

Algoritmo *Clustering Search* aplicado ao Problema de Localização com Cobertura Parcial

Jéssica Costa

Programa de Engenharia de Sistemas e Computação
Universidade Federal do Rio de Janeiro
jessicacosta@cos.ufrj.br

Gabriel Souto

Programa de Engenharia de Sistemas e Computação
Universidade Federal do Rio de Janeiro
soutog@cos.ufrj.br

Laura Bahiense

Programa de Engenharia de Sistemas e Computação
Universidade Federal do Rio de Janeiro
laura@cos.ufrj.br

Pedro Henrique González

Programa de Engenharia de Sistemas e Computação
Universidade Federal do Rio de Janeiro
pegonzalez@cos.ufrj.br

RESUMO

O Problema de Localização com Cobertura Parcial consiste na escolha de um conjunto de facilidades geograficamente localizadas de modo a minimizar seu custo total de abertura e garantir que determinada quantidade parcial de demanda dos clientes seja coberta. Este artigo apresenta um algoritmo *Clustering Search* (CS) associado à metaheurística *Generalized Simulated Annealing* (GSA) para resolver esse problema. No experimento computacional, foram testadas instâncias com 10.000 e 50.000 clientes encontradas na literatura utilizando o algoritmo CS-GSA. Os resultados obtidos mostram que o CS-GSA apresentou um comportamento superior ao SA.

PALAVRAS CHAVE. Problema de Localização com Cobertura Parcial, *Generalized Simulated Annealing*, *Clustering Search*.

Tópicos: SE-CRIS, Logística e Transportes.

ABSTRACT

The Partial Set Covering Location Problem (PSCLP) consists of choosing a set of facilities geographically located to minimize their total opening cost and ensure that a certain partial amount of customer demand is covered. This article presents a *Clustering Search* algorithm (CS) associated with the *Generalized Simulated Annealing* metaheuristic (GSA) to solve this problem. In the computational experiment, instances with 10.000 and 50.000 clients found in the literature were tested using the CS-GSA algorithm. The results show that CS-GSA performed better than SA.

KEYWORDS. Partial Set Covering Location Problem, *Generalized Simulated Annealing*, *Clustering Search*.

Paper Topics: SE-CRIS, Logistics and Transport.

1. Introdução

A ausência de um planejamento logístico adequado na escolha do local para instalação de um conjunto de facilidades pode acarretar consequências não apenas econômicas desfavoráveis para uma empresa, mas também impactos negativos significativos no ambiente e na comunidade. Isso se deve à considerável capacidade de emissão de poluentes inerente à instalação e ao funcionamento das facilidades, bem como ao transporte relacionado aos clientes atendidos.

A determinação da localização das facilidades ao longo da cadeia de suprimentos é uma decisão crucial, com implicações financeiras e ambientais significativas. Este é um desafio complexo, pois influencia diretamente a eficiência operacional e a satisfação do cliente. Em essência, o objetivo é identificar os locais estratégicos onde as facilidades devem ser instaladas para fornecer serviços acessíveis aos clientes, otimizando critérios como o número de instalações necessárias, a cobertura da demanda dentro de uma distância específica e a eficiência da cobertura geográfica [Daskin e Owen, 1999; Arakaki e Lorena, 2006; García e Marín, 2015].

Quando o critério a ser otimizado é a cobertura da demanda dos clientes considerados na rede, o problema de localização de facilidades pode ser classificado como um problema de cobertura. Embora as primeiras abordagens sobre o problema de cobertura tenham sido introduzidas por Berge [1957]; Hakimi [1964], tal classe foi tratada como uma aplicação à área de localização de facilidades pela primeira vez em Toregas et al. [1971]. Nesse contexto, a literatura descreve duas abordagens para problemas de cobertura: (i) a cobertura máxima, que visa localizar um número limitado de facilidades de modo a maximizar a quantidade de clientes cobertos; e (ii) a cobertura de custo mínimo, cujo objetivo é minimizar o custo total de localização de um conjunto de instalações de modo que todos os clientes sejam cobertos [Church e Reville, 1974].

Entretanto, nem sempre a satisfação da demanda de todos os clientes garante a eficiência operacional, pois a adição de um único cliente à rede pode resultar em um aumento no número de facilidades abertas, podendo causar impactos econômicos negativos. Além disso, há situações em que não há a necessidade de cobrir igualmente todos os clientes, como por exemplo, a instalação de pontos de distribuição logística [Church e Reville, 1974; García e Marín, 2015; Cardoso et al., 2023]. Para modelar esses casos, Daskin e Owen [1999] introduziram mais um problema no âmbito de conjunto de cobertura: o Problema de Localização com Cobertura Parcial - *Partial Set Covering Location Problem* (PSCLP).

Dos trabalhos pesquisados na literatura, poucos são aqueles em que os autores não utilizaram metaheurísticas para resolver o PSCLP. Cordeau et al. [2018] utilizaram *Benders Decomposition* e ainda criaram um conjunto de instâncias para realizar os testes. Liu et al. [2022] utilizaram *Ant Colony Optimization* e utilizaram instâncias disponibilizadas na *OR-Library*. Cardoso et al. [2022] utilizaram *Iterated Local Search* aproveitando as instâncias disponibilizadas por Cordeau et al. [2018]. Cardoso et al. [2023] implementaram duas versões de *Simulated Annealing*, uma sem calibração de parâmetros (*SA1*) e outra com os parâmetros calibrados pelo método *irace* [López-Ibáñez et al., 2016] (*SA2*). Chen et al. [2023] apresentaram cinco métodos de pré-processamento adaptados para aprimorar a capacidade de empregar resolvidores baseados em programação inteira mista na resolução do problema. Eles utilizaram quatro conjuntos de instâncias baseados em referências da literatura. Uma das referências foi Cordeau et al. [2018].

Uma técnica que tem sido aplicada a diversos problemas de otimização há alguns anos, mas que ainda não foi aplicada ao PSCLP é a de *Clustering Search* (CS) [Oliveira e Lorena, 2007]. O CS foi proposto como um método híbrido que combina uma metaheurística de busca com *clustering* a fim de detectar áreas de busca promissoras antes de aplicar procedimentos de busca local [De Oliveira et al., 2013]. Este trabalho apresenta a aplicação da técnica de CS combinada com a metaheurística *Generalized Simulated Annealing* para resolver o PSCLP.

O restante deste trabalho está organizado da seguinte forma: a Seção 2 apresenta o PSCLP. Na Seção 3, é apresentada a metodologia utilizada neste estudo. A Seção 4 aborda os experimentos computacionais realizados. Na Seção 5 são apresentados os Resultados Obtidos. Por fim, na Seção 6, são apresentadas as conclusões e trabalhos futuros.

2. Problema de Localização de Cobertura de Conjunto Parcial

O PSCLP consiste em encontrar um conjunto de facilidades cujo custo total de abertura seja mínimo, de modo a cobrir um determinado valor de demanda, que representa somente uma fração da demanda de todos os clientes presentes dentro do raio de cobertura das facilidades em potencial. Em outras palavras, o cliente pode estar dentro do raio de cobertura da facilidade e, mesmo assim, não ser atendido. A formulação matemática de Cordeau et al. [2018] para o PSCLP é apresentada pela Tabela 1 e pelo modelo a seguir:

Dados	
I	conjunto de facilidades i ;
J	conjunto de clientes j ;
f_i	custo de instalação da facilidade i ;
a_{ij}	matriz de cobertura;
d_j	quantidade de demanda de cada cliente j ;
D	demanda mínima a ser atendida;
Variáveis	
y_i	variável binária que vale 1 se a facilidade $i \in I$ estiver aberta, e vale 0, caso contrário;
z_j	variável binária que vale 1 se o cliente $j \in J$ for coberto por pelo menos uma facilidade, e vale 0, caso contrário.

Tabela 1: Notação

$$\text{Minimizar} \quad \sum_{i \in I} f_i y_i \quad (1)$$

$$\text{sujeito a:} \quad \sum_{i \in I} a_{ij} y_i \geq z_j, \quad \forall j \in J \quad (2)$$

$$\sum_{j \in J} d_j z_j \geq D, \quad (3)$$

$$y_i \in \{0, 1\}, \quad \forall i \in I \quad (4)$$

$$z_j \in \{0, 1\}, \quad \forall j \in J \quad (5)$$

A Função Objetivo (1) modela o cálculo do custo total de instalação de facilidades. A família de Restrições (2) garante que, caso haja uma instalação aberta, haverá pelo menos um cliente j coberto por tal facilidade dentro do seu raio de cobertura. A família de Restrições (3) garante que a fração de demanda dada por D seja atendida. As Restrições (4) e (5) definem o domínio das variáveis de decisão y_i e z_j . A seguir, será apresentada a metodologia proposta nesse trabalho para resolver o PSCLP modelado conforme acima.

3. Metodologia

Esta seção apresenta a metaheurística híbrida *Clustering Search-Generalized Simulated Annealing* (CS-GSA) para resolver o PSCLP. A organização desta seção ocorre de modo que a subseção 3.1 apresenta a representação da solução inicial. A Seção 3.2 apresenta como a solução inicial foi construída para o problema abordado. A subseção 3.3 apresenta o *Generalized Simulated Annealing* e suas subseções apresentam o critério de aceitação e o critério utilizado para a obtenção de soluções vizinhas respectivamente. Enquanto que na subseção 3.4 será apresentada a busca local

utilizada no CS-GSA. Por fim, a subseção 3.5 apresenta como acontece o processo de *Clustering Search* utilizando o algoritmo proposto de CS-GSA.

3.1. Representação da Solução

Neste trabalho, a solução para o PSCLP é representada pela composição dos vetores y e z , que funcionam como as variáveis de decisão descritas no modelo matemático na seção anterior. A dimensão do primeiro vetor equivale à $|I|$, sendo $i \in I$ uma facilidade candidata. Quando y_i vale 1 em uma dada posição $i \in I$, significa que a facilidade i está aberta. Por outro lado, quando vale 0, significa que a facilidade i está fechada. O outro vetor é o z que representa os clientes cobertos. A dimensão desse vetor equivale à $|J|$, sendo $j \in J$ um cliente candidato. Quando z_j vale 1 em uma dada posição $j \in J$, significa que o cliente j está coberto. Um exemplo de solução s considerando $|I| = 10$ e $|J| = 15$ é: $s = (y, z) = ([1, 1, 0, 0, 1, 0, 0, 1, 0, 1], [1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1])$.

3.2. Construção da Solução Inicial

O Algoritmo 1, de construção aleatória, utilizado para construir a solução inicial neste trabalho foi o mesmo descrito e utilizado por Cardoso et al. [2022, 2023]. O procedimento é iniciado com nenhuma facilidade $i \in I$ aberta e com nenhum cliente $j \in J$ coberto. Após esse passo, uma facilidade fechada é escolhida para ser aberta e todos os clientes cobertos pela facilidade aberta são assinalados com base no raio de cobertura. Como cada cliente está associado a uma demanda, as demandas referentes aos clientes cobertos definem a demanda total atendida pela solução corrente. O valor de demanda mínima a ser atendida D é dado por uma determinada porcentagem do somatório das demandas de todos os clientes, mesmo os que não estejam cobertos. Caso a demanda mínima seja atendida, o procedimento é encerrado, pois uma solução viável foi encontrada. Caso contrário, mais uma facilidade fechada é escolhida para ser aberta e repete-se esse processo.

Algorithm 1: *Construção Aleatória de Cardoso et al. [2022, 2023]*

```

1  $y \leftarrow \bar{0}$ 
2  $z \leftarrow \bar{0}$ 
3  $\bar{d} \leftarrow \bar{0}$ 
4  $C \leftarrow$  Conjunto de todas facilidades candidatas
5 while ( $\bar{d} < D$ ) do
6    $i \leftarrow$  Selecciona aleatoriamente uma facilidade em  $C$ 
7    $C \leftarrow C \setminus \{i\}$ 
8    $y_i \leftarrow 1$ 
9    $z_j \leftarrow 1$ , para todo  $j$  tal que  $a_{ij} = 1$ 
10   $\bar{d} \leftarrow$  demanda total coberta pelas facilidades abertas
11 end
12 return  $s = (y, z)$ 
13 end-Construção Aleatória
  
```

A solução gerada por esse algoritmo é utilizada para inicializar a solução inicial s do Algoritmo 2, que representa o CS-GSA, e será detalhado na seção seguinte.

3.3. Generalized Simulated Annealing

O *Generalized Simulated Annealing* (GSA) foi proposto por Tsallis e Stariolo [1996] com o intuito de permitir mais hibridizações e ser mais eficiente na busca de ótimos globais em funções de custo complexas e não convexas, inspirado no *Simulated Annealing* (SA) proposto por Kirkpatrick et al. [1983]. O SA simula o processo físico de aquecimento de um material e resfriamento lento com o intuito de escapar de mínimos locais. No entanto, o SA pode ter dificuldade para escapar de mínimos locais profundos devido à sua utilização de uma distribuição gaussiana para as visitas de estado e as estatísticas de Boltzmann-Gibbs.

O GSA generaliza o SA e o *Fast Simulated Annealing* (FSA) [Szu e Hartley, 1987], que emprega uma distribuição Cauchy-Lorentz para um resfriamento mais rápido e eficiente, permitindo passos maiores que ajudam a escapar de ótimos locais. A FSA é mais eficiente que o SA devido às suas caudas mais pesadas na distribuição de visitas, permitindo maior flexibilidade na busca. O GSA, inspirado na termostática generalizada [Tsallis, 1988], introduz dois parâmetros, q_v e q_A , que controlam as distribuições de visitas e aceitação de maneira mais eficaz.

Através do ajuste dos novos parâmetros introduzidos q_v (distribuição de visitação) e q_A (probabilidade de aceitação), o GSA proporciona uma metodologia que pode se adaptar a vários cenários de otimização, garantindo uma convergência mais rápida e uma maior probabilidade de encontrar mínimos globais em comparação com os métodos tradicionais como SA e FSA. Dessa forma, o GSA constitui-se uma técnica estocástica para abordar problemas complexos de otimização, oferecendo uma solução versátil para encontrar ótimos globais.

3.3.1. Critério de Aceitação do GSA

A probabilidade de aceitação no GSA é um componente chave que determina se um novo estado é aceito ou rejeitado. Este mecanismo garante que o algoritmo possa escapar dos mínimos locais e eventualmente convergir para o mínimo global. No SA, a probabilidade de aceitação é baseada no critério de Metropolis [Metropolis et al., 1953]:

- Se o novo estado x_{t+1} tiver uma energia (ou custo) $f(x_{t+1})$ menor que o estado atual $f(x_t)$, o movimento é sempre aceito; e
- Se o novo estado tiver uma energia mais alta, o movimento é aceito com uma probabilidade que diminui à medida que a diferença de energia $f(x_{t+1}) - f(x_t)$ aumenta e a temperatura (T_A) diminui.

Diferentemente, a probabilidade de aceitação do GSA $P_{q_A}(x_t \rightarrow x_{t+1})$ é definida da seguinte forma:

$$\begin{cases} 1 & , \text{ se } f(x_{t+1}) < f(x_t). \\ \left[1 + (q_A - 1) \frac{f(x_{t+1}) - f(x_t)}{T_A(t)} \right]^{\frac{1}{q_A - 1}} & , \text{ se } f(x_{t+1}) \geq f(x_t). \end{cases} \quad (6)$$

De modo similar ao SA, esta fórmula garante que movimentos que aumentam a energia sejam aceitos com uma probabilidade que diminui à medida que a diferença de energia aumenta, e o parâmetro q_A controla a forma desta distribuição de probabilidade, da seguinte forma:

- Quando $q_A \rightarrow 1$, a probabilidade de aceitação reduz-se à mesma utilizada no SA tradicional, no qual as temperaturas mais altas aceitam mais movimentos de piora:

$$P_1(x_t \rightarrow x_{t+1}) = \exp \left(-\frac{f(x_{t+1}) - f(x_t)}{T_A} \right); \text{ e}$$

- Quando $q_A = 2$, a probabilidade de aceitação segue uma distribuição diferente que permite resfriamentos mais rápidos:

$$P_2(x_t \rightarrow x_{t+1}) = \left[1 + \frac{f(x_{t+1}) - f(x_t)}{T_A} \right]^1,$$

e essa probabilidade decresce mais rapidamente com o aumento da diferença de energia, permitindo uma exploração mais rápida do espaço de busca, o que configura o FSA.

Assim, o parâmetro q_A ajusta o equilíbrio entre exploração e intensificação, permitindo também a escolha de diferentes valores de q_A baseado no problema e na velocidade de convergência desejada. Com isso, o GSA proporciona um mecanismo versátil e maior eficiência, ao combinar elementos do SA e do FSA em um *framework* unificado. Essa flexibilidade permite o GSA alcançar uma melhor performance e melhor convergência para uma gama de problema de otimização.

3.3.2. Obtenção de soluções vizinhas

Para a obtenção de soluções vizinhas, é preciso definir o número de vezes que ocorrerá a perturbação e quais operações de perturbação serão aplicadas na solução corrente. O número de perturbações é também conhecido como o número de passos que a solução corrente irá realizar até chegar em sua nova configuração. Neste trabalho, utilizamos apenas duas operações de perturbação. Uma é escolher uma facilidade fechada para ser aberta (*Open-facility*). A outra é escolher uma facilidade aberta para ser fechada (*Close-facility*). A escolha de abrir ou fechar uma facilidade ocorre de maneira aleatória numa determinada quantidade de vezes a cada iteração, ou seja, essa quantidade de vezes que as operações de perturbação serão chamadas é calculada a cada iteração do método. Para realizar o cálculo dessa quantidade de passos, utilizamos uma estratégia baseada na distribuição de *Cauchy*, como explicada a seguir.

A estratégia de cálculo da quantidade de passos busca equilibrar exploração e intensificação, considerando o progresso da iteração ($p_i = \frac{iter}{max_iter}$), a temperatura inicial (T_i), a temperatura corrente (T_A) e a qualidade da solução (i), de modo que combinando esse parâmetros é possível calcular o *scale* (λ), de acordo com a Equação 7. Vale ressaltar que esta escala (λ) diminui linearmente de uma escala inicial (λ_i) para uma escala final (λ_f), de acordo com a temperatura corrente:

$$\lambda = (\lambda_i \times (\frac{T_A}{T_i}) + \lambda_f \times (1 - (\frac{T_A}{T_i}))) \times (1 - p_i) \times (i \times p_i), \quad (7)$$

em que

$$i = \max \left(0, \frac{f(s^*) - f(s)}{\max(|f(s^*)|, 1 \times 10^{-10})} \right). \quad (8)$$

O valor de *scale* λ obtido é, então, utilizado para gerar o número de passos através da função de densidade de probabilidade (PDF) da distribuição de *Cauchy-Lorentz*:

$$f(x; x_0, \lambda) = \frac{1}{\pi \lambda \left[1 + \left(\frac{x - x_0}{\lambda} \right)^2 \right]}. \quad (9)$$

3.4. Busca Local

A busca local aplicada ao PSCLP é uma heurística utilizada para melhorar uma solução viável (s) por meio da exploração de vizinhanças. O objetivo é minimizar o custo total enquanto atende a uma demanda mínima. A busca começa com uma solução inicial, onde certas instalações estão abertas e outras fechadas. O custo inicial dessa configuração é calculado e definido como o melhor custo $f(s^*)$ encontrado até o momento.

O algoritmo então entra em um *loop* em que, em cada iteração, são geradas novas soluções vizinhas ao trocar o estado de instalações abertas (Fa) com fechadas (Ff), conforme a Figura 1. Para cada nova configuração gerada, o algoritmo verifica se ela é viável. Se a nova solução for viável e tiver um custo menor do que o melhor custo encontrado até então, ela é aceita como a nova melhor solução, e o processo de busca continua a partir dessa nova configuração.



Figura 1: Troca - Abertura da facilidade 1 e fechamento da facilidade 2

O laço de repetição continua até que nenhuma melhoria é encontrada. Nesse ponto, o algoritmo retorna a melhor configuração de instalações encontrada. Esse processo de troca e checagem de viabilidade, permite que a busca local explore eficientemente o espaço de soluções, procurando por melhores soluções, sem a necessidade de explorar todas as combinações possíveis, o que seria computacionalmente impraticável para problemas de grande escala.

Vale ressaltar que checagem de viabilidade assegura que todos os clientes cobertos por instalações abertas estejam efetivamente cobertos e que a soma das demandas dos clientes cobertos atenda a demanda.

3.5. CS-GSA

A metaheurística híbrida *Clustering Search* (CS) combina uma metaheurística auxiliar com técnicas de *clustering* para criar agrupamentos de soluções. Essas soluções são geradas inicialmente pela metaheurística auxiliar e depois agrupadas, permitindo que a busca local se concentre apenas nas regiões promissoras identificadas. Originalmente proposta por Oliveira e Lorena [2007], a técnica utilizava um algoritmo evolucionário para gerar soluções viáveis. Contudo, como mencionado em Souto et al. [2021], qualquer metaheurística pode ser empregada para gerar soluções para o CS. Nesse sentido, há na literatura diversos trabalhos que substituem o algoritmo evolucionário por distintas metaheurísticas como *GRASP* Chaves e Lorena [2007], *SA* em Chaves e Lorena [2010]; Rabello et al. [2013], *ALNS* em Souto et al. [2021] e diversos outros de Abreu et al. [2020]; González et al. [2019] para resolver problemas de otimização.

Apesar do CS ter sido explorado com diversas metaheurísticas para diferentes problemas de otimização, não havia registros do uso do GSA como gerador de soluções no contexto do PSCLP. Este artigo propõe, portanto, o uso do GSA, para desempenhar esse papel. Um *cluster* no CS é entendido como uma estrutura com um centro, representado pela melhor solução encontrada até o momento, e um volume, que é o número de soluções associadas a ele. À medida que novas soluções são incorporadas, o centro do *cluster* pode ser atualizado, e o volume é aumentado até atingir um limite máximo, indicando que o *cluster* está saturado.

No CS-GSA, cada solução obtida pelo GSA é analisada para determinar em qual *cluster* será incorporada, utilizando a distância de Hamming como medida de similaridade. A solução é adicionada ao *cluster* com menor distância, e se tiver um custo menor que o centro do *cluster*, ela se torna o novo centro. Quando o volume do *cluster* atinge o limite, a busca local é aplicada ao centro. Se uma solução melhor for encontrada, ela se torna a melhor solução global e o *cluster* é descartado. Esse processo se repete até que o critério de parada da metaheurística geradora de soluções seja atendido, garantindo uma busca eficiente por soluções ótimas.

Alguns trabalhos na literatura como em Oliveira e Lorena [2007]; Chaves e Lorena [2007, 2010]; De Oliveira et al. [2013] costumam organizar o processo de CS em 4 etapas: Meta-heurística (MH), Agrupador Iterativo (AI), Analisador de Agrupamentos (AA) e Busca Local (BL).

A metaheurística (MH) utilizada, neste caso, para gerar soluções para o processo de *clustering* é o GSA que inicialmente recupera a solução inicial s , gerada pelo Algoritmo 1. Em seguida, enquanto a temperatura inicial (T_i) não atingiu a temperatura final (T_f), a temperatura corrente (T_A) vai decrescendo de um fator ($0 < \alpha < 1$). Ao mesmo tempo, uma solução vizinha s' é gerada a cada iteração *iter* e sua energia ou custo $f(s')$ é calculado, e se $\Delta = f(s') - f(s) < 0$, a solução

vizinha s' gerada é melhor e, portanto, a solução corrente s é atualizada e comparada com a melhor solução encontrada (s^*) até o momento pelo algoritmo. Caso contrário, é calculada a probabilidade de aceitação (p_{q_A}), conforme descrita na Seção 3.3.1, e comparada com um número aleatório gerado entre 0 e 1, caso seja aceita, a corrente s é atualizada e comparada com a melhor solução encontrada s^* até o momento pelo algoritmo.

Além disso, foi implementada uma estratégia de reaquecimento, isto é, após um número específico de iterações sem melhorias (r_i), desde que o número máximo de eventos de reaquecimento (max_r) não tenha sido excedido, a temperatura corrente (T_A) é aumentada por um fator ($r_f > 1$). Esta abordagem visa promover uma maior diversificação, facilitando o escape de ótimos locais. As ideias discutidas são detalhadas de forma sequencial no Algoritmo 2.

A etapa AI compreende a parte do processo de CS que realiza a alocação da solução ao *cluster* mais próximo e a atualização do centro desse *cluster*. Na etapa AA, acontece a análise do *cluster* quando o mesmo encontra-se saturado, o que indica que tal *cluster* precisa ser melhor investigado para acelerar o processo de convergência. Por fim, a etapa de BL, conforme descrita na Seção 3.4, é acionada quando o C_i atingi o volume máximo de soluções e, desse modo, é chamado um procedimento de busca local que terá a responsabilidade de intensificar a busca por melhores soluções em um espaço promissor.

4. Experimentos Computacionais

Para realizar os experimentos computacionais dos algoritmos implementados, utilizamos um computador com sistema operacional Linux em um processador AMD Ryzen 5 4600G com Radeon Graphics executando em uma frequência de 4.2GHz com 16GB RAM. A linguagem de programação utilizada para implementação foi Python, com o interpretador na versão 3.10.12.

Os experimentos computacionais tiveram como objetivo avaliar a eficiência do algoritmo CS-GSA proposto nas instâncias do Problema de Localização com Cobertura Parcial. Para isso, identificamos dois artigos de Cardoso et al. [2022, 2023] que já tratavam do PSCLP. Os autores aplicaram no primeiro trabalho a metaheurística *Iterated Local Search* (ILS) e no segundo o *Simulated Annealing* (SA), que superou em qualidade de solução na maioria das instâncias o primeiro trabalho. Tendo isso em vista, nosso trabalho concentrou-se em reproduzir, conforme reportado em Cardoso et al. [2023], o melhor trabalho (SA). Aproveitando-se, dessa base, implementamos o CS-GSA (Seção 3.5), diferenciando-se apenas do SA no que diz respeito à perturbação, ao critério de aceitação e a técnica de reaquecimento empregada, de modo que comparação fosse mais justa possível.

4.1. Instâncias e Calibração de Parâmetros

As instâncias utilizadas nos experimentos são encontradas no trabalho de Cordeau et al. [2018]. Das instâncias disponibilizadas, utilizamos as mesmas consideradas nos trabalhos anteriores que considera 100 facilidades potenciais para 10.000 e 50.000 clientes. Para cada uma dessas situações, os raios de cobertura e o percentual de demanda mínima foram variados de acordo com a Tabela 2:

Demanda Mínima (D)	Raio de Cobertura (R)
$D = 50\% D$	$R \in \{5, 5; 5, 75; 6, 6, 25\}$
$D = 60\% D$	$R \in \{5, 5; 5, 75; 6, 6, 25\}$
$D = 70\% D$	$R \in \{3, 25; 3, 5; 3, 75; 4, 4, 25\}$

Tabela 2: Instâncias para 10.000 e 50.000 clientes

A partir deste ponto, executamos os experimentos dez vezes para cada instância, considerando os seguintes parâmetros obtidos empiricamente para o CS-GSA: A temperatura inicial (T_i)

Algorithm 2: *Algoritmo CS-GSA*

```

1  Procedimento CS-GSA:
2  Criar um Cluster inicial
3  {Etapa (MH):}
4   $s^* = s$ 
5   $iter = 0$ 
6   $T_A = T_i$ 
7   $eventos = 0$ 
8  while ( $T_A \geq T_f$ ) do
9    while ( $iter < max_{iter}$ ) do
10      $iter = iter + 1$ 
11      $last\_impr = last\_impr + 1$ 
12      $s' = gera\_vizinho(s, T_A, T_i, f(s), f(s^*), iter, max_{iter})$ 
13      $\Delta = f(s') - f(s)$ 
14     if ( $\Delta < 0$ ) then
15        $s = s'$ 
16       if ( $f(s') < f(s^*)$ ) then
17          $s^* = s$ 
18          $last\_impr = 0$ 
19       end
20     end
21     else
22       Tome  $x \in rand(0, 1)$ 
23       if ( $x < P_{q_A}$ ) then
24          $s = s'$ 
25       end
26     end
27     {Etapa (AI):}
28     Calcular a distância da solução entre os clusters do CS;
29     Inserir a solução no cluster  $C_i$ ;
30     Atualizar o centro do cluster  $C_i$ ;
31     {Etapa (AA):}
32     Verificar se o volume máximo do cluster foi alcançado. Se sim, o cluster  $C_i$  é promissor e a
        etapa BL é ativada;
33     {Etapa (BL):}
34     Aplicar a busca local no centro cluster  $C_i$  no qual o volume máximo foi alcançado para obter a
        solução  $s_{bl}$ . Se a solução da busca local, obtiver uma solução melhor que a  $s^*$  atual,  $s^* = s_{bl}$ 
35   end
36    $iter = 0$ ;
37    $T_A = T_A * \alpha$ ;
38   if ( $last\_impr \geq r_i$ ) and ( $eventos \leq max_r$ ) then
39      $T_A \leftarrow T_A * r_f$ 
40      $eventos = eventos + 1$ 
41   end
42 end
43 return  $s^*$ 
44 end-Procedure CS-GSA

```

é 100, a temperatura final (T_f) é 2, e a taxa de resfriamento (α) é 0,70. O número máximo de iterações (max_{iter}) foi definido como 200, e o parâmetro q é 1,5. A estratégia de reaquecimento inclui um fator de aumento (r_f) de 1,5, sendo realizada após 100 iterações sem melhorias (r_i), e o número máximo de eventos de reaquecimento (max_r) é 5. O volume máximo do *cluster* (v_{max}) é 30, e a distância de *threshold* (d_{th}) dada pela fórmula $d_{th} = 0,005 \times 100$, onde 100 é o número máximo de facilidades potenciais para o PSCLP neste trabalho.

4.2. Resultados Computacionais

Na Tabela 3, foram apresentados os resultados das dez execuções dos experimentos para as 28 instâncias da literatura do PSCLP. A primeira coluna da tabela (N Clientes) mostra o número de clientes. A segunda coluna (Demanda (%)) o percentual da demanda a ser atendido. A terceira coluna (Raio) mostra o raio de cobertura. A quarta coluna (fo(CPLEX)) apresenta o valor da função objetivo encontrado pelo solver CPLEX; a quinta (fo(SA)), sexta (Tempo(s)) e sétima coluna (gap(%)) apresentam, respectivamente, o melhor valor encontrado pelo SA, o melhor tempo computacional gasto para executar as dez instâncias e o *gap* obtido para cada instância, calculado como $((Sol_{heuristica}/Sol_{CPLEX}) - 1) \times 100\%$, onde $Sol_{heuristica}$ representa o custo da solução da heurística empregada, e Sol_{CPLEX} denota o custo associado à melhor solução encontrada pelo solver CPLEX e executada até à otimalidade. A oitava (fo(CS-GSA)), nona (Tempo(s)) e décima (gap(%)) coluna apresentam, respectivamente, o melhor valor encontrado pelo CS-GSA, o melhor tempo computacional gasto para executar as dez instâncias e o *gap* obtido para cada instância.

N Clientes	Demanda (%)	Raio	fo(CPLEX)	fo(SA)	Tempo(s)	gap(%)	fo(CS-GSA)	Tempo(s)	gap(%)
10.000	50	5,50	74	74	22,30	0,00	74	109,72	0,00
		5,75	63	63	15,55	0,00	63	105,02	0,00
		6,00	60	60	15,38	0,00	60	109,06	0,00
		6,25	48	48	15,41	0,00	48	90,67	0,00
10.000	60	4,00	207	210	17,55	1,45	207	142,50	0,00
		4,25	176	178	25,93	1,14	176	124,89	0,00
		4,50	154	155	16,79	0,65	154	106,09	0,00
		4,75	133	133	19,18	0,00	133	119,88	0,00
10.000	70	5,00	120	120	18,26	0,00	120	107,07	0,00
		3,25	517	518	17,20	0,19	518	133,97	0,19
		3,50	408	416	16,43	1,96	408	171,06	0,00
		3,75	324	324	16,61	0,00	324	136,02	0,00
50.000	50	4,00	275	283	16,99	2,91	275	122,70	0,00
		4,25	238	238	17,02	0,00	238	105,81	0,00
		5,50	82	82	145,82	0,00	82	201,43	0,00
		5,75	76	77	146,45	1,32	76	187,29	0,00
50.000	60	6,00	68	68	144,42	0,00	68	194,12	0,00
		6,25	66	66	144,92	0,00	66	200,61	0,00
		4,00	248	252	146,69	1,61	248	469,51	0,00
		4,25	217	232	143,71	6,91	217	316,80	0,00
50.000	70	4,50	185	189	167,81	2,16	185	396,22	0,00
		4,75	161	171	146,23	6,21	161	311,08	0,00
		5,00	137	137	145,90	0,00	137	253,04	0,00
		3,25	611	629	161,41	2,95	619	793,25	1,31
50.000	70	3,50	478	500	163,01	4,60	491	652,46	2,72
		3,75	373	373	143,43	0,00	373	451,69	0,00
		4,00	325	325	141,39	0,00	325	635,52	0,00
		4,25	284	292	151,12	2,82	284	356,88	0,00

Tabela 3: Comparação de *gap* e tempo entre o SA e o CS-GSA

Analisando os resultados obtidos, é possível notar que, dentre as 28 instâncias analisadas, 25 delas obtiveram soluções ótimas e somente 3 obtiveram soluções com *gap* maior que zero quando o CS-GSA foi empregado. O que representa que, aproximadamente, 89,3% das instâncias alcançaram o valor ótimo e somente 10,7% não o alcançaram. Vale ressaltar que as poucas instâncias que não obtiveram a solução ótima apresentaram *gap* inferior a 3%.

Por outro lado, quando o SA foi empregado para resolver as mesmas instâncias, somente 14 delas obtiveram soluções ótimas. O que representa 50% das instâncias. A outra metade obteve soluções com *gap* inferior a 7%.

Sob outro ponto de vista, quando o CS-GSA foi utilizado considerando tanto 10.000 clientes quanto 50.000 clientes, para todas as instâncias com demanda mínima a ser atendida de 50% e de 60%, a solução ótima foi obtida. Quando a demanda mínima a ser atendida considerada foi de 70%, foram obtidas somente as 3 soluções com *gap* maior que zero. Dos quais todos inferiores a 3%. Ainda sob esse ponto de vista, cabe observar que, quando o SA foi utilizado, somente no grupo de instâncias de 10.000 cliente e demanda mínima a ser atendida de 50% foi que todas as soluções obtidas foram ótimas. Nos demais grupos de instâncias, a maioria das soluções obteve *gap* maior que zero. Dos quais todos inferiores a 7%.

No quesito tempo computacional, o SA obteve médias menores para todas as instâncias. Porém, de maneira geral, é possível dizer que o CS-GSA superou o SA para resolver o PSCLP, dado que, apesar de exigir maior esforço computacional, o CS-GSA conseguiu obter resultados com maior assertividade.

5. Conclusão

Dada a importância de estudos na área de problemas de localização de facilidades, e a flexibilidade e bom desempenho do método híbrido *Clustering Search* (CS) para resolver diversos problemas de otimização combinatória observados na literatura, este trabalho propôs um método com base na metaheurística CS para resolver as instâncias do *Partial Set Covering Location Problem* (PSCLP). O método aqui proposto foi o CS-GSA que utiliza a metaheurística *Generalized Simulated Annealing* (GSA) como gerador de soluções viáveis para serem agrupadas e analisadas no processo de CS a fim de identificar regiões promissoras nos agrupamentos realizados pelo algoritmo e intensificar a busca nas regiões identificadas como promissoras.

Os resultados computacionais indicaram que o CS-GSA obteve a melhor solução para as instâncias comparadas da literatura, levando um tempo máximo de 14 minutos, no pior caso. Isso mostra maior assertividade que SA, obtendo soluções ótimas em 89,3% dos casos. Enquanto que o SA obteve soluções ótimas em apenas 50% dos casos.

Como proposta de trabalhos futuros, pretende-se testar outras estratégias de obtenção de vizinhança para as mesmas instâncias utilizadas neste trabalho. Além disso, testar outros conjuntos de instâncias com um número maior de clientes. É muito importante a realização de pesquisas nesse âmbito, que visem a otimização do planejamento logístico pois o desenvolvimento de estratégias logísticas que priorizem a redução de impactos ambientais e o bem-estar da população não apenas beneficia a empresa em termos de eficiência e competitividade, mas também contribui para a construção de uma sociedade mais sustentável para as gerações futuras.

Agradecimentos

Os autores agradecem à UFRJ (ALV 23079.227622/2023-70), à FAPERJ (E-26/201.241/2022), ao CNPq (307663/2021-3, 308792/2022-0, 88887.837779/2023-00) e à CAPES (88887.838048/2023-00) pelo financiamento parcial desta pesquisa. Agradecemos também aos revisores por seus valiosos comentários.

Referências

- Arakaki, R. G. I. e Lorena, L. A. N. (2006). Uma heurística de localização-alocação (hla) para problemas de localização de facilidades. *Produção*, 16:319–328.
- Berge, C. (1957). Two theorems in graph theory. *Proceedings of the National Academy of Sciences of the United States of America*, 43:842–844.
- Cardoso, L. C., De Sá, E. M., e De Souza, S. R. (2023). Meta-heurística simulated annealing aplicada ao problema de localização com cobertura parcial. *Anais do LV Simpósio Brasileiro de Pesquisa Operacional*.

- Cardoso, L. C., Mourão, F. P., De Sá, E. M., e De Souza, S. R. (2022). Metaheurística iterated local search aplicada ao problema de localização com cobertura parcial. *Anais do XIX Encontro Nacional de Inteligência Artificial e Computacional*.
- Chaves, A. A. e Lorena, L. A. N. (2007). Aplicação do algoritmo clustering search aos traveling salesman problems with profits. *Anais do XXXIX Simpósio Brasileiro de Pesquisa Operacional*.
- Chaves, A. A. e Lorena, L. A. N. (2010). Clustering search algorithm for the capacitated centered clustering problem. *Computers Operations Research*, (37):552–558.
- Chen, L., Chen, S.-J., Chen, W.-K., Dai, Y.-H., Quan, T., e Chen, J. (2023). Efficient presolving methods for solving maximal covering and partial set covering location problems. *European Journal of Operational Research*, 311:73–87.
- Church, R. e ReVelle, C. (1974). The maximal covering location problem. *Papers in Regional Science*, 32:101–118.
- Cordeau, J.-F., Furini, F., e Ljubic, I. (2018). Benders decomposition for very large scale partial set covering and maximal covering location problems. *European Journal of Operational Research*.
- Daskin, M. S. e Owen, S. H. (1999). Two new location covering problems: The partial p-center problem and the partial set covering problem. *Geographical Analysis*, 31(3).
- de Abreu, V. H. S., González, P. H., Mauri, G. R., Ribeiro, G. M., Orrico, R. D., Júnior, N. F. R. C., e Abramides, C. A. (2020). Network sensor location problem with monitored lanes: Branch-and-cut and clustering search solution techniques. *Computers & Industrial Engineering*, 150:106827.
- De Oliveira, A. C. M., Chaves, A. A., e Lorena, L. A. N. (2013). Clustering search. *Brazilian Operations Research Society*, (33):105–121.
- García, S. e Marín, A. (2015). *Location Science: Covering Location Problems*.
- González, P. H., Clímaco, G., Mauri, G. R., Vieira, B. S., Ribeiro, G. M., Orrico Filho, R. D., Simonetti, L., Perim, L. R., e Hoffmann, I. C. S. (2019). New approaches for the traffic counting location problem. *Expert systems with applications*, 132:189–198.
- Hakimi, S. (1964). Optimum location of switching centers and the absolute medians of a graph. *Operations Research*, (12):450–459.
- Kirkpatrick, S., Gelatt, C. D., e Vecchi, M. (1983). Optimization by simulated annealing. *Science*, 220(4598).
- Liu, X., Zhou, Y., Yin, M., e Lv, S. (2022). An argentine ant system algorithm for partial set covering problem. *Data Technologies and Applications*, 56(5):762–781.
- López-Ibáñez, M., Dubois-Lacoste, J., Pérez Cáceres, L., Stützle, T., e Birattari, M. (2016). The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, 3:43–58.
- Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., e Teller, A. H. (1953). Equation of state calculations by fast computing machines”. *The Journal of Chemical Physics*, 21(6).
- Oliveira, A. e Lorena, L. A. N. (2007). Hybrid evolutionary algorithms and clustering search. *Studies in Computational Intelligence*, (75).
- Rabello, R. L., Mauri, G. R., Ribeiro, G. M., e Lorena, L. A. N. (2013). Meta-heurística clustering search aplicada ao problema de rotulação cartográfica de pontos. *Anais do XLV Simpósio Brasileiro de Pesquisa Operacional*.
- Souto, G., Morais, I., Mauri, G. R., Ribeiro, G. M., e González, P. H. (2021). A hybrid matheuristic for the two-stage capacitated facility location problem. *Expert Systems With Applications*, (185).
- Szu, H. e Hartley, R. (1987). Fast simulated annealing. *Physics letters A*, 122(3-4):157–162.
- Toregas, C., Swain, R., ReVelle, C., e Bergman, L. (1971). The location of emergency service facilities. *Guidelines for the Practice of Operations Research*, 19:1259–1551.
- Tsallis, C. (1988). Possible generalization of boltzmann-gibbs statistics. *Journal of statistical physics*, 52:479–487.
- Tsallis, C. e Stariolo, D. A. (1996). Generalized simulated annealing. *Physica A: Statistical Mechanics and its Applications*, 233(1-2):395–406.