

METAHEURÍSTICA ILS APLICADA AO PROBLEMA DE p -MEDIANAS CAPACITADO

Brasílio Alves Freitas^{1,2}, Sérgio Ricardo de Souza², Elisangela Martins de Sá²

¹ Instituto Federal de Educação, Ciência e Tecnologia de Minas Gerais

Rua Itaguaçu, 595, CEP: 32677-562, Betim, MG, Brasil

² Centro Federal de Educação Tecnológica de Minas Gerais

Av. Amazonas, 7675, CEP: 30510-000, Belo Horizonte, MG, Brasil

brasilio.freitas@ifmg.edu.br, sergio@cefetmg.br,

elisangelamartins@cefetmg.br

RESUMO

Este trabalho propõe um algoritmo baseado na meta-heurística *Iterated Local Search* (ILS) para a solução do problema de p -medianas capacitado. A construção da solução inicial é baseada na fase construtiva da meta-heurística *Greedy Randomized Adaptive Search Procedure* (GRASP). Foram realizados experimentos computacionais utilizando 35 instâncias da literatura. Os resultados, obtidos em 30 execuções da heurística, mostram *gaps* médios inferiores a 0,60% para os melhores valores da função objetivo, em todas as instâncias, comparados com os valores obtidos pelo *solver* CPLEX e por Stefanello et al. [2015], e valores médios para a função objetivo com *gap* médio inferior a 2,00%. Além disso, em comparação aos resultados de Vasconcelos [2023], obteve-se *gap* médio negativo para os melhores valores de função objetivo e tempos médios de execução menores em aproximadamente 50% das instâncias.

PALAVRAS CHAVE. Problema de p -medianas capacitado, Localização de Facilidades, Metaheurística ILS.

Tópicos: MH – Meta-heurísticas, OC – Otimização Combinatória.

ABSTRACT

This work proposes an algorithm based on the *Iterated Local Search* (ILS) metaheuristic to solve the capacitated p -median problem. The construction of the initial solution is based on the constructive phase of the *Greedy Randomized Adaptive Search Procedure* (GRASP) metaheuristic. Computational experiments were carried out using 35 instances from the literature. The results, obtained in 30 executions of the heuristic, show average *gaps* lower than 0.60% for the best values of the objective function, in all instances, compared with the values obtained by the *solver* CPLEX and by Stefanello et al. [2015], and average values for the objective function with an average *gap* lower than 2.00%. Furthermore, compared to the results of Vasconcelos [2023], a negative average *gap* was obtained for the best objective function values, and lower average execution times in approximately 50% of the instances.

KEYWORDS. Capacitated p -median problem. Facility location. ILS metaheuristic.

Paper topics: MH - Metaheuristics, OC- Combinatorial optimization.

1. Introdução

A demanda por soluções de problemas de otimização tem crescido consideravelmente. Com isso, surge a necessidade de se aplicar métodos para ajudar na tomada de decisões, dentre eles estão o modelamento matemático computacional para analisar e solucionar situações mais complexas.

Uma classe de problema de otimização muito importante é a classe de Problemas de Localização. Segundo Drezner [1996], dentro da classe de Problemas de Localização, é de grande interesse as decisões sobre a melhor configuração para instalação de facilidades destinadas ao atendimento da demanda de uma população. Problemas de localização de facilidades ocorrem em diferentes contextos e possuem grande importância econômica para o planejamento estratégico, tanto de setores privados quanto públicos, visto que a otimização destes modelos pode trazer grandes economias.

Problemas de localização de facilidades consistem em selecionar um conjunto de instalações, dentre um conjunto de candidatas, de forma a otimizar um ou mais objetivo(s) que atendam a um conjunto de restrições. Dentre os problemas de localização, os problemas de p -medianas (PPM), descrito pela primeira vez por Hakimi [1964], são muito significativos. Nestes problemas, dado um conjunto com n instalações candidatas, que podem se referir a pontos de demanda, o objetivo é localizar um subconjunto com p elementos (medianas) e alocar às mesmas as demais instalações (clientes), de forma a minimizar a soma das distâncias de cada instalação até a mediana mais próxima ([Avella et al., 2007]). Mesmo que a localização das medianas sejam conhecidas, o problema associado à alocação de clientes às medianas corresponde ao problema generalizado de atribuição (*Generalized Assignment Problem* - GAP), que é um problema NP-difícil ([Fisher et al., 1986]). Há duas versões básicas para problemas de localização de p -medianas: (i) **não-capacitado** (PPMnC), quando as instalações não possuem limite de capacidade, ou (ii) **capacitado** (PPMC), quando as instalações possuem limite de capacidade.

De modo que se entenda melhor a representação de um problema de p -medianas, a Figura 1 mostra um exemplo de solução em uma rede com 10 instalações e 3 medianas.

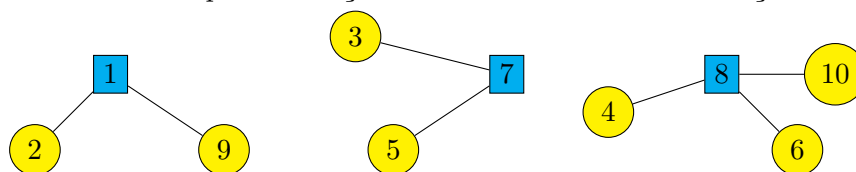


Figura 1: Exemplo de um problema de p -medianas

Várias estratégias já foram aplicadas na resolução do PPM, por exemplo, a relaxação Lagrangeana em Avella et al. [2012]; a geração de colunas em Lorena e Senne [2004]; o método *branch-and-price* em Ceselli e Righini [2005], o método *branch-and-cut* em Bektaş et al. [2019], método heurístico em Herda [2015]. Vasconcelos [2023] apresenta quatro variantes da meta-heurística *General Variable Neighborhood Search* - (GVNS), que se diferem na forma de construção da solução inicial e no método de busca local. Stefanello et al. [2015] apresentam uma abordagem matheurística que hibridiza meta-heurísticas baseadas em busca local e técnicas de programação matemática para resolver o problema p -mediana capacitado. Segundo Vasconcelos [2023], muitas dessas estratégias são baseadas em métodos exatos, mas, devido ao custo computacional elevado, o uso de algoritmos baseados em heurísticas e meta-heurísticas têm sido propostos.

Este artigo propõe um algoritmo baseado na meta-heurística *Iterated Local Search* - (ILS) ([Lourenço et al., 2002]) para a solução do PPMC e está organizado da seguinte

forma: a Seção 2 apresenta a descrição do PPMC e as formulações matemáticas utilizadas no presente trabalho. A Seção 3 descreve o algoritmo proposto. Os experimentos computacionais são apresentados na Seção 4. A Seção 5 apresenta as considerações finais e propostas de trabalhos futuros.

2. Descrição do Problema e Formulações Matemáticas

Em [Yaghini et al., 2013], o PPMC é definido a partir de um grafo $G = (V, A)$ não direcionado, em que V é o conjunto dos nós e A é o conjunto das arestas. Seja N o conjunto de instalações e M o conjunto de clientes, com $N \subseteq V$ e $M \subseteq V$, sendo $n = |N|$ o número de instalações candidatas e $m = |M|$ o número de clientes a serem atendidos. A cada cliente $j \in M$ está associada uma demanda q_j e a cada instalação $i \in N$ está associada uma capacidade de atendimento Q_i . O custo unitário do transporte entre uma instalação $i \in N$ e um cliente $j \in M$ é proporcional à distância entre as mesmas (d_{ij}), sendo $d_{ij} > 0$ e $d_{jj} = 0$. O valor $p \leq n$ define o número de medianas.

O PPMC consiste em determinar um conjunto de medianas $N_p \subseteq N$, alocar às mesmas as instalações clientes, de modo a minimizar a soma das distâncias entre os clientes e as medianas à qual estão alocados, sem infringir o limite de capacidade da instalação (Q_i). Seja y_i a variável de decisão binária que representa a escolha da instalação i como mediana, sendo $y_i = 1$ quando i for definida como mediana; e $y_i = 0$, caso contrário. Seja x_{ij} a variável de decisão binária que assume valor $x_{ij} = 1$ se o cliente j é atendido pela instalação i ; e o valor $x_{ij} = 0$, caso contrário.

Para o problema das p -medianas capacitado, descrito neste trabalho, foram estudadas duas formulações matemáticas, que se diferem devido a inclusão das restrições de acoplamento desagregadas na Formulação 2. As duas formulações são:

$$\text{(Formulação 1)} \quad \min \quad Z = \sum_{i \in N} \sum_{j \in M} d_{ij} x_{ij} \quad (1)$$

$$\text{sujeito a:} \quad \sum_{i \in N} x_{ij} = 1, \quad \forall j \in M \quad (2)$$

$$\sum_{i \in N} y_i = p \quad (3)$$

$$\sum_{j \in M} q_j x_{ij} \leq Q_i y_i, \quad \forall i \in N \quad (4)$$

$$x_{ij} \in \{0, 1\}, \quad y_i \in \{0, 1\} \quad \forall i \in N, j \in M \quad (5)$$

$$\text{(Formulação 2)} \quad \min \quad Z = \sum_{i \in N} \sum_{j \in M} d_{ij} x_{ij} \quad (6)$$

$$\text{sujeito a:} \quad (2), (3), (4), (5) \\ x_{ij} \leq y_j \quad \forall i \in N, j \in M \quad (7)$$

Nas duas formulações, as funções objetivo (1) e (6), a serem minimizadas, representam as somas dos custos (distâncias) entre os clientes e suas medianas. As restrições (2) garantem que cada cliente é atendido por apenas uma instalação. As restrições (3) garantem que serão escolhidas exatamente p instalações como medianas. As restrições (4) impõem respeito ao limite de capacidade das instalações. As restrições (5) definem o domínio das variáveis de decisão. As desigualdades (7), na Formulação 2, mostram que as restrições de acoplamento estão desagregadas, ou seja, um cliente só será alocado a uma instalação se ela for mediana.

3. Descrição do Algoritmo Proposto

Esta seção descreve o algoritmo proposto para solucionar o PPMC. A Subseção 3.1 mostra como é feita a representação e a avaliação de uma solução para o PPMC. A Subseção 3.2 detalha o procedimento construtivo da solução inicial. A Subseção 3.3 descreve a estrutura de vizinhança e o método de busca local utilizado. Na Subseção 3.4 é descrito o processo de perturbação. O algoritmo ILS aplicado ao PPMC é mostrado na Subseção 3.5.

3.1. Representação e Função de Avaliação da Solução

Uma solução para o PPMC é representada por dois vetores de inteiros. O primeiro vetor (v_m) tem dimensão p (número de medianas) e mostra quais os índices das instalações escolhidas como medianas. Os elementos desse vetor compõem o conjunto das p -medianas (N_p). O segundo vetor (v_a) tem dimensão n (número de instalações) e mostra o índice da mediana em que cada instalação está alocada. Assim, por exemplo, para $n = 10$ instalações e $p = 3$ medianas, os vetores $v_m = [1, 7, 8]$ e $v_a = [1, 1, 7, 8, 7, 8, 7, 8, 1, 8]$ mostram que as instalações 1, 7 e 8 são medianas e que os clientes 1, 2 e 9 estão alocados à mediana $j = 1$; os clientes 3, 5 e 7 estão alocados à mediana $j = 7$ e os clientes 4, 6, 8 e 10 estão alocados à mediana $j = 8$.

Definidas as medianas, as alocações das instalações podem ser feitas aleatoriamente, formando *clusters*, respeitando a capacidade de atendimento de cada instalação. As alocações obedecem ao *Generalized Assignment Problem* (GAP). O GAP consiste em encontrar o menor custo para a atribuição de n tarefas a p agentes, sendo que cada tarefa deve ser atribuída a apenas um agente, sujeito à capacidade dos mesmos.

A avaliação da solução é feita utilizando as expressões:

$$f_o = \sum_{i \in M} d_{ij}, \quad \forall j \in N \quad (8)$$

$$Q_j = \sum_{i \in M} q_{ij}, \quad \forall j \in N_p \quad (9)$$

A Expressão (8) calcula a soma das distâncias entre cada cliente (elementos do vetor v_a) e a mediana na qual está alocado. Esta expressão é usada para avaliar o valor de função objetivo. A Expressão (9) retorna o somatório das demandas associadas a cada *cluster*. Esta expressão é usada para avaliar a viabilidade de uma solução.

3.2. Construção da Solução Inicial

Esta seção apresenta o procedimento de construção da solução inicial por meio do procedimento construtivo da meta-heurística *Greedy Randomized Adaptive Search Procedure* (GRASP) [Feo e Resende, 1995], cujos passos estão descritos a seguir.

Dado um conjunto N de instalações candidatas e um conjunto M de clientes, cada elemento candidato $i \in N$ será avaliado por meio da soma acumulada das distâncias que o separa de todos os clientes $j \in M$, usando a função:

$$g(i) = \sum_{j \in M} d_{ij} \quad (10)$$

Após a avaliação, insere-se os candidatos em uma lista ordenada (LC) de forma crescente. Em seguida, constrói-se uma lista restrita de candidatos (LRC), usando a expressão:

$$LRC = \{i \in N \mid g(i) \leq g_{\min} + \alpha \times (g_{\max} - g_{\min})\} \quad (11)$$

em que $g_{\min}$ e $g_{\max}$ são, respectivamente, o menor e o maior valor da lista ordenada LC; o parâmetro $\alpha \in [0, 1]$ (cujo valor deve ser definido por meio de *software* para calibração de

parâmetros) mensura a forma de escolha do candidato a entrar na lista (mais aleatória ou mais gulosa). Então, a partir da lista restrita de candidatos LRC, escolhe-se aleatoriamente as p instalações que serão medianas. Finalmente, aloca-se as instalações clientes às medianas, respeitando a capacidade de atendimento das mesmas. Caso o número de tentativas de construção da solução ($NMax$) seja atingido, é retornada uma solução vazia.

O Algoritmo 1 mostra o pseudocódigo do procedimento GRASP utilizado para a construção da solução inicial.

Algoritmo 1 Algoritmo GRASP para construção da solução inicial

Entrada: $M, N, m, n, p, NMax, \alpha, g(.)$
Saída: $s^0 = (v_m, v_a)$ ▷ Vetor de medianas e vetor de alocações

- 1: **para** $i \leftarrow 1$ até n **faça**
- 2: Calcula as somas ordenadas das distâncias acumuladas usando expressão 10
- 3: Constrói a LRC por meio da expressão 11
- 4: **fim para**
- 5: $s^0 \leftarrow \emptyset$ ▷ Solução inicial vazia
- 6: $NMax \leftarrow 0$ ▷ Iniciando a contagem das tentativas
- 7: $v_m \leftarrow \emptyset$ ▷ Iniciando o vetor de medianas
- 8: $k \leftarrow 0$ ▷ Iniciando a contagem das medianas
- 9: **enquanto** $k \leq p$ **faça**
- 10: $k \leftarrow k + 1$
- 11: Selecione aleatoriamente uma instalação $r \in LRC$ para ser mediana
- 12: Adicione a mediana r na k -ésima posição do vetor de medianas: $v_m[k] \leftarrow r$
- 13: **fim enquanto**
- 14: **enquanto** $M \neq \emptyset$ **faça**
- 15: Selecione aleatoriamente um cliente $j \in M$ e atribua à mediana r mais próxima que tenha capacidade de atendimento
- 16: $M \leftarrow M \setminus \{j\}$ ▷ Atualiza o vetor de clientes
- 17: **se** Tentativas $\leq NMax$ **então**
- 18: Retorna à linha 10
- 19: Tentativas $\leftarrow$ Tentativas +1
- 20: **senão**
- 21: **Retorne** $s^0 = \emptyset$, ou seja, não foi encontrada solução viável
- 22: **fim se**
- 23: **fim enquanto**

Saída: Vetor de medianas (v_m) e vetor de alocações (v_a)

3.3. Estrutura de Vizinhaça e Busca Local

Uma heurística de busca local é um procedimento de refinamento de uma solução s que utiliza uma ou mais estruturas de vizinhaça para encontrar um ótimo local com relação às estruturas utilizadas na exploração do espaço de soluções. Neste trabalho é proposto o uso da heurística *Variable Neighborhood Descent* (VND) [Hansen e Mladenovic, 2003] como procedimento de busca local, usando a estratégia do Melhor Vizinho (*Best Improvement*), em que a busca é feita em toda a vizinhaça da solução corrente s . Após identificar o melhor vizinho da solução s , com melhor valor de função objetivo, este melhor vizinho se torna a nova solução corrente. Neste caso, repete-se o procedimento de exploração da vizinhaça da nova solução. Caso contrário, o método é encerrado e retorna um ótimo local em relação à estrutura de vizinhaça. O Algoritmo 2 mostra o pseudocódigo do algoritmo VND usado como método de busca local para refinamento da solução inicial visando obter um ótimo local.

Para explorar o espaço de soluções do problema, os seguintes movimentos são usados para definir as estruturas de vizinhança:

- (i) M_1 - **Realocação**: Consiste em realocar um cliente j de uma mediana $r_1 \in N_p$ em uma outra mediana $r_2 \in N_p$;
- (ii) M_2 - **Substituição intra-cluster**: Consiste em substituir uma mediana $r_1 \in N_p$ por uma instalação não mediana r_2 que seja candidata do mesmo *cluster*, $r_2 \in N \setminus N_p$;
- (iii) M_3 - **Substituição inter-cluster**: Consiste em substituir uma mediana $r_1 \in N_p$ por uma instalação não mediana r_2 que seja candidata de outro *cluster*, $r_2 \in \{N - N_p\}$;
- (iv) M_4 - **Troca**: Consiste em trocar as alocações de um cliente j_1 de uma mediana $r_1 \in N_p$ com um cliente j_2 de uma outra mediana $r_2 \in N_p$.

Assim, o movimento M_j dá origem à estrutura de vizinhança N_j , que é o conjunto de todos os vizinhos que podem ser obtidos pela aplicação do movimento.

Algoritmo 2 Algoritmo de Busca Local

Entrada: $s, f(\cdot), N(s), n, k_{max}$

```

1:  $k \leftarrow 1$                                 ▷ Iniciando com a estrutura de vizinhança  $M_1$ 
2: enquanto  $k \leq k_{max}$  faça
3:    $s' \leftarrow$  Gera um vizinho de  $s$  com base no movimento da estrutura de vizinhança
4:   se  $f(s') < f(s)$  então
5:      $s \leftarrow s'$                                 ▷ Atualiza a solução corrente
6:      $k \leftarrow 1$                                 ▷ Retorna para a primeira estrutura de vizinhança
7:   senão
8:      $k \leftarrow k + 1$                             ▷ Passe para a próxima estrutura de vizinhança
9:   fim se
10: fim enquanto
Saída: Solução  $s$  refinada

```

3.4. Perturbação

Segundo Lourenço et al. [2002], diferentes ótimos locais podem ser gerados a partir de alterações na solução corrente por meio de um mecanismo aleatório que permita ir para vizinhanças intermediárias. Neste procedimento, denominado **perturbação**, efetua-se gradativamente uma sequência aleatória de movimentos, de modo a reiniciar a busca local em regiões mais promissoras.

A presente implementação utiliza três níveis de perturbação para ampliar o espaço de buscas. No nível 1, serão executados dois movimentos de troca; no nível 2, serão executados três movimentos de troca; e, no nível 3, serão executados quatro movimentos de troca. O Algoritmo 3 apresenta o pseudocódigo para o procedimento de perturbação de nível igual a k .

3.5. Meta-heurística *Iterated Local Search* (ILS)

A meta-heurística *Iterated Local Search* (ILS), proposta por Lourenço et al. [2002], consiste em explorar o espaço de busca por meio de perturbações em ótimos locais. Os desenvolvedores desse método buscavam uma técnica que fosse adaptável a diferentes tipos de problemas, garantindo a sua eficiência. O ILS combina, portanto, intensificação, com a

Algoritmo 3 Perturbação

Entrada: Solução refinada s , Vizinhança M_4 , Nível k de perturbações

- 1: $s' \leftarrow s$ ▷ Copie a solução corrente
 - 2: $i \leftarrow 1$ ▷ Inicie o contador de perturbações
 - 3: **para** $i \leq k$ **faça** ▷ Aplique o movimento M_4 em s'
 - 4: Selecione de forma aleatória s'' vizinha de s'
 - 5: $s' \leftarrow s''$ ▷ Atualize a solução após perturbação
 - 6: $i \leftarrow i + 1$ ▷ Atualize o contador de perturbações
 - 7: **fim para**
- Saída:** s'
-

exploração da região atual de busca através de busca local, e diversificação, aumentando-se gradativamente a quantidade de perturbações na solução ótima local corrente.

O método tenta melhorar o valor da função objetivo usando uma mudança do ponto de partida da busca local. Uma alternativa seria selecionar aleatoriamente o novo ponto de partida, mas o custo seria demasiadamente alto para instâncias de maiores dimensões. Em contrapartida, apenas a busca local, sem a perturbação, não consegue explorar suficientemente o espaço de busca, ocasionando um método que fica preso em ótimos locais. Baseado nisso, o ILS foi desenvolvido com um procedimento de busca estocástica que combina as duas técnicas, ou seja, o reinício em um novo ponto e a aplicação de busca local para a determinação de uma nova solução.

Algoritmo 4 ILS - Iterated Local Search

Entrada: $f(\cdot)$, $\mathcal{N}(\cdot)$, $NumIterMax$, $nivelPert$, s

- 1: $s^0 \leftarrow SolucaoInicial$; ▷ Construir $s^0 = (v_m, v_a)$ conforme Algoritmo 1
 - 2: $s \leftarrow BuscaLocal$; ▷ Aplicar busca local em s^0 conforme Algoritmo 2
 - 3: $iter \leftarrow 0$ ▷ Iniciando o contador de iterações
 - 4: $nivelPert \leftarrow 1$ ▷ Iniciando o contador de nível de perturbação
 - 5: **enquanto** $iter \leq NumIterMax$ **faça**
 - 6: $iter \leftarrow iter + 1$
 - 7: $s' \leftarrow Perturbacao$ ▷ Perturbar s' conforme Algoritmo 3
 - 8: $s'' \leftarrow BuscaLocal$ ▷ Aplicar busca local em s'' conforme Algoritmo 2
 - 9: **se** $f(s'') < f(s)$ **então**
 - 10: $s \leftarrow s''$; ▷ Atualiza a solução corrente
 - 11: $nivelPert \leftarrow 1$; ▷ Retorna ao nível 1
 - 12: **senão**
 - 13: $nivelPert \leftarrow nivelPert + 1$ ▷ Passa ao próximo nível
 - 14: **fim se**
 - 15: **fim enquanto**
- Saída:** s
-

Um pseudocódigo do ILS para um problema de minimização é apresentado no Algoritmo 4. Neste algoritmo, uma solução inicial s_0 é gerada (linha 1), sendo refinada com a estratégia de busca local do Melhor Vizinheiro (linha 2). No laço principal de repetição do algoritmo (linhas 5-15), que para após $NumIterMax$ iterações sem melhora, efetua-se uma perturbação na solução corrente (na linha 7) e uma busca local na solução perturbada (linha 8), obtendo-se s'' . Caso a solução s'' tenha melhorado o valor de função de avaliação,

ela passa a ser a solução corrente (linha 10), o nível de perturbação ($nivelPert$) retorna para 1. Caso não haja melhora, incrementa-se em 1 unidade o contador $nivelPert$ (linha 13), reiniciando o processo até o número de iterações alcançar o valor de $NumIterMax$.

4. Experimentos Computacionais

Esta seção apresenta os resultados dos experimentos computacionais realizados, com a seguinte organização: a Subseção 4.1 descreve os conjuntos de instâncias utilizados nos experimentos; a Subseção 4.2 relata o procedimento para calibração dos parâmetros do algoritmo; por fim, a Subseção 4.3 mostra as tabelas com os resultados dos experimentos.

4.1. Instâncias da Literatura

Para os testes computacionais, foram utilizados nove conjuntos de dados, variando de 318 instalações e 5 medianas até 4.461 instalações e 1.000 medianas. A instância é identificada pelo número de instalações candidatas, seguida do número de medianas a serem escolhidas. Assim, foram utilizados os conjuntos de dados *lin318*, *ali535*, *u724*, *rl1304*, *pr2392* e *fnl4461*, introduzidos por Stefanello et al. [2015], além do conjunto *p3038*, introduzido por Lorena e Senne [2004]. Cada conjunto possui 5 instâncias, totalizando 35 instâncias.

Cada arquivo fornece o número de instalações n e de medianas p , as coordenadas cartesianas das instalações, a capacidade Q_i e a demanda q_i de cada instalação.

Os experimentos foram executados em um computador 7ª Geração do Intel(R) Core(TM) i5-7200U, 2.50 GHz, com 20 GB de RAM, placa de vídeo NVIDIA GeForce 940MX 2GB, utilizando o ambiente Linux Ubuntu 22.04. O algoritmo ILS foi implementado na linguagem C++ e, para o cálculo das soluções exatas, foi utilizado o *solver* CPLEX 22.1.1, configurado com uma *thread* e demais configurações padrão, por meio da biblioteca *Concert Technology* usando a linguagem C++.

Os resultados para esses conjuntos de dados e a expressão para cálculo do *gap* são mostrados na Subseção 4.3.

4.2. Configurações dos Parâmetros do Algoritmo ILS aplicado ao PPMC

A calibração dos parâmetros a serem utilizados na execução do algoritmo implementado foi feita por meio da ferramenta *Iterated Racing for Automatic Algorithm Configuration* - (IRACE) [López-Ibáñez et al., 2016]. Para a calibração, foram utilizadas as instâncias *lin318_5*, *lin318_70*, *ali535_25*, *ali535_50* e *u724_75*, que representam 14% do total das instâncias testadas. Foi fixado um número máximo de 5000 execuções e demais configurações *default*. A Tabela 1 mostra os resultados retornados pela ferramenta.

Tabela 1: Valores retornados pelo IRACE para os parâmetros do Algoritmo ILS.

Parâmetro	Descrição	Faixa de Valor	Valor retornado
α	Valor utilizado na LRC	[0,1; 0,9]	0,6
$NumIterMax$	Número máximo de iterações sem melhora	{10, 20, 50, 100}	20

4.3. Resultados dos Experimentos

Para as instâncias com até 1304 instalações candidatas, o problema foi resolvido usando o *solver* CPLEX e o algoritmo ILS. As instâncias de maior dimensão foram solucionadas unicamente pelo algoritmo ILS. Como critério de parada para a resolução das formulações no *solver* CPLEX foi estabelecido tempo máximo de 129.600s. Para cada instância, no algoritmo ILS, foram realizadas 30 execuções. Dos valores retornados, foram utilizados o melhor valor da função objetivo na Tabela 2 e os valores médios da função objetivo e de tempo nas Tabelas 3 e 4.

Tabela 2: Comparação dos melhores valores para a função objetivo obtidos pelo Algoritmo ILS, pelo CPLEX, por Stefanello et al. [2015] e por Vasconcelos [2023]

Instâncias		Algoritmo ILS	Referências			gap(%)		
Grupo	p		CPLEX	Stefanello	Vasconcelos	CPLEX	Stefanello	Vasconcelos
lin318	5	180.343,00	180.281,21	180.281,2	180.281,2	0,03	0,03	0,03
	15	89.041,59	88.901,56	88.901,56	89.026,28	0,16	0,16	0,02
	40	48.091,32	47.988,38	47.988,38	47.982,75	0,21	0,21	0,23
	70	32.293,75	32.198,64	32.290,39	32.201,85	0,30	0,01	0,29
	100	23.041,87	22.942,69	22.942,69	22.971,72	0,43	0,43	0,31
ali535	5	10.003,21	9.956,77	9.956,77	9.933,43	0,47	0,47	0,70
	25	3.702,10	3.695,15	3.695,15	3.688,51	0,19	0,19	0,37
	50	2.503,34	2.461,41	2.464,01	2.458,70	1,70	1,60	1,82
	100	1.462,58	1.438,42	1.441,26	1.433,6	1,68	1,48	2,02
	150	1.036,85	1.032,28	1.035,52	1.025,37	0,44	0,13	1,12
u724	10	182.036,23	181.782,96	181.782,96	181.823,17	0,14	0,14	0,12
	30	95.201,56	95.034,01	95.034,01	95.222,02	0,17	0,17	-0,02
	75	54.912,78	54.735,05	54.735,05	54.913,01	0,32	0,32	0,00
	125	39.127,45	38.976,76	38.976,76	39.196,73	0,39	0,39	-0,18
	200	28.719,14	28.079,97	28.079,97	28.340,10	2,28	2,28	1,34
rl1304	10	2.158.869,31	2.146.484,10	2.146.484,10	2.221.470,50	0,58	0,58	-2,82
	50	803.974,59	802.283,41	803.106,89	803.150,33	0,21	0,11	0,10
	100	498.516,64	495.925,93	498.093,12	497.796,79	0,52	0,09	0,14
	200	277.698,41	276.977,60	276.977,60	276.310,23	0,26	0,26	0,50
	300	193.214,26	191.224,85	191.224,85	191.544,95	1,04	1,04	0,87
pr2392	20	2.244.317,72	–	2.236.720,55	2.332.221,94	–	0,34	-3,77
	75	1.099.723,42	–	1.094.979,83	1.132.362,51	–	0,43	-2,88
	150	713.906,38	–	711.128,32	733.273,63	–	0,39	-2,64
	300	459.906,71	–	458.157,54	475.276,82	–	0,38	-3,23
	500	317.689,13	–	316.046,35	321.174,72	–	0,52	-1,09
fnl4461	20	1.300.603,87	–	1.283.536,73	1.311.798,15	–	1,33	-0,85
	100	551.321,58	–	548.909,01	560.030,68	–	0,44	-1,56
	250	336.789,92	–	335.888,87	339.654,89	–	0,27	-0,84
	500	228.994,53	–	224.664,24	232.173,69	–	1,93	-1,37
	1000	146.508,37	–	145.862,47	150.366,44	–	0,44	-2,57
p3038	600	122.918,75	–	122.711,17	122.752,19	–	0,17	0,14
	700	110.023,64	–	109.677,30	109.381,22	–	0,32	0,59
	800	100.428,31	–	100.064,94	99.054,98	–	0,06	1,08
	900	92.587,42	–	92.310,09	90.969,96	–	0,30	1,78
	1000	86.153,89	–	85.854,05	83.693,38	–	0,35	2,94
Médias	–	361.010,39	235.120,06	359.200,11	367.855,90	0,58	0,51	-0,21

A Tabela 2 compara os resultados dos melhores valores para a função objetivo obtidos pelo algoritmo ILS, com os valores ótimos encontrados pelo *solver* CPLEX, com os melhores valores descritos em Stefanello et al. [2015] e em Vasconcelos [2023]. Os resultados mostram que o algoritmo ILS obteve desempenho muito bom, visto que o *gap* médio ficou abaixo de 1% em relação aos valores do CPLEX e os descritos em Stefanello et al. [2015] e superou os valores descritos em Vasconcelos [2023]. Para o cálculo do *gap*, em valores percentuais, tem-se:

$$gap\ ILS(\%) = \frac{Valor\ obtido\ no\ ILS - Valor\ referência}{Valor\ referência} \times 100.$$

Na Tabela 3 estão as comparações dos valores médios da função objetivo obtidos por meio do Algoritmo ILS com os valores ótimos do *solver* CPLEX, os valores médios descritos em Stefanello et al. [2015] e em Vasconcelos [2023]. Os resultados mostram que o algoritmo *ILS* obteve desempenho muito bom, visto que o *gap* médio ficou abaixo de 2% em relação aos valores descritos nas referências.

Na Tabela 4 estão os tempos médios de execução, em segundos, do Algoritmo ILS comparados com os tempos para as duas formulações do CPLEX e os tempos médios relatados em Stefanello et al. [2015] e em Vasconcelos [2023]. Os resultados mostram que o ILS superou os tempos de execução do CPLEX e obteve tempo praticamente igual aos tempos

Tabela 3: Comparação dos valores médios para a função objetivo obtidos pelo algoritmo ILS, por Stefanello et al. [2015], por Vasconcelos [2023] e valores ótimos do CPLEX

Instâncias		Algoritmo ILS	Referências			gap(%)		
Grupo	p		CPLEX	Stefanello	Vasconcelos	CPLEX	Stefanello	Vasconcelos
lin318	5	181.154,00	180.281,21	180.281,20	180.284,20	0,48	0,48	0,48
	15	89.205,59	88.901,56	88.901,56	88.905,20	0,34	0,34	0,34
	40	48.129,93	47.988,38	48.003,88	47.995,69	0,29	0,26	0,28
	70	32.405,49	32.198,64	32.290,39	32.200,53	0,64	0,36	0,64
	100	23.208,57	22.942,69	22.942,69	23.168,12	1,16	1,16	0,17
ali535	5	10.356,43	9.956,77	10.210,58	9.956,60	4,01	1,43	4,02
	25	3.826,45	3.695,15	3.701,88	3.695,22	3,63	3,45	3,63
	50	2.539,37	2.461,41	2.478,04	2.510,21	3,17	2,47	1,16
	100	1.502,58	1.438,42	1.448,00	1.451,92	4,46	3,77	3,49
	150	1.078,35	1.032,28	1.037,70	1.032,45	4,46	3,92	4,45
u724	10	182.971,52	181.782,96	182.611,19	181.586,01	06,5	0,20	0,76
	30	95.821,23	95.034,01	95.159,96	95.043,90	0,83	0,69	0,82
	75	55.152,13	54.735,05	54.735,50	54.737,77	0,76	0,76	0,76
	125	39.614,61	38.976,76	38.976,76	38.976,60	1,64	1,64	1,64
	200	28.897,76	28.079,97	28.082,72	28.592,14	2,91	2,90	1,07
rl1304	10	2.225.719,83	2.146.484,10	2.166.552,03	2.158.039,60	3,69	2,73	3,14
	50	809.531,15	802.283,41	806.425,28	799.718,88	0,90	0,39	1,23
	100	499.324,28	495.925,93	498.411,69	495.945,74	0,69	0,18	0,68
	200	278.316,37	276.977,60	276.983,47	277.477,09	0,48	0,48	0,30
	300	193.713,08	191.224,85	191.400,09	192.995,41	1,30	1,21	0,37
pr2392	20	2.336.529,16	–	2.250.292,41	2.235.448,34	–	3,83	4,52
	75	1.134.301,54	–	1.098.559,96	1.092.362,69	–	3,25	3,84
	150	714.281,51	–	711.656,01	711.157,29	–	0,37	0,44
	300	463.987,13	–	458.298,21	458.170,55	–	1,24	1,27
	500	319.698,34	–	316.230,72	316.056,12	–	1,10	1,15
fnl4461	20	1.314.504,75	–	1.292.621,57	1.296.295,22	–	1,69	1,40
	100	560.981,32	–	550.758,21	548.917,23	–	1,86	2,20
	250	340.197,14	–	336.006,96	335.911,49	–	1,25	1,28
	500	232.506,71	–	224.684,37	228.165,55	–	3,48	1,90
	1000	150.584,92	–	145.870,78	147.703,03	–	3,23	1,95
p3038	600	125.098,61	–	122.724,49	–	–	1,93	–
	700	111.086,21	–	109.695,61	–	–	1,27	–
	800	101.104,56	–	100.084,41	–	–	1,02	–
	900	93.287,42	–	92.317,78	–	–	1,05	–
	1000	86.934,94	–	85.854,05	–	–	1,26	–
Médias	–	368.215,89	235.120,06	360.751,24	402.816,69	1,83	1,62	1,65

citados em Vasconcelos [2023]. Entretanto, o desempenho foi ineficiente quando comparado com os resultados descritos em Stefanello et al. [2015].

5. Considerações Finais

Os resultados obtidos com os experimentos computacionais após a aplicação do Algoritmo ILS proposto para resolução do problema das p -medianas capacitado, usando as instâncias descritas na Subseção 4.1, mostram um bom desempenho da meta-heurística na obtenção de boas soluções. Tanto os melhores valores quanto os valores médios para a função objetivo ficaram muito próximos dos valores ótimos obtidos através do *solver* CPLEX e daqueles descritos em Stefanello et al. [2015] e em Vasconcelos [2023]. Quanto ao tempo de execução, o algoritmo obteve um desempenho muito bom quando comparado com o CPLEX e com Vasconcelos [2023], mas o desempenho não foi satisfatório quando comparado ao que está descrito em Stefanello et al. [2015].

Como trabalho futuro, sugere-se fazer alterações no algoritmo visando obter *gap* igual a zero para a função objetivo e minimizar o tempo de execução.

Agradecimentos

Os autores agradecem à Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), ao Conselho Nacional de Desenvolvimento Científico e Tecnológico

Tabela 4: Comparação dos valores médios dos tempos de execução gastos pelo algoritmo ILS, por Stefanello et al. [2015], por Vasconcelos [2023] e os tempos do CPLEX

Grupo	p	Tempos de Execução (segundos)					gap(%)			
		Algoritmo ILS	Formulação 1	Formulação 2	Stefanello	GG-VND	Formulação 1	Formulação 2	Stefanello	GG-VND
lin318	5	345,96	506,96	192,08	13,77	358,00	-31,76	80,11	2.413,19	-3,36
	15	598,87	685,57	352,19	39,64	574,00	-12,65	70,04	1.410,90	4,33
	40	840,59	7.700,33	16.217,65	480,64	853,00	-89,08	-94,82	79,26	-1,45
	70	537,69	81,60	140,75	191,75	527,00	558,93	282,02	187,43	2,03
	100	754,01	623,59	7.671,05	548,96	752,00	20,91	-90,17	40,79	0,27
ali535	5	2.348,63	17.576,43	1552,49	68,33	2.342,00	-86,64	51,28	3.337,36	0,28
	25	2.586,15	5.362,16	6.323,09	818,16	2.612,00	-51,77	-59,10	215,82	-0,99
	50	2.387,95	21.798,01	10.514,64	1092,75	2.320,00	-89,05	-77,29	118,53	2,93
	100	2.147,16	17.818,09	8.061,55	959,36	2.113,00	-87,95	-73,37	123,81	1,62
	150	2.546,97	19.553,91	9.936,63	1.145,43	2516,00	-86,97	-74,37	122,36	1,23
u724	10	1.023,17	53.783,90	1.399,61	89,74	1.017,00	-98,10	-26,90	1.040,17	0,61
	30	1.528,38	13.454,04	1.324,85	452,45	1.525,00	-88,64	15,36	237,80	0,22
	75	1.349,53	3.284,89	2.045,93	822,06	1.324,00	-58,92	-34,04	64,16	1,93
	125	1.674,26	9.165,77	3.492,87	967,89	1.675,00	-81,73	-52,07	72,98	-0,04
	200	2.019,60	2.798,56	1.764,63	1.062,64	2.114,00	-24,62	19,55	98,52	-0,21
rl1304	10	5.413,34	101.232,73	19.697,36	273,32	5.385,00	-94,65	-72,52	1.880,61	0,53
	50	3.785,08	49.837,61	4.647,09	1.805,42	3.715,00	-92,41	-18,55	109,65	1,89
	100	3.243,15	21.907,97	4.123,53	2.458,70	3.198,00	-85,20	-21,35	31,91	1,41
	200	4.209,47	10.856,25	3.795,19	1.847,24	4.179,00	-61,23	10,92	127,88	0,73
	300	5.394,30	6.811,78	19.107,83	1.431,96	5.411,00	-20,81	-71,77	276,71	-0,31
pr2392	20	3.852,16	—	—	830,24	3.796,00	—	—	363,98	1,48
	75	3.979,38	—	—	1.242,53	3.981,00	—	—	220,27	-0,04
	150	4.512,67	—	—	3.038,03	4.438,00	—	—	48,54	1,68
	300	3.629,41	—	—	3.584,41	3.635,00	—	—	48,54	-0,15
	500	4.763,56	—	—	3.614,83	4.717,00	—	—	31,78	0,99
fm14461	20	8.523,18	—	—	810,90	8.522,00	—	—	951,08	0,01
	100	9.046,80	—	—	5.838,20	9.115,00	—	—	54,96	-0,75
	250	9.219,85	—	—	6.909,87	9.411,00	—	—	33,43	-2,03
	500	9.804,23	—	—	5.886,33	9.781,00	—	—	66,56	0,24
	1.000	9.102,87	—	—	5.065,50	9.251,00	—	—	76,22	-1,60
p3038	600	4.603,21	—	—	2.752,51	4.549,00	—	—	67,42	1,19
	700	4.309,72	—	—	2.295,84	4.316,00	—	—	87,72	-0,15
	800	5.197,64	—	—	2.889,74	5.283,00	—	—	79,87	-1,62
	900	5.427,38	—	—	1.617,62	5.384,00	—	—	235,52	0,81
	1000	5.673,22	—	—	1.920,93	5.688,00	—	—	195,34	-0,26
Médias	—	3.899,13	18.242,01	6.094,55	1.810,96	3.896,49	-33,12	-11,85	413,96	0,38

(CNPq), à Fundação de Apoio à Pesquisa do Estado de Minas Gerais (FAPEMIG), ao Centro Federal de Educação Tecnológica de Minas Gerais (CEFET-MG) e ao Instituto Federal de Educação, Ciência e Tecnologia de Minas Gerais (IFMG) pelo apoio ao desenvolvimento deste trabalho.

Referências

- Avella, P., Boccia, M., Salerno, S., e Vasilyev, I. (2012). An aggregation heuristic for large scale p-median problem. *Computers & Operations Research*, 39(7):1625–1632.
- Avella, P., Sassano, A., e Vasil'ev, I. (2007). Computational study of large-scale p-median problems. *Mathematical Programming*, 109:89–114.
- Bektaş, T., Gouveia, L., e Santos, D. (2019). Revisiting the hamiltonian p-median problem: A new formulation on directed graphs and a branch-and-cut algorithm. *European Journal of Operational Research*, 276(1):40–64.
- Ceselli, A. e Righini, G. (2005). A branch-and-price algorithm for the capacitated p-median problem. *Networks: An International Journal*, 45(3):125–142.
- Drezner, E. (1996). Facility location: A survey of applications and methods. *Journal of the Operational Research Society*, 47(11):1421–1421.
- Feo, T. A. e Resende, M. G. (1995). Greedy randomized adaptive search procedures. *Journal of global optimization*, 6:109–133.

- Fisher, M. L., Jaikumar, R., e Van Wassenhove, L. N. (1986). A multiplier adjustment method for the generalized assignment problem. *Management science*, 32(9):1095–1103.
- Hakimi, S. L. (1964). Optimum locations of switching centers and the absolute centers and medians of a graph. *Operations research*, 12(3):450–459.
- Hansen, P. e Mladenovic, N. (2003). A tutorial on variable neighborhood search. *Les Cahiers du GERAD ISSN*, 711:2440.
- Herda, M. (2015). Combined genetic algorithm for capacitated p-median problem. In *2015 16th IEEE International Symposium on Computational Intelligence and Informatics (CINTI)*, p. 151–154. IEEE.
- López-Ibáñez, M., Dubois-Lacoste, J., Cáceres, L. P., Birattari, M., e Stützle, T. (2016). The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, 3:43–58.
- Lorena, L. A. e Senne, E. L. (2004). A column generation approach to capacitated p-median problems. *Computers & Operations Research*, 31(6):863–876.
- Lourenço, H., Martin, O., e Stützle, T. (2002). Iterated local search. In Glover, F. e Kochenberger, G., editors, *Handbook of Metaheuristics*, p. 43–58. International Series in Operations Research & Management Science.
- Stefanello, F., de Araújo, O. C., e Müller, F. M. (2015). Matheuristics for the capacitated p-median problem. *International Transactions in Operational Research*, 22(1):149–167.
- Vasconcelos, A. M. (2023). *Algoritmo GVNS Híbrido Aplicado ao Problema das p-Mediana Capacitado*. Doutorado, Programa de Pós-Graduação em Modelagem Matemática e Computacional, Centro Federal de Educação Tecnológica de Minas Gerais, Belo Horizonte.
- Yaghini, M., Karimi, M., e Rahbar, M. (2013). A hybrid metaheuristic approach for the capacitated p-median problem. *Applied soft computing*, 13(9):3922–3930.