Isso é possível a partir do mecanismo aleatório guloso inerente à metaheurística GRASP.

## Referências

Applegate, D. (2006). *The Traveling Salesman Problem: A Computational Study*. Princeton Series in Applied Mathematics. Princeton University Press.

Bondy, J. A. and Murty, U. S. R. (1976). *Graph theory with applications*, volume 290. Macmillan London.

Brasek, C. (2004). Urban utilities warm up to the idea of wireless automatic meter reading. *Computing Control Engineering Journal*, 15(6):10–14.

Chatterjee, S., Carrera, C., and Lynch, L. A. (1996). Genetic algorithms and traveling salesman problems. *European journal of operational research*, 93(3):490–510.

Chen, N., Meng, J., Rong, J., and Zhu, H. (2012). Approximation for dominating set problem with measure functions. *Computing and Informatics*, 23(1):37–49.

Croes, G. A. (1958). A method for solving traveling-salesman problems. *Operations research*, 6(6):791–812.

Goldbarg, M. and Luna, H. (2005). *Otimização combinatória e programação linear: modelos e algoritmos*. CAMPUS - RJ.

Jamil, T. (2008). Design and implementation of a wireless automatic meter reading system. In *Proceedings of the World Congress on Engineering 2008 – WCE 2008*.

Laporte, G., Gendreau, M., Potvin, J.-Y., and Semet, F. (2000). Classical and modern heuristics for the vehicle routing problem. *International transactions in operational research*, 7(4-5):285–300.

Papadimitriou, C. H. and Steiglitz, K. (1998). *Combinatorial optimization: algorithms and complexity*. Courier Corporation.

Resende, M. G. (2001). Greedy randomized adaptive search procedures (grasp). *Encyclopedia of optimization*, 2:373–382.

Usberti, F. L., França, P. M., and França, A. L. M. (2011a). Grasp with evolutionary path-relinking for the capacitated arc routing problem. *Computers and Operations Research*. doi: 10.1016/j.cor.2011.10.014.

Usberti, F. L., França, P. M., and França, A. L. M. (2011b). The open capacitated arc routing problem. *Computers and Operations Research*, 38(11):1543 – 1555.

Usberti, F. L., França, P. M., and França, A. L. M. (2008). *Roteamento de Leituristas: Um Problema NP-Difícil (in portuguese). In: XL SBPO Brazilian Symposium of Operational Research*. Annals XL SBPO, João Pessoa.