Renan Costalonga Monteiro

Clustering Search Multi-Heurística Paralelo para Resolução do Problema de Localização de Contadores de Tráfego em Redes de Transporte

Vitória, ES

2019

Renan Costalonga Monteiro

Clustering Search Multi-Heurística Paralelo para Resolução do Problema de Localização de Contadores de Tráfego em Redes de Transporte

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Informática da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do Grau de Mestre em Informática.

Universidade Federal do Espírito Santo – UFES

Centro Tecnológico

Programa de Pós-Graduação em Informática

Orientador: Geraldo Regis Mauri

Coorientadora: Maria Claudia Silva Boeres

Vitória, ES

2019

Renan Costalonga Monteiro

Clustering Search Multi-Heurística Paralelo para Resolução do Problema de Localização de Contadores de Tráfego em Redes de Transporte/ Renan Costalonga Monteiro. – Vitória, ES, 2019-

53 p. : il. (algumas color.) ; 30 cm.

Orientador: Geraldo Regis Mauri

Dissertação de Mestrado – Universidade Federal do Espírito Santo – UFES
Centro Tecnológico
Programa de Pós-Graduação em Informática, 2019.

1. Meta-Heurísticas. 2. *Clustering Search*. 3. GRASP. 4. ILS. 5. *Simulated Annealing*. 6. Paralelismo. I. Monteiro, Renan Costalonga. II. Universidade Federal do Espírito Santo. IV. *Clustering Search* Multi-Heurística Paralelo para Resolução do Problema de Localização de Contadores de Tráfego em Redes de Transporte

Renan Costalonga Monteiro

Clustering Search Multi-Heurística Paralelo para Resolução do Problema de Localização de Contadores de Tráfego em Redes de Transporte

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Informática da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do Grau de Mestre em Informática.

Trabalho aprovado. Vitória, ES, 29 de novembro de 2019:

Geraldo Regis Mauri
Orientador

Maria Claudia Silva Boeres
Coorientadora

André Renato Sales Amaral
Universidade Federal do Espírito Santo
(UFES)

Pedro Henrique González
Centro Federal de Educação Tecnológica
Celso Suckow da Fonseca (CEFET/RJ)

Vitória, ES
2019

Agradecimentos

Primeiramente agradeço a Deus pelo dom da vida e por nunca me desamparar durante todo o caminho percorrido até aqui.

Aos meus pais e meu irmão João Pedro, por todo amor, dedicação e incentivo na realização dos meus sonhos. Nada seria possível sem o apoio de vocês.

A toda a minha família que sempre esteve ao meu lado, em especial, aos meus avós Arlindo, Neli e Luís, que mesmo não estando presente em vida, estarão sempre em meu coração.

A todos os professores que passaram pela minha jornada acadêmica, em especial ao meu orientador Geraldo Regis Mauri e coorientadora Maria Claudia Silva Boeres por todo apoio na realização desse trabalho.

Por fim, a todos os pesquisadores do Brasil, que permanecem realizando trabalhos de qualidade apesar de todas as dificuldades enfrentadas diariamente.

Resumo

O Problema de Localização de Contadores de Tráfego (PLCT) tem como objetivo determinar a quantidade e a seleção de locais para a instalação de estações de contagem de tráfego de modo que uma rede de transporte seja totalmente coberta. Em geral, aparelhos contadores são utilizados para coleta de informações relacionadas ao fluxo de veículos da rede. Devido ao alto custo de instalação, busca-se minimizar o número de contadores utilizados para a cobertura total da rede. Neste trabalho, para resolução do PLCT, foram desenvolvidas duas novas versões da meta-heurística *Clustering Search* (CS), utilizando as meta-heurísticas GRASP e ILS como geradoras de soluções, ainda não aplicadas na literatura para a resolução do PLCT. Além disso, também é proposta uma versão inovadora e ainda não explorada do CS que utiliza meta-heurísticas de busca em paralelo para geração de soluções. Os métodos foram testados utilizando um conjunto de instâncias que descreve a estrutura rodoviária federal e estadual presente em cada estado brasileiro, e os resultados obtidos foram iguais ou superiores aos melhores apresentados na literatura para todas as instâncias.

Palavras-chaves: Meta-Heurísticas. *Clustering Search*. GRASP. ILS. *Simulated Annealing*. Paralelismo.

Abstract

The Traffic Counting Location Problem (TCLP) aims to determine the amount of sensors selecting locations for installing them so that a transport network is fully covered. In general, vehicle counter sensors are used to collect information related to vehicles flow in the network. Due to the high cost of installation, we have to minimize the number of sensors used to cover all the network. In this work, two new approaches of the Clustering Search (CS) metaheuristic were developed using the GRASP and ILS metaheuristics as solution generators. In addition, an innovative and unexplored approach of CS using parallel search metaheuristics for solution generation is also proposed. The methods were tested using a set of instances that describes the road structure of each Brazilian state, and the results obtained were equal or better to the best ones presented in the literature for all instances.

Keywords: Metaheuristics. Clustering Search. GRASP. ILS. Simulated Annealing. Parallelism.

Lista de Figuras

Figura 1 – Rede de Transporte Brasileira (GONZALEZ et al., 2019). 21
Figura 2 – Exemplo de rede de transporte e uma solução válida para o PLCT. . . 23
Figura 3 – Exemplo de matriz O-D. 24
Figura 4 – Exemplo de solução para o PLCT do estado do Rio de Janeiro (GONZALEZ et al., 2019). 24
Figura 5 – Fluxo de execução do algoritmo CS_{MHP}. 41

Lista de Tabelas

Tabela 1 – Características das instâncias utilizadas. 43
Tabela 2 – Parâmetros do *Clustering Search* 45
Tabela 3 – Resultados obtidos pelos algoritmos propostos. 46
Tabela 4 – Comparação com o CS_{SA} de Gonzalez et al. (2019). 48

Lista de abreviaturas e siglas

AM	Módulo de Análise
CS	*Clustering Search*
CS_{GRASP}	*Clustering Search* com *Greedy Randomized Adaptive Search Procedure*
CS_{ILS}	*Clustering Search* com *Iterated Local Search*
CS_{MHP}	*Clustering Search* Multi-Heurística Paralelo
CS_{SA}	*Clustering Search* com *Simulated Annealing*
DNIT	Departamento Nacional de Infraestrutura de Transportes
GRASP	*Greedy Randomized Adaptive Search Procedure*
IC	Procedimento Iterativo de Clusterização
ILS	*Iterated Local Search*
LS	Busca Local
O-D	Origem e destino
MH	Meta-Heurística
PFE	*Path Flow Estimator*
PLCT	Problema de Localização de Contadores de Tráfego
PNCT	Plano Nacional de Contagem de Tráfego
SA	*Simulated Annealing*
SM	Meta-Heurística de Busca

Sumário

1 INTRODUÇÃO 19
1.1 Objetivos 21
1.2 Justificativa e Contribuições 22
1.3 Estrutura da Dissertação 22

2 DESCRIÇÃO DO PROBLEMA 23

3 REVISÃO DA LITERATURA 27
3.1 *Clustering Search* 28
3.2 *Clustering Search* com SA - CS_{SA} 30
3.2.1 Solução Inicial 32
3.2.2 Similaridade H_i 33
3.2.3 Vizinhança $\psi(s)$ 33
3.2.4 Busca Local 34

4 METODOLOGIA 37
4.1 *Clustering Search* com GRASP - CS_{GRASP} 37
4.2 *Clustering Search* com ILS - CS_{ILS} 38
4.3 *Clustering Search* Multi-Heurística Paralelo - CS_{MHP} 39

5 EXPERIMENTOS E RESULTADOS 43
5.1 Descrição das Instâncias Utilizadas 43
5.2 Detalhes de Implementação 44
5.3 Escolha dos Parâmetros 44
5.4 Resultados Obtidos 44

6 CONCLUSÕES 49

REFERÊNCIAS 51

1 Introdução

O crescimento contínuo da frota de veículos nas vias brasileiras tem como consequência o surgimento de grandes congestionamentos, aumento na poluição devido à emissão de gases, desenvolvimento de doenças e diminuição da qualidade de vida da população afetada. Conhecer o fluxo de veículos das vias permite aos responsáveis pelo trânsito estimar, controlar e gerenciar as situações adversas causadas por ele.

É possível observar o comportamento do tráfego a partir de tecnologias existentes como, por exemplo, câmeras de vídeo e diversos tipos de sensores. De acordo com Gentili e Mirchandani (2005), os sensores podem ser divididos entre passivos e ativos. Sensores ativos são capazes de receber informações dos veículos como, por exemplo, a informação a respeito da rota de um caminhão de entrega. De modo contrário, os sensores passivos não necessitam receber informação para serem ativados. Câmeras de vídeo e contadores de veículos são sensores passivos, uma vez que detectam e monitoram os veículos sem receber nenhum sinal proveniente deles.

Assim, na maioria dos casos, são utilizados contadores de veículos para monitoramento de fluxo (GENTILI; MIRCHANDANI, 2012). Esses sensores podem ser instalados nas vias (arestas) ou interseções (nós) de uma rede de transporte. Caso seja instalado em uma aresta, o sensor é capaz de medir o fluxo total de veículos que passaram naquela rodovia em uma determinada janela de tempo. Quando instalado em um nó, é medido o fluxo de veículos agregado de todas as arestas adjacentes ao nó.

A partir dos dados coletados é possível derivar novas métricas relacionadas à característica do fluxo de veículos como, por exemplo: tráfego médio diário, tráfego anual, tráfego em horários de interesse e sazonalidade anual. Além disso, esse tipo de sensor pode ser usado para estimar, de modo eficiente, matrizes de origem destino (O-D). Uma matriz O-D representa a distribuição espacial entre as zonas de tráfego da área de interesse e são a principal fonte na realização de estudos no campo de gestão e controle de transportes (GARBER; HOEL, 1999).

Em geral, as matrizes O-D são estimadas com base em um estudo feito em grande escala, trabalhoso e que demanda uma grande quantidade de tempo para ser realizado. Em contrapartida, as matrizes estimadas com os dados coletados por sensores podem ser atualizadas frequentemente e com baixo custo quando comparadas ao método tradicional, representando um método eficaz na coleta de dados atualizados sobre o padrão do fluxo de veículos na rede.

Devido aos custos de instalação e manutenção, a instalação dos equipamentos de contagem em todos as arestas (ou nós) da rede não é economicamente viável. Assim, surge

o Problema de Localização de Contadores de Tráfego (PLCT) tratado neste trabalho, que resume-se na seleção de locais para a instalação dos equipamentos de modo que toda a rede de transporte seja coberta, minimizando o número de contadores utilizados. Segundo Chen et al. (2007), trata-se de um problema NP-Completo para a maioria das formulações, e uma solução poderia ser alcançada caso fosse possível identificar todos os caminhos que ligam cada par presente na matriz O-D. Entretanto, essa abordagem não é viável para instâncias do porte de redes reais, uma vez que o número de caminhos aumenta conforme a dimensão da rede analisada.

Por conta da complexidade em encontrar a localização ótima para os contadores, geralmente apenas uma parte da rede é observada. Na ocorrência de grandes congestionamentos, o trânsito tende a fluir das principais vias (geralmente monitoradas) para as vias secundárias não observadas, criando um novo padrão de fluxo. Portanto, um problema enfrentado pelas autoridades de trânsito é a falta de conhecimento sobre as características de trânsito em toda a rede de transporte.

Com início na década de 70, atualmente no Brasil é desenvolvido, sob a jurisdição do Departamento Nacional de Infra-Estrutura de Transportes (DNIT), o Plano Nacional de Contagem de Tráfego (PNCT) (DNIT, 1970). O PNCT tem como objetivo a coleta de métricas relacionadas ao fluxo de veículos utilizando estações de contagem de tráfego nas principais rodovias federais do território brasileiro.

Em 2018, foi divulgada pelo DNIT a informação que, atualmente, o Brasil conta com 57,2 mil quilômetros de rodovias federais pavimentadas sob sua jurisdição. A Figura 1 apresenta a rede de transporte nacional. O PNCT procura cobrir os trechos mais representativos da malha rodoviária presente em cada estado brasileiro. Esse estudo é, sem dúvidas, de suma importância na gestão das rodovias, pois seus resultados são a base para estudos de planejamento da infraestrutura do sistema rodoviário. A partir desses dados é possível realizar diversas ações como: programar necessidades e prioridades de melhorias no sistema rodoviário; medir a demanda atual de serviços por via rodoviária; justificar e planejar o policiamento, etc.

Para a realização do PNCT é necessário alocar os contadores na rede de transporte para a criação de uma matriz O-D que representará todo o país, sendo as origens e destinos representadas pelos municípios. Entretanto, a rede brasileira possui mais de 24 mil arestas (estradas) e mais de 20 mil nós (municípios ou interseção entre estradas). Portanto, é importante estimar o número mínimo de contadores a serem alocados de modo que todas as rotas entre pares de municípios tenha, pelo menos, um contador presente.

Devido ao tamanho da rede nacional, estimar a matriz O-D que representa todo o país é muito custoso. Por esse motivo, neste trabalho, o problema foi dividido considerando a divisão política dos estados brasileiros. Nesse caso, cada estado possui sua rede de transporte, ou seja, as rodovias contidas em seu território e a matriz O-D formada pelos

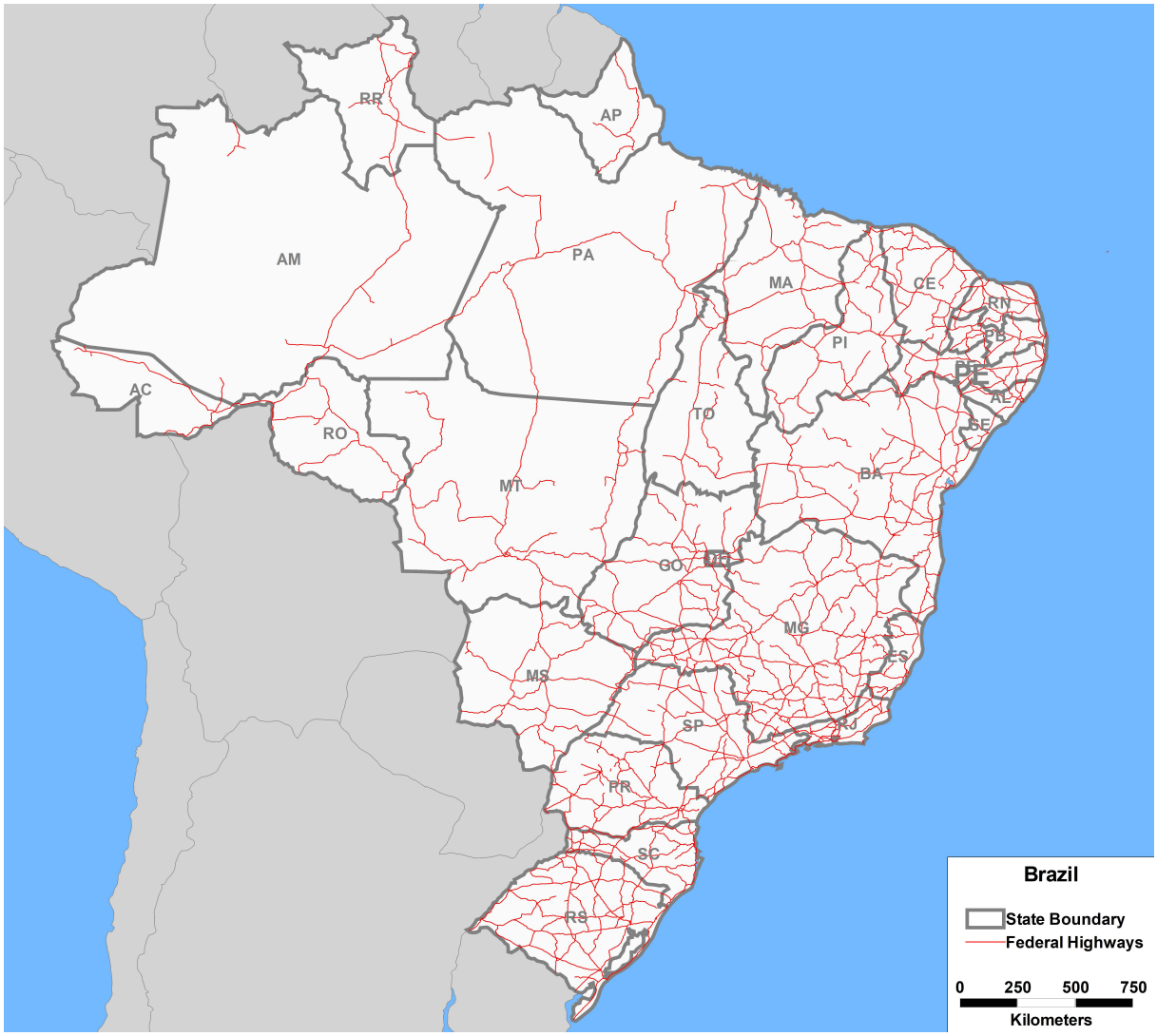

Figura 1 – Rede de Transporte Brasileira (GONZALEZ et al., 2019).

municípios do estado.

1.1 Objetivos

O objetivo geral deste trabalho consiste em investigar a aplicação de diferentes variações da meta-heurística *Clustering Search* (CS) para a resolução do PLCT considerando a rede de transporte brasileira. Como objetivos específicos, tem-se:

- implementar um CS utilizando a meta-heurística *Greedy Randomized Adaptive Search Procedure* (GRASP) como geradora de soluções;
- implementar um CS utilizando a meta-heurística *Iterated Local Seach* (ILS) como geradora de soluções;
- implementar um CS paralelo utilizando, de forma integrada, as meta-heurísticas

Simulated Annealing (SA), GRASP e ILS como geradoras de soluções;

- analisar o desempenho das diferentes variações do CS;
- comparar os resultados obtidos com os melhores apresentados na literatura até então.

1.2 Justificativa e Contribuições

Devido à complexidade do PLCT, existe a necessidade de se propor algoritmos eficientes que apresentem soluções satisfatórias e resolvam o problema em tempo computacional viável, independente da dimensão da rede. Este fato justifica a busca por novas alternativas que sejam eficientes para resolução de casos reais do PLCT.

A escolha do CS como base para esta pesquisa se deve ao fato do mesmo ter apresentado, em Mauri et al. (2017), bons resultados para resolução do PLCT considerando a rede de transporte brasileira. Mauri et al. (2017) utilizaram a meta-heurística SA como geradora de soluções para o CS.

Já a escolha das meta-heurísticas GRASP e ILS é devida ao fato de que as mesmas apresentarem bons resultados quando utilizadas com o CS (MELO; BARROS JUNIOR; MAURI, 2013; MELO; BARROS JUNIOR; MAURI, 2014) para resolução de outros problemas combinatoriais, além de ainda não terem sido exploradas para resolução do PLCT.

Assim, além desta pesquisa contribuir diretamente com a resolução de casos reais do PLCT, considerando a rede de transporte brasileira, são investigadas diferentes variações já conhecidas do CS ainda não exploradas para sua resolução. Por fim, é proposta também uma abordagem paralela e multi-heurística do CS, inédita na literatura até então, que pode ser utilizada em trabalhos futuros para resolução de outros problemas combinatoriais.

1.3 Estrutura da Dissertação

O restante desta dissertação está estruturado da seguinte maneira: O Capítulo 2 apresenta a descrição do PLCT. Uma revisão da literatura relacionada ao PLCT e ao CS é apresentada no Capítulo 3. No Capítulo 4 são apresentados os métodos propostos para resolução do PLCT. Os experimentos computacionais são apresentados no Capítulo 5, e a conclusão do trabalho é descrita no Capítulo 6.

2 Descrição do Problema

O Problema de Localização de Contadores de Tráfego (PLCT) pode ser modelado formalmente de acordo com Yang, Yang e Gan (2006). Nessa formulação, uma rede de transporte é representada como um grafo não orientado $G = (N, A)$, em que N representa o conjunto de nós presentes na rede e A o conjunto de arestas.

A Figura 2 exibe um exemplo de uma rede de transporte com 9 nós e 12 arestas, e uma possível solução para o PLCT. Nessa rede, as arestas são as rodovias e os nós representam municípios ou interseções (cruzamentos) entre diferentes vias. O seu fluxo de tráfego é gerado através de viagens entre os diferentes municípios, ou seja, um município pode ser a origem ou destino (O-D) de uma viagem.

Considerando M ($M \subseteq N$) o conjunto de municípios da rede, sendo $M = \{n_1, n_6, n_9\}$ (nós destacados em cinza) para a Figura 2, tem-se seis diferentes pares O-D: $w_1 = (n_1, n_6)$, $w_2 = (n_1, n_9)$, $w_3 = (n_6, n_9)$, $w_4 = (n_6, n_1)$, $w_5 = (n_9, n_1)$, $w_6 = (n_9, n_6)$. Considerando ainda que a rede é representada por um grafo não orientado, os pares w_4, w_5 e w_6 são redundantes, ou seja, seriam os caminhos "inversos" dos demais. Assim, o número total de pares pode ser calculado como $|W| = \dfrac{|M| \times (|M| - 1)}{2}$, sendo $W = \{w_1, w_2, w_3\}$ para esse exemplo.

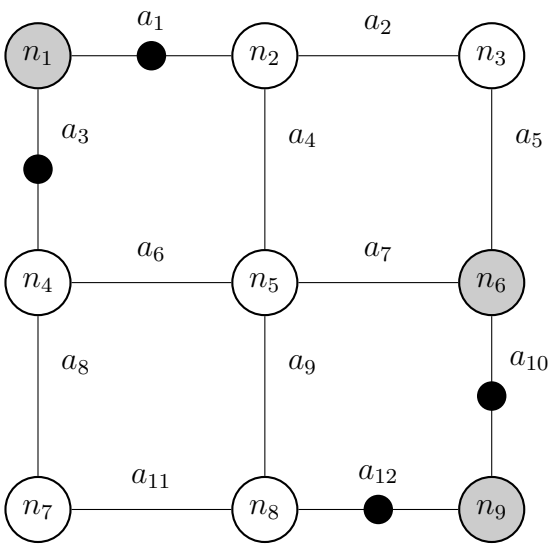

Figura 2 – Exemplo de rede de transporte e uma solução válida para o PLCT.

Uma solução para o PLCT é um subconjunto $S \subseteq A$ tal que a remoção desse conjunto desconecta todos os caminhos entre cada par O-D. Na Figura 2, $S = \{a_1, a_3, a_{10}, a_{12}\}$ (destacadas com um ponto preto) seria uma solução viável para o PLCT. Caso os elementos de S sejam removidos da rede, não será possível encontrar um caminho que ligue cada par O-D. Logo, S se resume como os locais escolhidos para a instalação dos contadores de

veículos, uma vez que todos os caminhos entre os pares possuem pelo menos uma estação de contagem, e $|S|$ indica a quantidade total de contadores a serem utilizados.

Diferente tipos de informações podem ser coletadas para o preenchimento de uma matriz O-D, como por exemplo: número total de veículos, número de veículos por tipo (motocicleta, carro, caminhão, etc.), peso dos veículos, entre outros. Entretanto, o tipo de informação está associado diretamente na escolha do aparelho contador a ser instalado, podendo ser abstraído para fins de modelagem matemática e otimização. A partir do exemplo anterior, uma matriz O-D pode ser definida de acordo com a Figura 3

$$\text{Matriz O-D} = \begin{array}{c} \\ n_1 \\ n_6 \\ n_9 \end{array} \begin{array}{c} \begin{array}{ccc} n_1 & n_6 & n_9 \end{array} \\ \begin{pmatrix} - & \mu_1 & \mu_2 \\ \mu_3 & - & \mu_4 \\ \mu_5 & \mu_6 & - \end{pmatrix} \end{array}$$

Figura 3 – Exemplo de matriz O-D.

Como pode ser observado na Figura 3, a matriz O-D possui dimensão $|M| \times |M|$, ou seja, de acordo com o número de municípios da rede. Já os valores dessa matriz $(\mu_1 \dots \mu_6)$ são aqueles obtidos pelos aparelhos contadores, de acordo com o contexto utilizado. A Figura 4 exibe um exemplo da alocação dos contadores para observar todas as rotas entre os municípios contidos na instância do estado brasileiro do Rio de Janeiro.

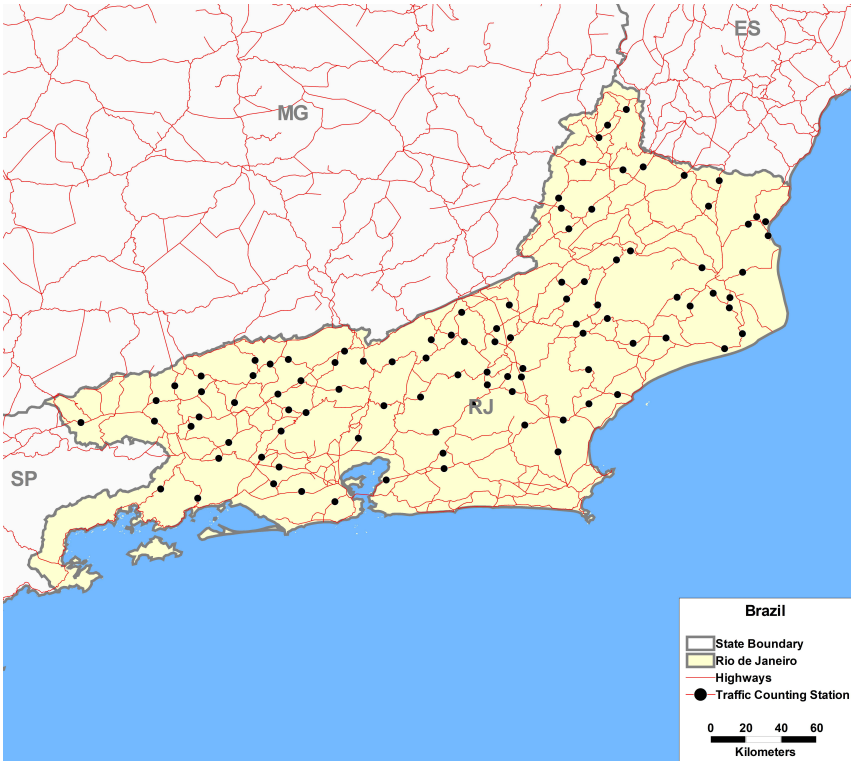

Figura 4 – Exemplo de solução para o PLCT do estado do Rio de Janeiro (GONZALEZ et al., 2019).

O modelo matemático para o PLCT utilizado nesse trabalho tem como função objetivo definir o número mínimo e a localização dos contadores de modo que todos os pares O-D sejam separados (cobertos) por no mínimo um contador. Logo, o PLCT pode ser formulado segundo Yang, Yang e Gan (2006), da seguinte maneira:

$$Minimizar \quad z = \sum_{a \in A} x_a \tag{2.1}$$

Sujeito a:

$$\sum_{a \in A} \delta_{ra}^{w} x_a \geq 1 \qquad r \in R_w, w \in W \tag{2.2}$$

$$x_a \in \{0, 1\} \qquad a \in A \tag{2.3}$$

Cada par $w \in W$ é composto por nós $n_o \in M$ e $n_d \in M$, sendo que n_o representa a origem e n_d o destino do par. $x_a \in \{0, 1\}$ é uma variável binária que indica a presença ($x_a = 1$) ou ausência ($x_a = 0$) de um contador na aresta $a \in A$. A distância do menor caminho entre o par O-D $w \in W$ é representada por $u_w(x)$ e, quando $u_w(x) > 0$, pelo menos um contador será instalado nesse caminho, indicando a cobertura (ou separação) do par. R_w representa o conjunto de todos os caminhos que ligam um par O-D w. Por fim, δ_{ra}^{w} é uma variável binária que é 1 se a aresta $a \in A$ está contida no caminho $r \in R_w$ e 0 caso contrário. A função objetivo (2.1) minimiza o número de contadores utilizados. As restrições (2.2) asseguram que todos os pares O-D sejam cobertos por pelo menos um contador, enquanto o domínio das variáveis é definido em (2.3).

3 Revisão da Literatura

Com início na década de 80, encontram-se na literatura trabalhos que apresentam diferentes técnicas para representar e resolver o Problema de Localização de Contadores de Tráfego (PLCT). Nos próximos parágrafos são destacados alguns trabalhos presentes na literatura, ordenados cronologicamente, de modo a destacar as diferentes abordagens adotadas na resolução do problema.

Zuylen e Willumsen (1980) propuseram dois modelos com base na minimização de informação e o princípio da maximização de entropia para estimar matrizes O-D a partir de contadores de veículos e um algoritmo para resolvê-los. Ambos os modelos utilizaram dados coletados pelos sensores combinados com informações coletadas a priori para estimar e atualizar a matriz O-D mais representativa à rede observada.

Yang e Zhou (1998) determinaram a quantidade e a alocação dos sensores a partir de um padrão de distribuição O-D já conhecido. Foram desenvolvidos algoritmos de programação inteira e heurísticas para a resolução do problema, respeitando as regras definidas.

Maher, Zhang e Vliet (2001) utilizaram otimização em dois níveis para estimar a matriz de fluxo e otimização de sinal de trânsito em redes de transporte congestionadas. Os autores apresentam um procedimento de vizinhança que pode ser aplicado a ambos os problemas com pequenas modificações, e realizaram testes em redes de diferentes tamanhos e topologias.

Path Flow Estimator (PFE) é um observador proposto na literatura utilizado para estimar o volume de fluxo e tempo de viagem dos caminhos presentes em uma rede de transporte. Esses volumes podem ser agregados para derivar matrizes O-D. Chen, Chootinan e Recker (2005) analisaram a capacidade do PFE em estimar o volume total do fluxo de uma rede conhecida, assim como demandas individuais entre origem e destino específicos. Foram disponibilizados diferentes exemplos para verificar a relação entre o total alocado e a localização dos contadores na qualidade das estimativas. Bell (1983) e Chen, Chootinan e Recker (2005) apresentaram alternativas de utilização de contadores de veículos para melhorar qualidade da estimativa de matrizes O-D.

Yang, Gan e Tang (2001) propuseram dois modelos matemáticos para minimizar o número de contadores utilizados. Foi realizado um estudo pioneiro utilizando linhas de contorno e linhas de corte, visando coletar o máximo de informação de tráfego possível e reduzindo recursos. Dado um número fixo de contadores, utilizou-se modelos de programação inteira para selecionar sua localização ótima a fim de maximizar os pares O-D cobertos. Além disso, os autores propuseram uma heurística para determinar o mínimo

necessário para a cobertura total da rede.

Yang, Yang e Gan (2006) exploraram o problema através de relaxação linear e dualidade. Os autores propuseram uma combinação do procedimento para geração de colunas de menor caminho entre os nós e o método *branch and bound*. Os resultados alcançados foram comparados com um Algoritmo Genético.

Chen et al. (2007) desenvolveram um Algoritmo Genético para a seleção de novos locais de instalação de estações de contagem para melhorar a estimativa de uma matriz O-D existente. Após a seleção, é utilizada uma modificação do PFE para a geração da nova matriz.

Gonzalez et al. (2016) apresentaram as heurísticas C1, C2 e C3 para determinar o total e a alocação dos contadores de tráfego na rede, de modo que todos os pares O-D sejam cobertos por pelo menos uma estação de contagem. As heurísticas foram testadas utilizando um novo conjunto de instâncias, descrevendo a configuração real das rodovias presentes em todos os estados brasileiros. Mauri et al. (2017) utilizaram o mesmo conjunto de instâncias apresentado por Gonzalez et al. (2016), e aplicaram a meta-heurística *Clustering Search* (CS) com *Simulated Annealing* (SA) para geração de soluções.

Por fim, Gonzalez et al. (2019) aplicaram uma versão recalibrada da meta-heurística *Clustering Search* (CS) com *Simulated Annealing* (SA) proposta por Mauri et al. (2017), e propuseram um algoritmo *Branch-and-Cut* para resolução das mesmas instâncias consideradas por Gonzalez et al. (2016) e Mauri et al. (2017). Os resultados foram comparados com o AG descrito por Chen et al. (2007), apresentando os melhores resultados conhecidos para as instâncias brasileiras até então.

Na próxima seção é realizada uma breve revisão sobre a meta-heurística *Clustering Search*, seguida pela descrição do algoritmo *Clustering Search* com *Simulated Annealing* - CS_{SA} desenvolvido por Gonzalez et al. (2019), detalhando seu componente para a criação de uma solução inicial, busca-local, vizinhança e similaridade entre soluções, utilizados também nas versões propostas nesse trabalho.

3.1 *Clustering Search*

A meta-heurística *Clustering Search* (CS) é um procedimento híbrido que combina o uso de meta-heurísticas e heurísticas aplicando *clusterização* com objetivo de encontrar e explorar regiões consideradas promissoras no espaço de busca. O CS tem obtido bons resultados para diversos problemas de otimização combinatória, como pode ser visto em Ribeiro, Laporte e Mauri (2012), Oliveira, Mauri e Lorena (2012), Costa, Fiedler e Mauri (2013), Rabello et al. (2014), Rodrigues et al. (2014), Rosa et al. (2016) e Rosa et al. (2017). Uma descrição detalhada do CS é apresentada por Oliveira, Chaves e Lorena (2013).

Durante sua execução, o CS tenta localizar regiões promissoras no espaço de busca, ou seja, regiões que se encontram as soluções de boa qualidade para o problema, separando-a em *clusters*. Um *cluster* i é definido como uma tupla (c_i, v_i, r_i) em que c_i representa o centro do *cluster*, ou seja, a melhor solução encontrada até o momento na região i, seu volume v_i indica o total de soluções associadas ao *cluster* i, e seu índice de ineficácia r_i, que representa o total de vezes que não foi obtida uma solução de melhor qualidade após a execução do procedimento de busca local em seu centro c_i.

Conforme Oliveira, Chaves e Lorena (2013), o CS é uma meta-heurística híbrida que pode ser separada em quatro componentes individuais com diferentes atribuições: Meta-heurística de Busca (SM); Procedimento Iterativo de Clusterização (IC); Módulo de Análise (AM) e Busca Local (LS). O Algoritmo 1 apresenta o pseudo-código do CS utilizado como base neste trabalho. Nos trabalhos citados no início desta seção, pode-se encontrar pseudo-códigos do CS com algumas variações, como: critérios de parada diferentes, centro dos *clusters* criados no início do algoritmo, etc.

Algoritmo 1: Pseudo-código do CS.

```
   Entrada: γ, λ, r_max, e tempo_max
   Saída: s_melhor
1  início
2      clt ← 0; r_i ← 0, v_i ← 0 ∀ i = 1, ..., γ;
3      enquanto tempo < tempo_max faça
4          s ← SM();
5          se clt < γ então
6              clt ← clt + 1; v_clt ← v_clt + 1; c_clt ← s;
7          senão
8              i ← argmin_{i=1,...,γ}{H_i}; v_i ← v_i + 1; c_i ← Melhor(c_i, s);
9              se v_i = λ então
10                 v_i ← 1;
11                 se r_i = r_max então
12                     r_i ← 0; c_i ← Perturbação(c_i);
13                 senão
14                     BuscaLocal(c_i);
15                     se c_i melhorou então
16                         r_i ← 0;
17                     senão
18                         r_i ← r_i + 1;
19                     fim
20                 fim
21                 s_melhor ← Melhor(s_melhor, c_i);
22             fim
23         fim
24     fim
25 fim
```

Na linha 2 são inicializadas as estruturas responsáveis por armazenar os dados dos γ *clusters*. Na linha seguinte, inicia-se a parte iterativa do método, onde as componentes do CS são executadas por um tempo limite $tempo_{max}$. O Componente SM (linha 4) é o procedimento responsável por gerar soluções. O algoritmo é executado de forma

independente aos outros componentes fornecendo novas soluções a serem associadas a um *cluster* em cada iteração. Assim, diversas meta-heurísticas e/ou heurísticas podem ser utilizadas nesse componente do CS.

No componente IC, as soluções similares são agrupadas em *clusters*, mantendo seu centro representativo. Para otimizar os recursos computacionais, esse componente é ativado a cada nova solução gerada pelo SM. O número máximo de *clusters* γ é definido a priori para evitar a criação ilimitada, assim como a métrica de similaridade entre as soluções. Caso o número máximo de *clusters* ainda não tenha sido atingido (linha 5), aloca-se a solução em um novo *cluster* (linha 6), e em caso contrário, a solução será associada ao *cluster* i com maior similaridade H_i e substituirá a solução corrente (linha 8) se possuir melhor qualidade.

O componente de análise AM é executado quando o volume v_i de algum *cluster* i atinge um limite superior λ (linha 9), definido anteriormente como parâmetro do CS. Em cada execução, o centro do *cluster* é analisado com o objetivo de acelerar o processo de convergência naquela região de busca. Caso r_i ultrapasse um *threshold* r_{max} (linha 11), aplica-se um algoritmo de perturbação em c_i (linha 12) e, caso contrário, aplica-se a busca local LS. O componente LS tem como entrada a solução contida em c_i, retornando possivelmente uma solução melhorada ao centro do *cluster* (linha 14). Por fim, o índice de ineficácia do *cluster* r_i é resetado ou incrementado, de acordo com o retorno de LS. (linhas 15 a 19).

3.2 *Clustering Search* com SA - CS_{SA}

Gonzalez et al. (2019) implementaram um *Clustering Search* utilizando um *Simulated Annealing* (KIRKPATRICK; GELATT; VECCHI, 1983) como componente SM para resolução do PLCT. De uma forma geral, a meta-heurística *Simulated Annealing* é uma metáfora ao processo metalúrgico de *Annealing*, utilizado para fundir metais, no qual estes são aquecidos à elevadas temperaturas e em seguida resfriados lentamente até que o material se solidifique, de modo que o produto final permaneça uniforme.

O Algoritmo 2 apresenta o pseudo-código do CS_{SA} adaptado do trabalho de Gonzalez et al. (2019). Os parâmetros de execução do SA são: as temperaturas inicial T_0 e final T_f, o número de vizinhos gerados a cada temperatura sa_{max} e a taxa de resfriamento da temperatura α. Além desses, tem-se os parâmetros necessários para o CS: γ, λ, r_{max} e $tempo_{max}$, descritos na seção anterior.

São inicializadas as variáveis relacionadas aos *clusters* (linha 2) e gerada a solução inicial, que até o momento é a melhor solução encontrada pelo procedimento (linha 3). De forma análoga ao processo de *Annealing*, a cada temperatura são gerados sa_{max} vizinhos (linha 8) utilizando o procedimento de vizinhança, descrito em detalhes na Seção 3.2.3.

Algoritmo 2: CS_{SA} para a resolução do PLCT. Adaptado de Gonzalez et al. (2019).

Entrada: T_0, T_f, sa_{max}, α, γ, λ, r_{max} e $tempo_{max}$
Saída: s_{melhor}

```
1  início
2      clt ← 0; r_i ← 0, v_i ← 0 ∀ i = 1, ..., γ;
3      s ← SoluçãoInicial;
4      s_melhor ← s;
5      enquanto tempo < tempo_max faça
6          T ← T_0;
7          enquanto T > T_f faça
8              para i ← 1...sa_max faça
9                  s' ← ψ(s);
10                 se f(s) > f(s') então
11                     s ← s';
12                     se f(s_melhor) > f(s') então
13                         s_melhor ← s';
14                     fim
15                 senão
16                     Com probabilidade de e^(−(f(s') − f(s))/T)  s ← s';
17                 fim
18             fim
19             T ← αT;
20             se clt < γ então
21                 clt ← clt + 1; v_clt ← v_clt + 1; c_clt ← s_melhor;
22             senão
23                 i ← argmin_{i=1,...,γ}{H_i}; v_i ← v_i + 1; c_i ← Melhor(c_i, s_melhor);
24                 se v_i = λ então
25                     v_i ← 1;
26                     se r_i = r_max então
27                         r_i ← 0; c_i ← Perturbação(c_i);
28                     senão
29                         BuscaLocal(c_i);
30                         se c_i melhorou então
31                             r_i ← 0;
32                         senão
33                             r_i ← r_i + 1;
34                         fim
35                     fim
36                     s_melhor ← Melhor(s_melhor, c_i);
37                 fim
38             fim
39         fim
40     fim
41 fim
```

Caso a solução vizinha s' seja melhor que a solução corrente s, esta é atualizada, assim como a melhor solução encontrada até o momento (linhas 10 a 14). No caso em que s' tem pior qualidade quando comparada com s, s' pode ser aceita com uma probabilidade $e^{-\frac{f(s') - f(s)}{T}}$ (linha 16). Quanto maior o valor de $f(s') - f(s)$ e menor a temperatura, menores serão a chances de aceitação da nova solução. O comportamento inicial do algoritmo é de aceitar grande amplitude de soluções quando a temperatura está alta e, à

medida que a temperatura cai, o algoritmo começa a convergir para uma solução ótima local. Em seguida, é realizado o resfriamento na temperatura (linha 19).

A cada resfriamento na temperatura, são iniciadas as componentes do CS. Caso o número máximo de *clusters* ainda não tenha sido atingido, aloca-se a solução s proveniente do SA em um novo centro de *cluster* (linha 21). Na situação em que todos os λ *clusters* já estejam utilizados, inicia-se o componente IC do CS, e s é alocada no centro do *cluster* com maior similaridade H_i (linha 23), descrita na Seção 3.2.2, caso seja melhor que a solução atual presente no centro do *cluster*. Por fim, é executado o componente de análise, em que a região de busca do *cluster* escolhido anteriormente é analisada (linhas 24 a 35). Caso a região seja considerada promissora, é realizada uma perturbação em seu centro e, caso contrário, é realizada um procedimento de Busca Local. A iteração é finalizada com a verificação de uma possível melhora após a execução dos procedimentos de perturbação e busca local. As etapas do CS_{SA} citadas são descritas com mais detalhes a seguir.

3.2.1 Solução Inicial

Uma solução é representada por um vetor binário $x_a \in \{0, 1\}$, que indica a presença de um contador ($x_a = 1$) ou não ($x_a = 0$) na rodovia $a \in A$. A solução inicial representa a solução base a ser manipulada durante a execução dos algoritmos propostos.

A ideia principal do procedimento de criação de uma solução inicial é a separação de todos os pares O-D da rede por meio da utilização contínua do algoritmo de corte mínimo (AHUJA; MAGNANTI; ORLIN, 1993). Um corte mínimo em uma rede de transporte é o menor número de arestas que, se removidas, desconectam os pares O-D.

Essa estratégia foi definida e denominada C3 no trabalho de Gonzalez et al. (2016), e é capaz de produzir soluções viáveis de boa qualidade em baixo tempo computacional. O Algoritmo 3 exibe o pseudo-código da C3.

Como entrada o algoritmo recebe a rede de transporte G, composta pelos nós (N) e arestas (A) e o conjunto de pares O-D (W) presentes na rede. No inicio da execução (linha 2), inicia-se o conjunto solução, ou seja, o conjunto de arestas que irão compor o corte mínimo. Em seguida, inicia-se a fase iterativa do algoritmo e, enquanto existir algum par OD $w \in W$ conectado, escolhe-se o primeiro par k do conjunto W (linha 3). Uma vez escolhido o par $k = (n_o, n_d)$ ($n_o,\ n_d \in M$), é aplicado um algoritmo de corte mínimo para determinar o corte mínimo que desconecte a origem n_o e o destino n_d de k (linha 5). O corte mínimo recebe como entrada G e k, e retorna o conjunto Q com os nós pertencentes a um dos subgrafos resultantes do corte e o conjunto de arestas E que compõem o corte. Após o corte, é verificado se algum outro par foi indiretamente desconectado (linhas 8 a 10). Os pares desconectados são armazenados no conjunto temporário H e removidos do conjunto W (linha 11). Por fim, as arestas selecionadas durante o corte são adicionadas a

Algoritmo 3: Heurística C3. Adaptado de Gonzalez et al. (2019).

```
   Entrada: G(N, A), W
   Saída: s
1  início
2      s ← {}
3      enquanto W ≠ {} faça
4          k ← W[0]
5          {Q, E} ← CorteMinimo(G, k)
6          Q̃ ← V \ Q
7          H ← {}
8          para k ∈ W | o_k ∈ Q e d_k ∈ Q̃ faça
9              H ← H ∪ {k}
10         fim
11         W ← W \ H
12         s ← s ∪ E
13     fim
14 fim
```

solução s (linha 12).

3.2.2 Similaridade H_i

A similaridade entre duas soluções é determinada pela distância de Hamming (H_i) (HAMMING, 1950), que é obtida pela diferença de bits entre duas soluções s_1 e s_2 quaisquer. Por exemplo, para as soluções s_1 e s_2 abaixo, $H_i = 3$. Esse critério é utilizado na associação de uma solução s ao centro do *cluster* que possui maior similaridade (linha 23 do Algoritmo 2), ou seja, menor distância de Hamming.

$$s_1 = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 \end{bmatrix}$$
$$s_2 = \begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 0 \end{bmatrix}$$

3.2.3 Vizinhança $\psi(s)$

O procedimento de definição do movimento para geração de uma solução vizinhança recebe uma solução s e retorna uma solução $s' \in \psi(s)$. A ideia principal do método proposto por Gonzalez et al. (2019) consiste em selecionar aleatoriamente uma aresta a da rede e trocar seu valor binário, ou seja, no caso em que a possuir um contador instalado ($x_a = 1$), o contador será removido ($x_a = 0$) e, caso contrário, um contador será instalado em a. Em seguida, caso a solução se torne inviável, o método se concentra em torna-la viável novamente, de modo que o total de contadores utilizado não seja alterado.

No Algoritmo 4, inicialmente é escolhida uma aresta aleatória $a \in A$ (linha 2). Caso a não possua um contador instalado (linha 3), um contador é instalado e, por $|A|$ iterações é selecionada uma aresta b aleatoriamente, tal que b seja diferente de a e possua

um contador instalado ($x_b = 1$). O contador de b é retirado e a viabilidade da função é verificada (linha 8). Se algum par O-D ficou descoberto (solução inviável) após a remoção, o contador é alocado novamente em b (linha 9). Do contrário, caso a já possua um contador instalado (linha 13), esse é removido e enquanto a solução permanecer inviável, uma aresta b que não possua um contador é selecionada e um contador é instalado (linha 17). Caso o total pares O-D cobertos não aumente, a operação é desfeita.

Algoritmo 4: Vizinhança $\psi(s)$. Adaptado de Gonzalez et al. (2019).

```
   Entrada: s
   Saída: s'
 1 início
 2     a ← aleatório |A|;
 3     se x_a = 0 então
 4         x_a ← 1; i ← 0;
 5         enquanto i < |A| faça
 6             b ← aleatório |A| \ b ≠ a e x_b = 1;
 7             x_b = 0;
 8             se s é inviável então
 9                 x_b ← 1;
10             fim
11             i ← i + 1;
12         fim
13     senão
14         x_a ← 0;
15         enquanto s for inviável faça
16             b ← aleatório |A| \ x_b = 0;
17             x_b ← 1;
18             se o número de pares cobertos não aumentou então
19                 x_b ← 0;
20             fim
21         fim
22     fim
23 fim
```

3.2.4 Busca Local

A busca local tem como objetivo melhorar a solução construída inicialmente. Para atingir esse objetivo, é necessária a definição de estratégias para exploração da vizinhança de uma solução. O pseudo-código é apresentado no Algoritmo 5, que tem como foco a seleção de duas arestas aleatórias e distintas $a, b \in A$ (linhas 4 e 5) e a troca de seus valores na solução s de modo que essa continue viável, ou seja, caso a ou b possuam um contador instalado, ele é removido e, caso contrário é instalado (linhas 6 e 7). Caso a solução se torne inviável após a troca, a operação é desfeita (linhas 8 a 11). O processo de escolha é repetido enquanto não for encontrada uma troca que não prejudique a viabilidade de s. Logo após, por $|A|$ iterações, seleciona-se aleatoriamente uma nova aresta c diferente de a e b que possua um contador instalado, e esse é removido (linha 15). Se após a troca, a solução ficar inviável, o contador é instalado novamente em c (linha 18). O procedimento

é executado enquanto houver melhora na solução.

Algoritmo 5: Busca local. Adaptado de Gonzalez et al. (2019).

```
início
    repita
        repita
            a ← aleatório |A|;
            b ← aleatório |A| | x_b ≠ x_a;
            x_a ← 1 − x_a;
            x_b ← 1 − x_b;
            se solução ficar inviável então
                x_a ← 1 − x_a;
                x_b ← 1 − x_b;
            fim
        até encontrar uma solução viável;
        i ← 0;
        enquanto i < |A| faça
            c ← aleatório |A| | c ≠ b e c ≠ a e x_c = 1;
            x_c ← 0;
            se solução ficar inviável então
                x_c ← 1;
            fim
            i ← i + 1;
        fim
    até solução não melhorar;
fim
```

4 Metodologia

Neste trabalho, três novas versões do CS foram implementadas, tomando como base o CS_{SA} apresentado por Gonzalez et al. (2019), que utiliza a meta-heurística *Simulated Annealing* (SA) como geradora de soluções. A primeira versão (CS_{GRASP}) substitui o SA pela meta-heurística *Greedy Randomized Adaptive Search Procedure* (GRASP), enquanto na segunda (CS_{ILS}), o SA é substituído pela meta-heurística *Iterated Local Search* (ILS). Já a terceira utiliza as três meta-heurísticas (SA, GRASP e ILS) em paralelo para geração de soluções para o CS. As versões do CS para resolução do Problema de Localização de Contadores de Tráfego (PLCT) propostas neste trabalho são apresentadas em detalhes a seguir.

4.1 *Clustering Search* com GRASP - CS_{GRASP}

O Algoritmo GRASP foi apresentado por Feo e Resende (1989) como uma alternativa de solução para o Problema de Cobertura de Conjuntos. O procedimento iterativo é composto por iterações independentes, divididas em duas etapas. A Fase de Construção é a primeira etapa do algoritmo e combina técnicas gulosas e aleatórias na produção de um conjunto de soluções iniciais viáveis para a fase de busca local. A partir de uma solução s obtida na fase de construção, a fase de busca local procura uma solução de melhor qualidade s' em uma vizinhança de s. O Algoritmo 6 apresenta o pseudo-código do CS implementado para resolução do PLCT utilizando o GRASP como gerador de soluções.

O método recebe como entrada os parâmetros do CS, já descritos na Seção 3.1, além do número máximo de iterações do GRASP ($grasp_{max}$). Os $clusters$ são inicializados e o GRASP é executado (linhas 4 a 8). A melhor solução obtida pelo GRASP é enviada ao CS, cujo funcionamento é similar ao apresentado no capítulo anterior. Os algoritmos utilizados para a criação da solução inicial e no procedimento de busca local são aqueles apresentados nas Seções 3.2.1 e 3.2.4.

Tradicionalmente o procedimento de solução inicial implementado no GRASP recebe o conjunto de elementos que irão compor a solução LC e um parâmetro α ($0 \leq \alpha \leq 1$). A cada iteração dessa fase, o custo para a inserção de cada elemento ainda disponível é avaliado separadamente e, logo após, é formado um novo conjunto LRC com os α elementos que possuem menor custo de inserção e o próximo elemento a ser alocado é escolhido aleatoriamente entre eles. Conforme o valor de α tende a zero, apenas os elementos com baixo custo de inserção irão compor LRC. Esse parâmetro não foi considerado na implementação do CS_{GRASP}, pois a heurística C3 (Algoritmo 3) utilizada na fase construtiva tem características aleatoriedade e gulosidade, simulando o comportamento de

Algoritmo 6: CS_{GRASP} para a resolução do PLCT.

Entrada: $grasp_{max}$, γ, λ, r_{max} e $tempo_{max}$
Saída: s_{melhor}
início
  $clt \leftarrow 0$; $r_i \leftarrow 0, v_i \leftarrow 0 \ \forall \ i = 1, ..., \gamma$;
  enquanto $tempo < tempo_{max}$ **faça**
    para $i \leftarrow 1...grasp_{max}$ **faça**
      $s \leftarrow$ SoluçãoInicial;
      BuscaLocal(s);
      $s_{melhor} \leftarrow$ Melhor(s, s_{melhor});
    fim
    se $clt < \gamma$ **então**
      $clt \leftarrow clt + 1$; $v_{clt} \leftarrow v_{clt} + 1$; $c_{clt} \leftarrow s_{melhor}$;
    senão
      $i \leftarrow argmin_{i=1,...,\gamma}\{H_i\}$; $v_i \leftarrow v_i + 1$; $c_i \leftarrow$ Melhor(c_i, s_{melhor});
      se $v_i = \lambda$ **então**
        $v_i \leftarrow 1$;
        se $r_i = r_{max}$ **então**
          $r_i \leftarrow 0$; $c_i \leftarrow$ Perturbação(c_i);
        senão
          BuscaLocal(c_i);
          se c_i *melhorou* **então**
            $r_i \leftarrow 0$;
          senão
            $r_i \leftarrow r_i + 1$;
          fim
        fim
        $s_{melhor} \leftarrow$ Melhor(s_{melhor}, c_i);
      fim
    fim
  fim
fim

α.

4.2 *Clustering Search* com ILS - CS_{ILS}

O *Iterated Local Search* é uma meta-heurística proposta por Lourenço, Martin e Stützle (2003) que se baseia na teoria de que um procedimento de busca local pode ter uma melhor performance quando aplicada a diferentes soluções iniciais. Para isso, obtém-se novas soluções a partir de perturbações realizadas na solução corrente, visando explorar o espaço de busca.

O ILS é um procedimento que inicia com a construção de um solução inicial qualquer que, em seguida, é submetida a uma busca local, retornando possivelmente uma

solução de melhor qualidade. Em seguida, o método inicia sua etapa iterativa, na qual a cada nova iteração são executadas três ações: realiza-se uma perturbação na solução corrente, aplica-se uma busca-local e, por fim, verifica-se a possível aceitação da solução por meio de um critério definido previamente.

O Algoritmo 7 apresenta o pseudo-código do CS utilizando o ILS como gerador de soluções para resolução do PLCT. Além dos parâmetros de configuração utilizados pelo CS (γ, λ, r_{max} e $tempo_{max}$), descritos na Seção 3.1, o algoritmo necessita também da quantidade de iterações a serem realizadas pelo ILS (ils_{max}). A execução do algoritmo ocorre de forma similar aos algoritmos CS_{SA} e CS_{GRASP}, sendo a diferença na aplicação do ILS nas linhas 6 a 10. A melhor solução obtida pelo ILS é então enviada ao CS, e o restante do processo então é similar às demais versões já apresentadas. Os procedimentos de solução inicial, busca local e perturbação utilizados são apresentados nas seções 3.2.1, 3.2.4 e 3.2.3, respectivamente.

4.3 *Clustering Search* Multi-Heurística Paralelo - CS_{MHP}

Nas Seções 3.2 (CS_{SA}), 4.1 (CS_{GRASP}) e 4.2 (CS_{ILS}) foram exibidas três diferentes versões do método *Clustering Search* utilizando, respectivamente, GRASP, SA e ILS como procedimentos geradores de soluções. Entretanto, em cada versão os algoritmos implementados executaram de forma sequencial, ou seja, enquanto o procedimento escolhido como gerador de soluções é executado, os outros componentes do CS aguardam seu término para serem executados. Como alternativa a essa situação e visando uma maior exploração do espaço de busca, foi proposta uma nova versão do CS utilizando paralelismo - CS_{MHP}. Nessa versão, as três meta-heurísticas são executadas de modo independente e simultâneo, compartilhando a mesma estrutura de *clusters*.

A Figura 5 apresenta o fluxo de execução do método que de modo geral se comporta de forma similar as versões já apresentadas. Inicialmente são inicializadas as estruturas que representam os λ *clusters*. Logo após é criada uma *thread* para cada meta-heurística que será executada em paralelo e, quando o número máximo de iterações definido para cada meta-heurística é atingido, a melhor solução obtida é submetida ao CS.

O Algoritmo 8 apresenta o pseudo-código do CS_{MHP}. O procedimento recebe como entrada os parâmetros relacionados ao CS (γ, λ, r_{max} e $tempo_{max}$), além dos parâmetros exclusivos do SA (T_0, T_f, sa_{max} e α), Grasp ($grasp_{max}$) e ILS (ils_{max}).

Sua execução segue o mesmo princípio dos algoritmos CS_{SA}, CS_{ILS} e CS_{GRASP}, detalhados nas seções anteriores. Do mesmo modo, são iniciadas as estruturas de dados que caracterizam os λ *clusters* (linha 2). Após a inicialização, são criadas *threads* individuais

Algoritmo 7: CS_{ILS} para a resolução do PLCT.

Entrada: ils_{max}, γ, λ, r_{max} e $tempo_{max}$
Saída: s_{melhor}

```
1  início
2      clt ← 0; r_i ← 0, v_i ← 0 ∀ i = 1, ..., γ;
3      s ← SoluçãoInicial; s_melhor ← s;
4      BuscaLocal(s);
5      enquanto tempo < tempo_max faça
6          para i ← 1...ils_max faça
7              Perturbação(s);
8              BuscaLocal(s);
9              s_melhor ← Melhor(s, s_melhor);
10         fim
11         se clt < γ então
12             clt ← clt + 1; v_clt ← v_clt + 1; c_clt ← s_melhor;
13         senão
14             i ← argmin_{i=1,...,γ}{H_i}; v_i ← v_i + 1; c_i ← Melhor(c_i, s_melhor);
15             se v_i = λ então
16                 v_i ← 1;
17                 se r_i = r_max então
18                     r_i ← 0; c_i ← Perturbação(c_i);
19                 senão
20                     BuscaLocal(c_i);
21                     se c_i melhorou então
22                         r_i ← 0;
23                     senão
24                         r_i ← r_i + 1;
25                     fim
26                 fim
27                 s_melhor ← Melhor(s_melhor, c_i);
28             fim
29         fim
30     fim
31 fim
```

para a execução de cada meta-heurística responsável pela geração de soluções (linha 3). Nesse momento, as meta-heurísticas são executadas paralelamente, compartilhando a mesma estrutura de *clusters* definida anteriormente.

Dentro da *thread* t_{sa} é executada a meta-heurística SA de modo semelhante ao detalhado na Seção 3.2. Inicialmente, é gerada uma solução inicial utilizando o procedimento descrito na Seção 3.2.1, (linha 3 - Algoritmo 2). A cada temperatura analisada pelo procedimento são gerados sa_{max} vizinhos e, caso o vizinho tenha melhor qualidade que a solução corrente, este é aceito. Caso contrário, o vizinho poderá ser aceito dada uma probabilidade. Em seguida, é feito o resfriamento da temperatura e a solução corrente

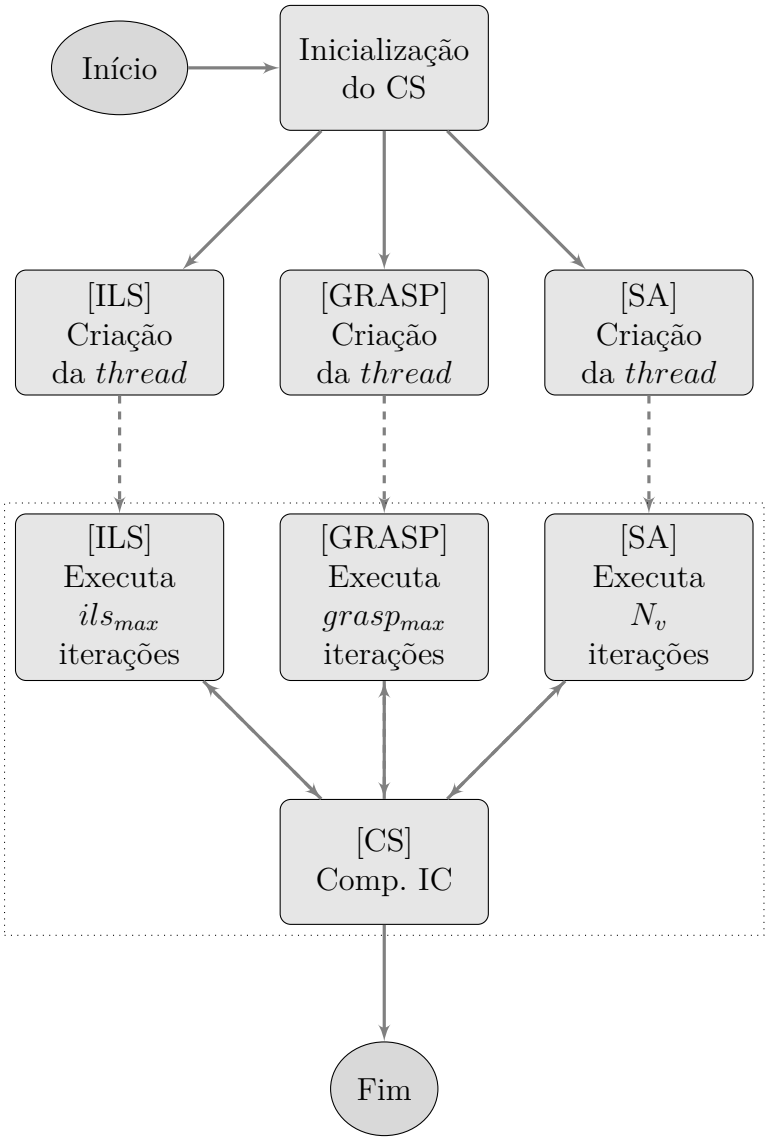

Figura 5 – Fluxo de execução do algoritmo CS_{MHP}.

é enviada a componente IC do CS. O mesmo ocorre para as meta-heurísticas GRASP e ILS. Na *thread* t_{grasp} é executado o GRASP com a mesma lógica escrita na Seção 4.1, representada pelas linhas 4 a 8 do Algoritmo 6. A melhor solução alcançada entre as $grasp_{max}$ iterações é enviada à mesma componente IC do CS. No ILS (Seção 4.2), representado pela *thread* t_{ils}, é criada uma solução inicial (linha 4 - Algoritmo 7), executada as três componentes do método por ils_{max} vezes e a melhor solução também é enviada a IC do CS.

A componente IC é executada analogamente ao descrito nas seções anteriores (3.2, 4.1 e 4.2) e seu pseudo-código é descrito nas linhas 11 a 30 do Algoritmo 7, por exemplo. A solução final s_{melhor} do CS_{MHP} é a melhor solução encontrada entre as *threads* t_{sa}, t_{grasp} e t_{ils}.

Algoritmo 8: CS_{MHP} para a resolução do PLCT.

Entrada: γ, λ, r_{max}, $tempo_{max}$, T_0, T_f, sa_{max}, α, $grasp_{max}$ e ils_{max}
Saída: s_{melhor}

```
1  início
2      clt ← 0; r_i ← 0, v_i ← 0 ∀ i = 1, ..., γ;
3      t_sa, t_grasp, t_ils ← Threads();
4      enquanto tempo < tempo_max faça
5          t_sa(T_0, T_f, sa_max, α)
6          t_grasp(grasp_max)
7          t_ils(ils_max)
8      fim
9      s_melhor ← Melhor(t_sa, t_grasp, t_ils)
10 fim
```

5 Experimentos e Resultados

Neste capítulo são descritos os experimentos computacionais realizados e apresentados os resultados computacionais obtidos. Além disso, são descritos detalhes de implementação, as instâncias usadas para testar os algoritmos, a escolha dos parâmetros utilizados no *Clustering Search* e os parâmetros específicos das meta-heurísticas SA, Grasp e ILS. No final do capítulo, os resultados alcançados são comparados com os melhores conhecidos na literatura.

5.1 Descrição das Instâncias Utilizadas

As instâncias utilizadas neste trabalho foram as mesmas propostas por Gonzalez et al. (2016). Para cada estado brasileiro, foi gerada uma instância que o representa de acordo com a base rodoviária brasileira georreferenciada, disponibilizada pelo DNIT em 2015. No total, foram utilizadas 26 instâncias, sendo que cada nó representa um município ou um cruzamento ao longo das rodovias, e cada aresta representa um trecho de rodovia federal ou estadual contida no estado da instância. Os municípios foram considerados para a definição de pares O-D, ou seja, foram considerados os fluxos de veículos entre todas as combinações de pares de municípios situados no mesmo estado. Na Tabela 1 são apresentados os atributos mais relevantes de cada instância. $|N|$ representa o total de nós e $|A|$ o total de arestas, $|M|$ é a quantidade de municípios contidos no estado e, com isso $|W|$ o total de pares a serem cobertos. É importante destacar que $|M| \leq |N|$ dado que, além dos municípios, os cruzamentos também são considerados nós na rede de transporte. A maior instância encontrada é a do estado de Minas Gerais, com 1474 nós, 1917 arestas, 803 municípios e mais de 300 mil pares O-D.

Tabela 1 – Características das instâncias utilizadas.

Instância	$\|N\|$	$\|A\|$	$\|M\|$	$\|W\|$	Instância	$\|N\|$	$\|A\|$	$\|M\|$	$\|W\|$
AC	61	84	20	190	PB	384	480	213	22578
AL	169	219	97	4656	PE	362	472	172	14706
AM	74	77	37	666	PI	405	550	212	22366
AP	52	77	13	78	PR	780	1083	381	72390
BA	812	1113	395	77815	RJ	502	721	86	3655
CE	401	612	177	15576	RN	333	433	160	12720
ES	283	394	75	2775	RO	185	258	50	1225
GOeDF	799	1165	241	28920	RR	75	97	13	78
MA	257	355	163	13203	RS	675	859	391	76245
MG	1474	1917	803	322003	SC	481	604	266	35245
MS	343	497	76	2850	SE	183	248	74	2701
MT	711	1069	140	9730	SP	1280	1683	606	183315
PA	289	370	122	7381	TO	372	524	134	8911

5.2 Detalhes de Implementação

Todos os algoritmos apresentados foram implementados utilizando a linguagem de programação *C++11*, compiladas com *GCC* versão 4.9.2 64 bits e testadas em uma máquina com processador *Intel* i7 @ 3.4GHz, 16Gb de memória *RAM* e *Windows* 7.

Os nós $n \in N$ e as arestas $a \in A$ da rede foram rotulados com um índice inteiro único. Para o caso dos nós, foi criado um atributo binário para indicar o tipo do nó. Caso esse atributo seja 1, o nó é considerado um município, e 0 um cruzamento. A solução para o PLCT foi modelada com um vetor binário $\mathbf{x}$ de comprimento $|A|$, representando todas as arestas (rodovias) do estado. A posição $x_a \; \forall \; a = 1, .., |A|$ de $\mathbf{x}$ com valor 1 indica a presença de uma estação de contagem na aresta a, e 0 contrário.

Para auxiliar na verificação da cobertura completa dos caminhos entre os pares O-D, a rede foi modelada por uma estrutura de árvore N-ária, na qual os nós da árvore são os nós presentes na rede e as arestas determinam a ligação entre os nós. Para uma cobertura completa entre os pares O-D, todas as folhas acessíveis a partir do nó raiz da árvore devem conter pelo menos uma aresta com um contador alocado. Por ser uma operação que é realizada constantemente durante a execução das meta-heurísticas desenvolvidas, essa estrutura proporcionou a agilidade na verificação cobertura completa da rede.

5.3 Escolha dos Parâmetros

O CS possui três parâmetros principais: o número máximo de *clusters* γ, o volume máximo de cada *cluster* λ e o índice de ineficácia r_{max}. A meta-heurística SA possui como parâmetro as temperaturas inicial T_0 e final T_f, o número de iterações realizadas a cada temperatura sa_{max} e a taxa de resfriamento α. Os valores dos parâmetros do CS utilizando nas meta-heurísticas CS_{SA}, CS_{GRASP}, CS_{ILS} e CS_{MHP} foram baseados no trabalho de Gonzalez et al. (2019). Outros valores foram testados, porém nenhuma combinação foi capaz de gerar resultados médios melhores. Já os parâmetros relacionadas ao máximo de iterações do GRASP ($grasp_{max}$) e do ILS (ils_{max}) foram determinados como $2 \times |A|$, o mesmo valor de sa_{max} definido no SA de Gonzalez et al. (2019). Por fim, para realizar uma comparação justa com os resultados de Gonzalez et al. (2019), que apresentam os melhores resultados conhecidos até então, o tempo máximo de execução das meta-heurísticas foi limitado a 2 horas. A Tabela 2 exibe os parâmetros, o valor adotado e significado de cada um.

5.4 Resultados Obtidos

Os algoritmos CS_{SA} (GONZALEZ et al., 2019), CS_{GRASP}, CS_{ILS} e CS_{MHP} foram executados 10 vezes para cada instância com diferentes sementes para a geração de números

Tabela 2 – Parâmetros do *Clustering Search*

Parâmetro	Descrição	Valor
γ	Número máximo de $clusters$	3
λ	Volume máximo dos $clusters$	2
r_{max}	Índice máximo de ineficácia dos $clusters$	3
T_0	Temperatura inicial	$f(s_0)$
T_f	Temperatura de congelamento	$0,01$
sa_{max}	Número máximo de iterações por temperatura	$2\|A\|$
α	Taxa de Resfriamento	$0,975$
$grasp_{max}$	Número máximo de iterações	$2\|A\|$
ils_{max}	Número máximo de iterações	$2\|A\|$
$tempo_{max}$	Tempo limite (segundos)	7200

aleatórios e tempo máximo de execução definido em 7200 segundos (2 horas). A Tabela 3 apresenta os resultados obtidos para cada instância pelos métodos propostos neste trabalho. A primeira coluna indica o nome da instância e a segunda $Mel.f(s)$ apresenta a quantidade mínima de contadores de veículos utilizados para cobertura total do estado, ou seja, o melhor valor de função objetivo encontrado. A terceira e quinta colunas apresentam, respectivamente, a média da função objetivo das soluções encontradas e o tempo médio (em segundos). Por fim, a quarta coluna apresenta o desvio para cada instância, calculado da seguinte forma: $Des.\ (\%) = 100 \times \dfrac{Méd.\ f(s) - Mel.\ f(s)}{Mel.\ f(s)}$. Em negrito estão destacadas as melhores soluções obtidas.

Tabela 3 – Resultados obtidos pelos algoritmos propostos.

Instância	CS_{GRASP}				CS_{ILS}				CS_{MHP}			
	Mel. $f(s)$	*Méd.* $f(s)$	*Des.* (%)	*Tempo* (*seg.*)	*Mel.* $f(s)$	*Méd.* $f(s)$	*Des.* (%)	*Tempo* (*seg.*)	*Mel.* $f(s)$	*Méd.* $f(s)$	*Des.* (%)	*Tempo* (*seg.*)
AC	**30**	30,00	0,00	0,04	**30**	30,00	0,00	0,09	**30**	30,00	0,00	0,02
AL	**137**	137,00	0,00	0,08	**137**	137,00	0,00	0,44	**137**	137,00	0,00	0,03
AM	**39**	39,00	0,00	0,01	**39**	39,00	0,00	0,01	**39**	39,00	0,00	0,00
AP	**22**	22,00	0,00	0,03	**22**	22,00	0,00	0,10	**22**	22,00	0,00	0,01
BA	633	635,10	0,33	633,17	**630**	630,00	0,00	750,36	**630**	630,00	0,00	251,23
CE	334	335,00	0,30	72,81	**329**	330,60	0,48	143,79	**329**	329,70	0,21	27,34
ES	**144**	144,40	0,28	70,68	**144**	144,00	0,00	17,98	**144**	144,00	0,00	0,57
GOeDF	471	474,20	0,68	847,18	**464**	465,67	0,36	1045,83	**464**	464,50	0,11	65,94
MA	**250**	250,00	0,00	0,15	**250**	250,00	0,00	0,30	**250**	250,00	0,00	0,13
MG	1135	1136,30	0,11	3028,89	**1121***	1122,80	0,16	6191,76	**1121***	1122,40	0,04	2068,38
MS	151	151,30	0,20	468,18	**150**	150,67	0,44	832,47	**150**	150,00	0,00	2,80
MT	**308**	310,00	0,65	2985,40	**308***	312,20	1,35	4327,43	**305***	306,50	0,49	657,83
PA	175	175,90	0,51	105,83	**174**	174,.00	0,00	117,45	**174**	174,00	0,00	0,90
PB	**296**	296,00	0,00	0,98	**296**	296,00	0,00	2,63	**296**	296,00	0,00	0,29
PE	254	254,70	0,28	440,95	**252**	252,00	0,00	94,23	**252**	252,00	0,00	0,44
PI	318	318,80	0,25	54,80	**316**	316,00	0,00	50,46	**316**	316,00	0,00	2,50
PR	606	606,50	0,08	461,62	**599**	599,33	0,06	955,58	**599**	599,00	0,00	116,01
RJ	172	172,70	0,41	52,38	**166**	168,00	1,19	1485,18	**166**	167,40	0,08	15,24
RN	**233**	233,00	0,00	0,96	**233**	233,00	0,00	2,74	**233**	233,00	0,00	0,21
RO	**88**	88,70	0,80	34,19	**88**	88,00	0,00	5,57	**88**	88,00	0,00	0,18
RR	**19**	19,00	0,00	6,34	**19**	19,00	0,00	9,02	**19**	19,00	0,00	0,03
RS	547	547,60	0,11	702,21	**544**	544,00	0,00	250,44	**544**	544,00	0,00	18,22
SC	372	372,50	0,13	58,73	**371**	371,00	0,00	149,97	**371**	371,00	0,00	5,00
SE	113	113,00	0,00	4,12	**112**	112,00	0,00	7,63	**112**	112,00	0,00	1,49
SP	887	888,30	0,15	1082,32	875	877,70	0,31	1469,81	875	876,30	0,05	963,79
TO	235	236,00	0,43	34,49	**231**	231,50	0,22	35,36	**231**	231,00	0,00	10,91
Média	306,50	307,19	0,21	428,71	303,84	304,44	0,17	582,22	303,73	303,99	0,03	161,90

* Novas melhores soluções.

Como pode ser observado na Tabela 3, a meta-heurística CS_{MHP} apresentou o melhor desempenho, encontrando soluções equivalentes para todas as instâncias testadas, inclusive apresentando a melhor solução encontrada na literatura para o estado de SP. A performance do CS_{ILS} foi semelhante ao CS_{MHP}, exceto para as instâncias MT e SP. Já o CS_{GRASP} alcançou soluções com qualidade semelhante às demais meta-heurísticas apenas em algumas instâncias.

A média das soluções encontradas é praticamente a mesma entre as três meta-heurísticas testadas, sendo um pouco melhor para o CS_{MHP}. Já o desvio apresentado pelo CS_{MHP} é significativamente mais baixo em relação ao CS_{ILS} e ao CS_{GRASP} que tiveram o desempenho semelhante. Esse fato se dá pois o CS_{MHP} é constantemente alimentado com soluções geradas pelas diferentes meta-heurísticas, convergindo rapidamente para as regiões do espaço de busca que possuem as melhores soluções encontradas, característica que também reflete na média de tempo em que as melhores soluções são encontradas, cerca de 70% menor que os outros métodos apresentados.

A Tabela 4 apresenta uma comparação entre os melhores resultados obtidos neste trabalho, pelo CS_{MHP}, e os resultados obtidos pelo CS_{SA} de Gonzalez et al. (2019), que são os melhores conhecidos na literatura até então. Observa-se que as meta-heurísticas CS_{MHP} e CS_{SA} apresentam média similar para as melhores soluções obtidas. Entretanto, o desempenho do CS_{MHP} se destaca quando analisados os outros resultados, ou seja, foram obtidos, em média, menores desvios e soluções médias, além do tempo necessário para encontrar as melhores soluções, que foi cerca de 80% menor. O CS_{MHP} encontrou soluções iguais ou melhores para todas as instâncias, exceto para o estado de SP, para o qual a solução obtida foi ligeiramente pior (um contador a mais). Por fim, é possível observar que, paras as instâncias MG e MT, o CS_{MHP} encontrou soluções melhores do que as apresentadas na literatura até então, com menores desvios e tempos computacionais.

Tabela 4 – Comparação com o CS_{SA} de Gonzalez et al. (2019).

Instância	CS_{SA} (GONZALEZ et al., 2019)				CS_{MHP}			
	Mel. $f(s)$	*Méd.* $f(s)$	*Des.* (%)	*Tempo* (*seg.*)	*Mel.* $f(s)$	*Méd.* $f(s)$	*Des.* (%)	*Tempo* (*seg.*)
AC	**30**	30,00	0,00	0,09	**30**	30,00	0,00	0,02
AL	**137**	137,00	0,00	0,44	**137**	137,00	0,00	0,03
AM	**39**	39,00	0,00	0,01	**39**	39,00	0,00	0,00
AP	**22**	22,00	0,00	0,10	**22**	22,00	0,00	0,01
BA	**630**	630,00	0,00	750,36	**630**	630,00	0,00	251,23
CE	**329**	329,00	0,00	143,79	**329**	329,70	0,21	27,34
ES	**144**	144,00	0,00	17,98	**144**	144,00	0,00	0,57
GOeDF	**464**	465,50	0,32	1045,83	**464**	464,50	0,11	65,94
MA	**250**	250,00	0,00	0,30	**250**	250,00	0,00	0,13
MG	1122	1124,70	0,24	6191,76	**1121***	1122,40	0,04	2068,38
MS	**150**	150,00	0,00	832,47	**150**	150,00	0,00	2,80
MT	310	311,80	0,58	4327,43	**305***	306,50	0,49	657,83
PA	**174**	174,00	0,00	117,45	**174**	174,00	0,00	0,90
PB	**296**	296,00	0,00	2,63	**296**	296,00	0,00	0,29
PE	**252**	252,00	0,00	94,23	**252**	252,00	0,00	0,44
PI	**316**	316,00	0,00	50,46	**316**	316,00	0,00	2,50
PR	**599**	599,80	0,13	955,58	**599**	599,00	0,00	116,01
RJ	**166**	167,70	1,02	1485,18	**166**	167,40	0,08	15,24
RN	**233**	233,00	0,00	2,74	**233**	233,00	0,00	0,21
RO	**88**	88,00	0,00	5,57	**88**	88,00	0,00	0,18
RR	**19**	19,00	0,00	9,02	**19**	19,00	0,00	0,03
RS	**544**	544,00	0,00	250,44	**544**	544,00	0,00	18,22
SC	**371**	371,00	0,00	149,97	**371**	371,00	0,00	5,00
SE	**112**	112,00	0,00	7,63	**112**	112,00	0,00	1,49
SP	**874**	877,30	0,38	1469,81	875	876,30	0,05	963,79
TO	**231**	304,38	0,00	35,36	**231**	231,00	0,00	10,91
Média	303,92	307,20	0,10	690,26	303,73	303,99	0,04	161,90

* Novas melhores soluções.

6 Conclusões

Esta dissertação tratou o Problema de Localização de Contadores de Tráfego (PLCT) em redes de transporte, considerando dados reais referentes às rodovias estaduais e federais dos estados brasileiros. Para a resolução do PLCT, foram utilizadas três novas alternativas ainda não exploradas da meta-heurística *Clustering Search* (CS): CS_{GRASP}, CS_{ILS} e CS_{MHP}. As meta-heurísticas GRASP e ILS foram testadas, além do SA, na geração de soluções para o CS. Além disso, foi proposta também uma versão paralela utilizando as três heurísticas em conjunto, sendo esta uma abordagem ainda não explorada para resolução desse problema na literatura.

Para avaliar o desempenho das três versões propostas, experimentos computacionais foram realizados considerando as mesmas instâncias utilizadas em outro trabalho recente da literatura. Destacam-se os resultados obtidos pelas versões CS_{ILS} e CS_{MHP} que conseguiram alcançar soluções equivalentes para 23 instâncias, além de 2 novas soluções de melhor qualidade. Todas as soluções foram encontradas em menor tempo computacional quando comparados aos resultados presentes na literatura. Os resultados obtidos pelo CS_{MHP} demonstram um algoritmo rápido e robusto, capaz de produzir boas soluções com baixo desvio e utilizando 80% menos tempo computacional.

Como trabalhos futuros, pode-se apontar a extensão na realização de testes com o método proposto CS_{MHP} considerando diferentes configurações de parâmetros, a implementação de novas meta-heurísticas geradoras de solução para a resolução do PLCT e, por fim, a aplicação do método em outros problemas combinatórios.

Referências

AHUJA, R. K.; MAGNANTI, T. L.; ORLIN, J. B. *Network Flows: Theory, Algorithms, and Applications*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1993. Citado na página 32.

BELL, M. G. H. The estimation of an origin-destination matrix from traffic counts. *Transportation Science*, v. 17, n. 2, p. 198–217, 1983. Citado na página 27.

CHEN, A.; CHOOTINAN, P.; RECKER, W. Examining the quality of synthetic origin–destination trip table estimated by path flow estimator. *Journal of Transportation Engineering*, v. 131(7), p. 506–513, 2005. Citado na página 27.

CHEN, A. et al. Strategies for selecting additional traffic counts for improving od trip table estimation. *Transportmetrica*, v. 3, n. 3, p. 191–211, 2007. Citado 2 vezes nas páginas 20 e 28.

COSTA, M.; FIEDLER, N.; MAURI, G. Clustering search e simulated annealing para resolução do problema de escalonamento de motoristas no transporte de madeira. *Expert Systems with Applications*, v. 41, n. 99, p. 299–305, 2013. Citado na página 28.

DNIT. *Departamento Nacional de Infraestrutura de Transportes. Plano Nacional de Contagem de Tráfego - PNCT*. 1970. <http://servicos.dnit.gov.br/dadospnct/Inicio/institucional>. Acessado em: 20/07/2019. Citado na página 20.

FEO, T. A.; RESENDE, M. G. C. A probabilistic heuristic for a computationally difficult set covering problem. *Operations Research Letters*, v. 8, n. 2, p. 67–71, 1989. Citado na página 37.

GARBER, N. J.; HOEL, L. A. Traffic and highway engineering. *CL Engineering.*, 1999. Citado na página 19.

GENTILI, M.; MIRCHANDANI, P. Locating active sensors on traffic networks. *Annals of Operations Research*, v. 136, n. 1, p. 229–257, 1 2005. Citado na página 19.

GENTILI, M.; MIRCHANDANI, P. Locating sensors on traffic networks: Models, challenges and research opportunities. *Transportation Research Part C: Emerging Technologies*, v. 24, p. 227–255, 2012. Citado na página 19.

GONZALEZ, P. H. et al. New approaches for the traffic counting location problem. *Expert Systems with Applications*, v. 132, p. 189–198, 2019. Citado 14 vezes nas páginas 11, 13, 21, 24, 28, 30, 31, 33, 34, 35, 37, 44, 47 e 48.

GONZALEZ, S. P. H. et al. Heurísticas para o problema de localização de contadores de tráfego em redes de transporte. *Anais do XXX ANPET - Congresso de Pesquisa e Ensino em Transportes*, 12 2016. Citado 3 vezes nas páginas 28, 32 e 43.

HAMMING, R. W. Error detecting and error correcting codes. *Bell System Technical Journal*, v. 29, n. 2, p. 147–160, 1950. Citado na página 33.

KIRKPATRICK, S.; GELATT, C. D.; VECCHI, M. P. Optimization by simulated annealing. *Science*, v. 220, n. 4598, p. 671–680, 1983. Citado na página 30.

LOURENÇO, H. R.; MARTIN, O. C.; STÜTZLE, T. Iterated local search. In: *Handbook of Metaheuristics*. [S.l.: s.n.], 2003. p. 320–353. Citado na página 38.

MAHER, M. J.; ZHANG, X.; VLIET, D. V. A bi-level programming approach for trip matrix estimation and traffic control problems with stochastic user equilibrium link flows. *Transportation Research Part B: Methodological*, v. 35, n. 1, p. 23–40, 2001. Citado na página 27.

MAURI, G. et al. Meta-heurística cs para o problema de localização de contadores de tráfego em redes de transporte. *XLIX SBPO - Simpósio Brasileiro de Pesquisa Operacional*, Blumenau (Brasil), 2017. Citado 2 vezes nas páginas 22 e 28.

MELO, R.; BARROS JUNIOR, A.; MAURI, G. Clustering search com iterated local search para resolução do problema de planejamento florestal. *XLV SBPO - Simpósio Brasileiro de Pesquisa Operacional*, Natal (Brasil), Setembro 2013. Citado na página 22.

MELO, R.; BARROS JUNIOR, A.; MAURI, G. Resolução de um problema de planejamento florestal via clustering search utilizando a meta-heurística grasp como geradora de soluções. *XLVI SBPO - Simpósio Brasileiro de Pesquisa Operacional*, Salvador (Brasil), Setembro 2014. Citado na página 22.

OLIVEIRA, A. C. M.; CHAVES, A. A.; LORENA, L. A. N. Clustering search. *Pesquisa Operacional*, v. 33, n. 1, p. 105–121, 2013. Citado 2 vezes nas páginas 28 e 29.

OLIVEIRA, R.; MAURI, G.; LORENA, L. Clustering search for the berth allocation problem. *Expert Systems with Applications*, v. 39, n. 5, p. 5499–5505, 2012. Citado na página 28.

RABELLO, R. L. et al. A clustering search metaheuristic for the point-feature cartographic label placement problem. *European Journal of Operational Research*, v. 234, n. 3, p. 802–808, 2014. Citado na página 28.

RIBEIRO, G. M.; LAPORTE, G.; MAURI, G. R. A comparison of three metaheuristics for the workover rig routing problem. *European Journal of Operational Research*, v. 220, n. 1, p. 28–36, 2012. Citado na página 28.

RODRIGUES, P. et al. Resolução de um caso real do problema dial-a-ride multi-critério via clustering search. *Produção*, v. 24, n. 3, p. 572–582, 2014. Citado na página 28.

ROSA, R. et al. Clustering search e simulated annealing para resolução do problema de escalonamento de motoristas no transporte de madeira. *Computers & Industrial Engineering*, v. 101, p. 303–312, 2016. Citado na página 28.

ROSA, R. et al. Planning the berth allocation problem in developing countries with multiple cargos and cargo priority by a mathematical model and a clustering search metaheuristic. *International Journal of Logistics Systems and Management*, v. 28, n. 4, p. 397–418, 2017. Citado na página 28.

YANG, H.; GAN, L.; TANG, W. H. Determing cordons and screen lines for origin-destination trip studies. *Proceedings of the Eastern Asia Society for Transportation Studies*, 2001. Citado na página 27.

YANG, H.; YANG, C.; GAN, L. Models and algorithms for the screen line-based trafficcounting location problems. *Computers & Operations Research*, v. 33, n. 3, p. 836–858, 2006. Citado 3 vezes nas páginas 23, 25 e 28.

YANG, H.; ZHOU, J. Optimal traffic counting locations for origin-destination matrix estimation. *Transportation Research Part B: Methodological*, v. 32, p. 109–126, 1998. Citado na página 27.

ZUYLEN, H. J. V.; WILLUMSEN, L. G. The most likely trip matrix estimated from traffic counts. *Transportation Research Part B: Methodological*, v. 14, n. 3, p. 281–293, 1980. Citado na página 27.