Sistemas especialistas com aplicativos 82 (2017) 67-76

Listas de conteúdos disponíveis em ScienceDirect

Sistemas especialistas com aplicativos

Página inicial do jornal: www.elsevier.com/locate/eswa

GRASP e heurísticas híbridas GRASP-Tabu para resolver um problema de localização de cobertura máxima com pedido de preferência do cliente

Juan A. Díaz [uma], Dolores E. Luna [uma, b, *], José-Fernando Camacho-Vallejo [c], Martha-Selene Casas-Ramírez [c]

[uma] *Depto. de Actuaría, Física y Matemáticas, Universidad de las Américas Puebla, México*

[b] *Departamento de Ingeniería Industrial y Mecánica, Universidad de las Américas Puebla, Ex Hacienda Santa Catarina Mártir. CP. 72810, San Andrés Cholula, Puebla, México*

[c] *Facultad de Ciencias Físico Matemáticas, Universidad Autónoma de Nuevo León, México*

informações do artigo

Historia do artigo:
Recebido em 22 de junho de 2016
Revisado em 31 de março de 2017
Aceito em 1 de abril de 2017
Disponível online 2 de abril de 2017

Palavras-chave:
Cobertura máxima
Metaheurísticas
APERTO
Pesquisa tabu
Programação de dois níveis

abstrato

Neste estudo, um problema de localização de cobertura máxima é investigado. Neste problema, queremos maximizar a demanda de um conjunto de clientes atendidos por um conjunto de *p* instalações localizadas entre um conjunto de locais potenciais. Presume-se que exista um conjunto de instalações pertencentes a outras empresas e que os clientes escolham livremente a alocação às instalações dentro de um raio de cobertura. O problema pode ser formulado como um problema de programação matemática de dois níveis, no qual o líder localiza instalações a fim de maximizar a demanda coberta e o seguidor aloca clientes para a instalação preferida entre as selecionadas pelo líder e instalações de outras empresas. Nós propomos uma heurística de procedimento de busca adaptativa aleatória gananciosa (GRASP) e uma heurística híbrida GRASP-Tabu para encontrar soluções quase ótimas. Os resultados das abordagens heurísticas são comparados às soluções obtidas com uma reformulação do problema em um único nível. Experimentos computacionais demonstram que os algoritmos propostos podem encontrar soluções de muito boa qualidade com uma pequena carga computacional. A característica mais importante das heurísticas propostas é que, apesar de sua simplicidade, soluções ótimas ou quase ótimas podem ser determinadas de maneira muito eficiente.

1. Introdução

Neste estudo, consideramos uma variante do problema de localização de cobertura máxima. Estudamos uma situação em que uma empresa deseja localizar uma série de *p* instalações para maximizar a demanda capturada em um mercado onde várias empresas já operam. Identificamos dois níveis de decisão neste problema: a localização das instalações e a alocação de clientes. Normalmente, na maioria dos problemas de localização, ambas as decisões são assumidas por um único tomador de decisão. No entanto, existem situações em que os clientes são livres de escolher um prestador de serviços, especialmente nos casos em que os clientes se deslocam às instalações dos prestadores de serviços. Na maioria dos problemas de alocação de localização na literatura, os clientes são designados para a instalação mais próxima. No entanto, o comportamento do cliente é influenciado por muitos fatores, como idade, renda, educação, etc .; portanto, os clientes fazem nem sempre vá para a instalação mais próxima. Consideramos que existe uma distância limite que representa a distância máxima que um cliente está disposto a percorrer para obter um serviço. Portanto, um cliente irá patrocinar sua instalação preferida entre aqueles dentro da distância limite. Modelamos a ordem de preferência do cliente como em Belotti, Labbé, Ma ffi oli e Ndiaye (2007) ; Cánovas, García, Labbé e Marín (2007) ; Hanjoul e Thill (1987) ; Hansen, Kochetov e Mladenovi (2004) . Para cada cliente, é usada uma lista indicando a ordem de preferência para cada instalação potencial dentro da distância limite. Ognjanovic´, Gaševi e Bagheri (2013) sugerem que uma lista ordenada de preferências de instalações do cliente pode ser considerada para muitas aplicações do mundo real em diversos campos / áreas. Eles propõem preferências de ordenação de forma lexicográfica nas variáveis de alocação, o que é muito semelhante à lista ordenada de preferências usada em nossa pesquisa. Também consideramos que o modelo competitivo é estático, conforme definido em Plastria (2001) , onde se presume que a concorrência não reagirá no curto prazo devido a restrições econômicas ou ao tempo necessário para reagir.

O problema de localização de cobertura máxima (introduzido em Church e ReVelle, 1974) localiza um número fixo de *p* instalações de um conjunto de locais potenciais para maximizar a demanda do cliente coberta dentro de um raio de cobertura. Neste artigo, estudamos uma extensão do max-

* Autor correspondente em: Departamento de Ingeniería Industrial y Mecánica, Universidad de las Américas Puebla, Ex Hacienda Santa Catarina Mártir. CP. 72810, San Andrés Cholula, Puebla, México.

Endereço de e-mail: juana.diaz@udlap.mx (JA Díaz), dolorese.luna@udlap.mx (DE Luna), jose.camachovl@uanl.edu.mx (J.-F. Camacho-Vallejo), martha.casasrm@uanl.edu.mx (M.-S. Casas-Ramírez).

http://dx.doi.org/10.1016/j.eswa.2017.04.002
0957-4174 / © 2017 Elsevier Ltd. Todos os direitos reservados.

Um problema de localização de cobertura, em que se presume que existe um conjunto de instalações pertencentes a outras empresas e que os clientes podem escolher a instalação que prestará o serviço dentro de um raio de cobertura. Esse problema pode ser formulado como um problema de programação matemática de dois níveis, em que o líder determina as localizações das instalações a partir de um conjunto de locais em potencial e o seguidor determina a alocação de clientes para seus locais preferidos. De acordo com uma extensa revisão da literatura, apenas Lee e Lee (2012) considere as preferências do cliente em relação aos problemas de cobertura. Nesse artigo, o objetivo é maximizar a demanda ponderada coberta com relação a um índice de preferência do cliente. Os autores também consideram as restrições com relação aos limites superior e inferior do número de clientes alocados para cada instalação.

Muitos estudos de problemas de localização consideraram a preferência do cliente. Por exemplo, Hansen et al. (2004) considerou as preferências do usuário em relação ao problema de localização de instalação não capacitada Nesse problema, a função objetivo do líder é minimizar os custos de localização e distribuição e a função objetivo do seguidor otimiza as preferências do cliente. Nesse estudo, os autores propuseram uma nova reformulação e mostraram que ela fornece melhores relaxamentos de programação linear (LP) em comparação com três reformulações da literatura. Com base em experimentos usando instâncias de dados aleatórios,
Hansen et al. (2004) concluiu que a reformulação proposta pode encontrar uma solução ótima mais rápido do que as abordagens comparadas. Outra reformulação de nível único do mesmo problema foi proposta por Vasil'ev, Klimentova e Kochetov (2009) . Nessa reformulação, as desigualdades de clique foram usadas para obter limites inferiores. Mais tarde, Vasilyev e Klimentova (2010) empregou relaxamento linear do problema de dois níveis para obter limites inferiores e aplicou uma heurística de recozimento simulado para obter soluções viáveis. Um algoritmo de ramificação e corte que combina os dois esquemas de limitação foi usado para obter soluções quase ótimas. Métodos metaheurísticos também têm sido usados para obter soluções para este problema. Um algoritmo evolutivo de Stackelberg foi proposto em Camacho-Vallejo, CorderoFranco e González-Ramírez (2014) . Nesse estudo, soluções viáveis de boa qualidade para até 500 instalações e 1000 clientes foram obtidas de forma eficiente. Além disso, Maric´, Stanimirovic´ e Milenkovic´ (2012) propuseram diferentes metaheurísticas para obter soluções viáveis para o problema, ou seja, um método de otimização por enxame de partículas, um método de recozimento simulado e um método de busca de vizinhança adaptado. Nesses métodos, o problema de nível inferior pode ser obtido de maneira ótima reorganizando a matriz de preferência do cliente.

Problemas de localização de instalações competitivas de dois níveis também foram considerados na literatura. Kress e Pesch (2012) levantados problemas de localização competitiva sequencial. Eles dividiram os problemas em duas categorias (veja Hakimi, 1983): ($r \mid X_p$) - medianoide e ($r \mid p$) - problemas de centróide. Os problemas foram formulados usando um líder - abordagem do seguidor. No ($r \mid X_p$) - problema medianoide, visto que o líder tem p instalações, o seguidor localiza de maneira ideal r instalações de um conjunto de sites em potencial X_p. No ($r \mid p$) - problema de centróide, o líder localiza de maneira ideal p instalações sabendo que o seguidor irá reaja localizando r instalações.

Neste estudo, propomos uma heurística de procedimento de busca adaptativa randomizada gananciosa (GRASP) e uma heurística híbrida GRASP-Tabu. A heurística híbrida GRASP-Tabu combina GRASP e pesquisa tabu para encontrar limites inferiores de maneira eficiente para instâncias de grande escala para o problema de localização de cobertura máxima com a ordem de preferência do cliente. Para avaliar a qualidade dos limites obtidos, reformulamos o problema como um problema de programação inteira de nível único usando desigualdades válidas, como em Cánovas et al. (2007) , para garantir que cada cliente seja alocado à instalação preferencial entre as que estão abertas. De acordo com os resultados computacionais, apesar de sua simplicidade, os algoritmos propostos fornecem soluções ótimas ou quase ótimas em um pequeno tempo de computação.

O restante deste artigo está organizado da seguinte forma. Dentro Seção 2 , uma formulação de dois níveis do problema é apresentada e uma reformulação de um nível equivalente é proposta. Dentro Seção 3 , as heurísticas GRASP e híbrida GRASP-Tabu são descritas. Experimentos computacionais são relatados em Seção 4 , e as conclusões são apresentadas em Seção 5 .

2. Declaração do problema

Nesta seção, descrevemos o problema de localização de cobertura máxima de dois níveis (*BLMCLP),* uma formulação matemática de dois níveis do problema e uma reformulação de nível único que substitui o problema de decisão do seguidor por um conjunto de desigualdades válidas. Essa substituição garante que as demandas do cliente sejam alocadas às instalações localizadas nos locais mais preferidos.

O *BLMCLP* considera a seguinte situação: uma empresa pretende entrar no mercado abrindo p instalações, que devem estar localizadas dentro de um conjunto pré-definido de locais potenciais, para atender à demanda de um conjunto de clientes que são livres para patrocinar qualquer instalação entre as que já existem ou aquelas a serem inauguradas. Supõe-se que a empresa conhece as preferências do cliente. Então, podemos considerar que o líder é a empresa concorrente que está entrando. Nesta situação, o líder determina a localização exata de p instalações para maximizar a demanda capturada e o seguidor é um conjunto de clientes que escolherá qual instalação fornecerá o serviço de acordo com suas preferências e o raio de cobertura. Além disso, consideramos que as instalações existentes no mercado não podem ser encerradas.

A situação descrita acima pode ser modelada com programação de dois níveis, uma vez que a empresa (líder) determina os locais das instalações e os clientes (seguidor) patrocinam suas instalações preferidas entre as instalações abertas. Assim, devido à hierarquia de tomada de decisão e à falta de cooperação entre os tomadores de decisão, o problema pode ser formulado como um problema de programação de dois níveis.

2.1. Formulação matemática de dois níveis

Deixar eu_1 ser o conjunto de locais potenciais para localizar as instalações e eu_2 ser o conjunto de locais de instalações existentes, onde $Eu = eu_1 \cup eu_2$ Deixar J_2 seja o conjunto de clientes cobertos por uma instalação existente e J_1 ser o conjunto de clientes não cobertos pelas instalações existentes (ou seja, não há um ex-instalação dentro do seu raio de cobertura), onde $J = J_1 \cup J_2$ Para cada $j \in J$, nós definimos $Eu\ j)$ como o conjunto de índice de instalações que cobrem a demando do cliente j (ou seja, o conjunto de locais potenciais não excedendo uma distância máxima do local do cliente) e, para cada $eu \in EU$, nós definimos $J\ (i)$ como o conjunto de índices de clientes cuja demanda pode ser coberto por uma instalação localizada no local eu. Deixar D_j ser a demanda associada ao cliente $j \in J$, p ser o número de instalações que o líder deseja localizar e $g_{eu\,j}$ seja a preferência do cliente $j \in J$ em direção à instalação localizada em $eu \in EU$.

Considere as seguintes variáveis de decisão:

$$y_i = \begin{cases} 1, & \text{se uma instalação está localizada no local } eu \\ 0, & \text{por outro lado} \end{cases}, \quad eu \in eu$$

e

$$x_{ij} = \begin{cases} 1, & \text{se cliente } j \text{ é alocado para instalação localizada no site } eu \\ 0, & \text{por outro lado} \end{cases}, \quad j \in J,\ eu \in Eu\ j)$$

O *BLMCLP* pode ser formulado como um programa de dois níveis como segue.

$$(BLMCLP) \max_{y,\,x} \sum_{eu \in eu_1} \sum_{j \in J(i)} D_j x_{eu\,j} \tag{1}$$

sujeito a: $y_i = 1\ \forall\ eu \in eu_2$ (2)

$$\sum_{eu \in eu_1} y_i = p \quad (3)$$

$$y_{eu} \in \{0, 1\} \; \forall \; eu \in eu \quad (4)$$

Onde x resolve

$$\max_{x} \sum_{eu \in eu} \sum_{j \in J(i)} g_{eu\,j}\, x_{eu\,j} \quad (5)$$

sujeito a:

$$\sum_{eu \in Eu\,j)} x_{ij} = 1 \; \forall \; j \in J_2 \quad (6)$$

$$x_{eu\,j} \leq y_{eu} \forall \; eu \in Eu\,j \in J\,(i) \quad (7)$$

$$\sum_{eu \in Eu\,j)} x_{eu\,j} \leq 1 \; \forall \; j \in J_1 \quad (8)$$

$$x_{eu\,j} \in \{0, 1\} \; \forall \; eu \in Eu\,j \in J\,(i) \quad (9)$$

Nesta formulação, a função objetivo do líder (1) maximiza a demanda coberta pelo novo p instalações. Restrições (2) e (3) garantir que as instalações existentes permaneçam abertas e que exatamente p novas instalações serão localizadas, respectivamente. Os valores de as variáveis de decisão $x_{eu\,j,}\; eu \in Eu\,j \in J\,(i),$ representam a solução ótima do problema do seguidor definido por (5) - (9) . Do seguidor função objetiva (5) maximiza a soma das preferências do cliente. Restrições (6) garantir que a demanda do cliente que é coberta por uma instalação existente permanecerá coberta pela mesma instalação ou por uma nova instalação. Restrições (7) garantir que a demanda do cliente só possa ser atendida por uma instalação aberta localizada dentro de seu conjunto de cobertura. Finalmente, as restrições (8) garantir que as demandas de clientes não cobertas possam permanecer descobertas se nenhuma instalação estiver localizada dentro de seu raio de cobertura.

Um problema de dois níveis bem definido depende da existência e exclusividade de uma solução ótima para o problema de nível inferior (Vasil'ev et al., 2009) Em outras palavras, se existem soluções ótimas alternativas para o problema do seguidor para a decisão de um determinado líder, o problema de dois níveis está mal colocado. No problema considerado aqui, o problema do seguidor terá uma solução ótima única se para cada cliente $j \in J,$ suas preferências são números consecutivos positivos de 1 a $|\, I\,(j)\,|$.

2.2. Reformulação de nível único

Como mencionado em Bard (1998) , Colson, Marcotte e Savard (2007) , e Kalashnikov, Dempe, Pérez-Valdéz, Kalashnikova e Camacho-Vallejo (2004) , resolver um problema de dois níveis pode não ser uma tarefa fácil; mesmo um problema linear de dois níveis é NPhard (ver Jeroslow, 1985 e Hansen, Jaumard e Savard, 1992) Além disso, uma vez que não existe uma metodologia geral e, portanto, nenhum software de programação matemática para resolver qualquer problema de dois níveis, métodos específicos para obter soluções ótimas ou quase ótimas precisam ser explorados. O *BLMCLP* descrito anteriormente pode ser reduzido a um problema de programação inteira mista de nível único. Nas abordagens clássicas, as reformulações são realizadas usando as relações primal-duais para o problema do seguidor. Para garantir a otimização do problema do seguidor, podemos forçar a igualdade das funções objetivo dos problemas primais e duais ou incluir restrições de folga complementares. Neste estudo, consideramos essas duas reformulações e uma reformulação direta de nível único que garante a otimização do problema do seguidor, o que força a demanda do cliente a ser alocada ao máximo instalação preferida. Após a realização de testes computacionais preliminares com as três reformulações, observou-se que os limites de relaxação linear são muito melhores para a reformulação proposta. Descrevemos o problema de localização de cobertura máxima de nível único (*SLMCLP)* reformulação no seguinte.

Uma vez que o problema do seguidor garante que a demanda de clientes que estão dentro do raio de cobertura de uma ou mais instalações seja alocada à instalação mais preferida, o programa matemático de dois níveis pode ser formulado como um programa de nível único. Como em Cánovas et al. (2007) , consideramos uma reformulação de nível único do problema, substituindo o problema de decisão do seguidor por um conjunto de desigualdades válidas que garantem que sempre que um cliente estiver dentro do raio de cobertura de mais de uma instalação, a demanda do cliente será sempre alocada para a mais preferida instalação.

Para cada $j \in J,$ deixar $eu_{1,}\, eu_{2,}\ldots,\, eu_{|\,I(j)\,|}$ sejam os elementos em $Eu\,j)$ de tal modo que $g_{eu\,1,\,j} > g_{eu\,2,\,j} > \ldots > g_{i\,|\,I(j)\,|,\,j.}$ Então, para garantir que a demanda de cada cliente seja alocada para a instalação preferida, incluímos o seguindo desigualdades válidas.

$$\sum_{s\,=\,k\,+\,1}^{|\,I(j)\,|} x_{eu\,s,\,j} + y_{eu\,k} \leq 1 \qquad \forall \; j \in J,\; k \in 1,\ldots,\,|\,I\,(j)\,| - 1 \quad (10)$$

Essas desigualdades garantem que, se não houver instalações abertas no k o local preferido para o cliente $j,$ então o cliente deve ser alocado a uma instalação cuja localização seja menos preferida do que a k o local ou sua demanda não é alocada a nenhuma instalação aberta. Então, o problema pode ser modelado com o seguinte modelo de programação matemática de nível único.

$$(\,SLMCLP)\; \max_{y,\,x} \sum_{eu \in eu_1} \sum_{j \in J\,(i)} D_j\, x_{eu\,j} \quad (1)$$

sujeito a:

$$y_i = 1 \; \forall \; eu \in eu_2 \quad (2)$$

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

$$y_{eu} \in \{0, 1\} \; \forall \; eu \in eu \quad (4)$$

$$\sum_{eu \in Eu\,j)} x_{ij} = 1 \; \forall \; j \in J_2 \quad (6)$$

$$x_{eu\,j} \leq y_{eu} \forall \; eu \in Eu\,j \in J\,(i) \quad (7)$$

$$\sum_{eu \in Eu\,j)} x_{eu\,j} \leq 1 \; \forall \; j \in J_1 \quad (8)$$

$$x_{eu\,j} \in \{0, 1\} \; \forall \; eu \in Eu\,j \in J\,(i) \quad (9)$$

$$\sum_{s\,=\,k\,+\,1}^{|\,I(j)\,|} x_{eu\,s,\,j} + y_{eu\,k} \leq 1 \; \forall \; j \in J,\; k \in 1,\ldots,\,|\,I\,(j)\,| - 1 \quad (10)$$

Na formulação acima, observamos que as restrições $x_{eu\,j} \in \{0, 1\},\; eu \in Eu\,j \in J\,(i)$ pode ser substituído por $x_{eu\,j} \geq 0,\; eu \in Eu\,j \in J\,(i).$ Dada qualquer solução viável, onde $eu_* = \{\, eu \in I\!:\, y_i = 1\}$ denota o conjunto de instalações abertas, para um determinado cliente $j \in J,$ deixar $eu_{1,}\, eu_{2,}\ldots,\, eu_{|\,I(j)\,|}$ tal naquela $g_{eu\,1,\,j} > g_{eu\,2,\,j} > \ldots > g_{i\,|\,I(j)\,|,\,j.}$ Então se $x_{ik\,j} > 0$ por restrições (7), isso implica que $y_i = 1.$ Além disso, pelas restrições (10) e (7), $y_{i_r} = 0$ e $x_{eu\,j} = 0$ para $r = k + 1,\ldots,\, |\,I\,(j)\,| - 1.$ Caso contrário, $x_{ik\,j}$ não pode ser maior que zero porque se $y_i = 1,$ para alguns $r = 1,\ldots,\, k - 1,\; x_{ik\,j}$ deve ser zero para satisfazer as restrições (10) associadas à localização da instalação eu_r e cliente $j.$ Finalmente, $x_{ir\,j} = 0$ para todos $r = k + 1,\ldots,\, |\,I\,(j)\,|$ dado que $y_i = 1.$ Portanto, por restrições (6) ou con-restrições (8) e assumindo que $D_j > 0,\; x_{ik\,j} = 1.$ Aqui, provamos a equivalência entre *BLMCLP* e *SLMCLP.*

Proposição 1. *Se (x, y) é uma solução viável para BLMCLP, então as desigualdades válidas (10) estão satisfeitos e, portanto, (x, y) também é viável para SLMCLP.*

Prova. Deixar $\bar{x}$ ser a solução ótima para o problema de nível inferior para uma determinada solução viável da solução do líder $\bar{y}.$

Portanto, em qualquer solução viável do problema de dois níveis, para cada cliente $j \in J$ e um dado $k \in [\,1,\, |\,I\,(j)\,|\,J,$ temos os seguintes casos.

1 $\bar{y}_k = 0$. Observamos que em qualquer solução viável de *BLMCLP*, $\forall j \in i_k$, $\sum_{eu \in Eu\,j)} x_{eu\,j}$ é no máximo 1, pois se $j \in J_2$, então $\sum_{eu \in Eu\,j)} \bar{x}_{ij} = 1$ e E se $j \in J_1$, então $\sum_{eu \in Eu\,j)} \bar{x}_{eu\,j} \le 1$. Portanto, $\sum_{s=k+1}^{|I(j)|} \bar{x}_{é,\,j} \le 1$ desde o set $\{ eu_{k+1}, \ldots, eu \mid_{I(j)|}\} \subset Eu\,j)$.

2 $\bar{y}_i = 1$. Portanto, $x_{eu_j} = 0$ para r tudo $s > k$ desde restrições (6) - (8) garantir que a demanda de cada cliente seja alocada em mais uma instalação aberta e se $x_{é,\,j} = \bar{1}$ para alguns $s > k$, isso vai contradizer a otimização do problema de nível inferior. Além disso ção, uma vez que é um problema não capacitado e $g_{ik,\,j} > g_{é,\,j}$ para todos $s > k$, a demanda do cliente j pode ser alocado para seus k^{o} localização preferencial porque há uma instalação localizada nesse local (ou seja, $\bar{y}_k = 1$).

Portanto, em ambos os casos, as restrições (10) segure.

Proposição 2. *Seja (x, y) uma solução ótima de SLMCLP. Então, x é a solução ótima do problema de nível inferior no BLMCLP para o y dado; portanto, (x, y) é viável para BLMCLP.*

Prova. Deixar $I = \{i \in I: y = 1\}$ Para cada $j \in J$, temos os seguintes casos mutuamente exclusivos.

1. Se $j \in J_2$, então a demanda do cliente j deve ser alocado para ex- na verdade uma instalação aberta porque sua demanda já está coberta pelas instalações existentes. Portanto, deixe eu_k ser a facilidade para a qual a demanda do cliente j está alocado, ou seja, $x_{ik,\,j} = 1$ para alguns $k \in [1, \mid I(j)\mid]$. Então, $x_{é,\,j} = 0$ para todos $s \in I(j)\{k\}$ porque con tensões (6) deve esperar. Além disso, $y = 1$ porque $x_{ik,\,j} \le y_{ik}$ Restrições (10) Garanta que $y = 0$ para todos $s < k$ desde se $y = 1$ para alguns $s < k$, $x_{ik,\,j}$ deve ser igual a 0. Portanto, a demanda do cliente j é alocado à sua instalação preferida entre as instalações abertas, ou seja, $eu_k = \text{argmax}_{eu \in Eu\,j)\cap EU\{g_{eu\,j}\}}$.
2. Se $j \in J_1$, então a demanda do cliente j deve ser alocado para no máximo uma instalação aberta porque as restrições (8) deve esperar. Portanto, da mesma forma que no caso em que $j \in J_1$, E se a demanda do cliente é alocada à instalação eu_k, Isso significa que $x_{ik,\,j} = 1$ para alguns $k \in [1, \mid I(j)\mid]$ e $x_{é,\,j} = 0$ para todos $s \in I(j)\{k\}$. Dentro Adição, $y_i = \bar{1}$ desde $x_{ik,\,j} \le y_{ik}$ assim, restrições (10) garantir naquela $y_i = 0$ para todos $s < k$ e $eu_k = \text{argmax}_{eu \in Eu\,j)\cap EU\{g_{eu\,j}\}}$. Se o de- mando do cliente j não está alocado, isso significa que $y = 0$ e $x_{eu_j} = 0$, $\forall\, eu \in Eu\,j)$ porque se $y = 1$ para pelo menos um $k \in Eu\,j)$, esta irá contradizer a otimização do *SLMCLP* porque as demandas são consideradas positivas e, portanto, o valor objetivo pode ser aumentado por D_j pela configuração $x_{kj} = 1$

Proposição 3. *BLMCLP e SLMCLP fornecem as mesmas soluções ideais.*

Prova. Deixar (x, y) ser uma solução ótima para *BLMCLP*. De Proposição 1 , sabemos que esta solução também é viável para *SLMCLP*. Suponha que exista uma solução ótima (x, y) para a formulação de nível único, de modo que

$$\sum\sum_{eu \in eu_1\, j \in J(i)} D_j \bar{x}_{ij} > \sum\sum_{eu \in eu_1\, j \in J(i)} D_j x_{eu\,j}.$$

De Proposição 2 , sabemos que esta solução também é viável para a formulação de dois níveis; portanto, contradiz a otimização de (x, y) para *BLMCLP*.

Por outro lado, deixe (x, y) ser uma solução ótima para *SLMCLP*. De Proposição 2 , sabemos que esta solução também é viável para *BLMCLP*. Suponha que haja outra solução viável (x, y) para *BLMCLP* de tal modo que $\sum\sum_{eu \in eu_1\, j \in J(i)} D_j \bar{x}_{ij} > \sum\sum_{eu \in eu_1\, j \in J(i)} D_j x_{eu\,j}$.

De Proposição 1 , (x, y) também é viável para o *SLMCLP*, contradizendo assim a otimização de (x, y) para o *SLMCLP*.

A partir dos últimos resultados, pode-se concluir que ambas as formulações matemáticas são equivalentes.

Observe que esta reformulação de nível único pode ser usada para obter um valor de referência a fim de medir o desempenho dos algoritmos propostos.

3. Heurísticas GRASP e híbridas GRASP-Tabu

A inteligência artificial (IA) tem sido aplicada em muitos domínios, como robótica, reconhecimento de fala, planejamento e programação para muitas tarefas, planejamento de logística, reconhecimento de padrões e design VLSI (Russell e Norvig, 2010) Ao tentar fornecer soluções para esses problemas, a IA encontra muitos obstáculos. Um obstáculo está relacionado ao fato de que muitos dos problemas envolvem problemas combinatórios nos quais encontrar todas as soluções alternativas possíveis e investigá-las é, em termos práticos, impossível. Em tais situações, é muito útil ter estratégias de pesquisa informadas que possibilitem processos de pesquisa mais eficientes. Os métodos heurísticos e metaheurísticos ajudam a gerenciar esses problemas de decisão complexos, tirando proveito da estrutura e das características dos problemas a serem resolvidos, fornecendo assim à IA métodos de pesquisa eficientes (Glover, 1989)

A metaheurística GRASP foi proposta pela primeira vez por Feo e Resende (1995) . É um procedimento de pesquisa iterativo. Em cada iteração, uma nova solução é construída por um procedimento aleatório guloso, que é então aprimorado usando um procedimento de busca local. Dentro Feo e Resende (1995) , os autores fornecem uma justificativa intuitiva para GRASP como uma técnica de amostragem repetitiva. Em cada iteração do algoritmo guloso, um novo elemento é selecionado a partir de um *lista restrita de candidatos* e adicionado à solução. A média e a variância da distribuição da amostra são função da cardinalidade da lista restrita de candidatos usada na fase construtiva do GRASP. Como as soluções de amostra são selecionadas aleatoriamente, por estatísticas de pedido, pode-se esperar intuitivamente que o melhor valor encontrado supere o valor médio.

Pesquisa tabu (Glover, 1989; 1990) depende do uso de adaptativos memória e exploração responsiva. Ele usa memória adaptativa para registrar informações históricas do processo de pesquisa. O termo exploração responsiva se refere à capacidade do método de fazer escolhas estratégicas para atingir a eficácia.

As estruturas de memória usadas pela busca tabu operam referenciando quatro dimensões: recência, frequência, qualidade e influência. As formas mais simples de pesquisa tabu usam apenas memória baseada em recência para evitar ciclos. As estruturas de memória são utilizadas para registrar os atributos das soluções recentemente visitadas, que são rotuladas como tabuativas, e as soluções que contêm atributos tabuativos são proibidas. Portanto, o uso de memória baseada em recência é uma estratégia de exploração agressiva que tenta ir além da otimização local porque impede a revisitação de soluções visitadas no passado recente. Os critérios de aspiração também podem ser usados para substituir as restrições de tabu (remover atributos tabu-ativos). A forma mais simples de um critério de aspiração é remover a classificação tabu de uma solução que é melhor do que a melhor solução encontrada até agora.

A memória de longo prazo é freqüentemente usada por um procedimento de pesquisa tabu para fins de intensificação ou diversificação. Estratégias de intensificação podem ser usadas para explorar recursos historicamente considerados bons, enquanto

as estratégias de diversificação encorajam o processo de busca a explorar regiões não visitadas do espaço da solução. Comumente, a memória baseada em frequência é usada para implementar estratégias de intensificação e diversificação.

A dimensão qualidade refere-se à capacidade de diferenciar o mérito das soluções visitadas durante a busca, enquanto a dimensão influência se refere ao impacto das escolhas feitas durante o processo de busca tanto na qualidade quanto na estrutura das soluções visitadas.

A seguir, descrevemos as duas heurísticas propostas para encontrar limites inferiores para *BLMCLP:* uma heurística GRASP e uma heurística híbrida GRASP-Tabu, que é uma combinação de GRASP e pesquisa tabu.

3.1. Heurística GRASP

A heurística GRASP proposta constrói iterativamente uma solução inicial viável que é posteriormente aprimorada por uma busca local que explora uma vizinhança. A solução incumbente é atualizada cada vez que uma iteração obtém uma solução melhor. O algoritmo termina quando o critério de encerramento é atendido. Aqui, primeiro descrevemos o procedimento guloso randomizado usado na heurística GRASP proposta e, em seguida, descrevemos o procedimento de busca local.

3.1.1. Procedimento aleatório ganancioso

Para encontrar soluções viáveis iniciais para *BLMCLP,* um conjunto $S \subset eu_1$ de p as instalações devem ser selecionadas de forma que a demanda do cliente seja coberta por a empresa é maximizada. O procedimento guloso randomizado da heurística GRASP proposta é um procedimento iterativo que constrói uma solução inicial viável. Em cada iteração, um recurso do conjunto de instalações potenciais eu_1 é selecionado e os clientes cobertos são atribuídos às suas instalações preferidas entre as instalações existentes laços. Uma função gananciosa é usada para avaliar a contribuição para o função objetivo do líder de cada instalação candidata em eu_1 Para cada instalação candidata, a função gananciosa mede o decomando que pode ser coberto pela instalação, considerando a reação do seguidor. A demanda que pode ser capturada por uma instalação candidata é a demanda de clientes dentro de seu raio de cobertura e que não é coberta por nenhuma instalação aberta ou uma demanda que é coberta ered por alguma instalação em eu_2 mas o cliente prefere as instalações candidatas entre todas as instalações abertas. A seguir, explicamos o construtivo procedimento em detalhes.

Deixar *Abordado* ser o conjunto de clientes atendidos por instalações abertas. Inicialmente, *Abordado* é igual a J_2 Além disso, $A: J \rightarrow S \cup eu_2$, Onde $A(j) := i$ se cliente j está alocado à instalação $eu.$ Para cada $j \in J$, $A(j) :=$ argmax $_{eu \in (S \cup eu_2) \cap Eu\, j\}} \{g_{eu\, j}\}$, E se $(S \cup eu_2) \cap I(j) = \emptyset$. Por outro lado, $A(j) := nulo,$ o que significa que o cliente j não está dentro da cobertura raio de qualquer instalação aberta e, portanto, $j \in Co\ V\ ered.$ Inicialmente, S é igual a $\emptyset$ e é atualizado ao final de cada iteração. Para seensinam a facilidade a ser aberta a cada iteração, o valor ganancioso C_{eu} é calculado para cada instalação potencial $eu \in eu_1 \setminus S$. Como mencionado antes visivelmente, este valor ganancioso C_{eu} mede o aumento no valor da função objetivo do líder para abrir uma instalação no local $eu.$ Em outro palavras, para cada cliente j coberto por eu (ou seja, $j \in J(i)$), a demanda é adicionado a C_{eu} em uma das duas situações a seguir: 1) j não foi coberto anteriormente, $j \in Abordado,$ e 2) j foi coberto por um ity $A(j) \in eu_2$ e eu é mais preferido do que $A(j).$ Para cada $eu \in eu_1 \setminus S,$ nós temos o seguinte.

$$\beta_{ij} = \begin{cases} D_j & \text{E se } j \notin Co\ V\ ered \text{ ou } (g_{ij} > g_{A(j),\, j} \text{ e } A(j) \in eu_2) \\ 0 & \text{por outro lado} \end{cases}, \quad \forall\, j \in J(i)$$

portanto

$$C_i = \sum_{j \in J(i)} \beta_{eu\, j} \tag{11}$$

Deixar $C_{\min} = \min_{eu \in eu_1 \setminus S\{C_{eu}\}}$ e $C_{\max} = \max_{eu \in eu_1 \setminus S\{C_{eu}\}}$. Além disso, deixe $T = w_{\min} + \alpha\,(C_{\max} - C_{\min})$ ser um valor limite, onde $\alpha \in (0, 1)$ é um parâmetro que controla o grau de ganância ou dominação do procedimento ganancioso. Selecionamos aleatoriamente uma instalação eu_* de uma lista restrita de candidatos RCL que contém instalações candidatas laços de quem $C_{eu} \geq T.$ Instalação eu é adicionado ao conjunto de instalações abertas $S.$ No final de cada iteração, o *Abordado* conjunto é atualizado e cusclientes $j \in J(eu_*)$ são alocados para eu_* se eles não foram cobertos anteriormente ou se eu_* é mais preferido do que sua atribuição anterior $A(j).$ Além disso, $A(j)$ é atualizado para todos $j \in Abordado.$ Em outras palavras, $Covered$ está configurado para $\bigcup_{eu \in (S \cup eu_2)} J(i),$ que é o conjunto de todos os clientes dentro o raio de cobertura de alguma instalação aberta. Para todos $j \in Coberto,\ A(j)$ é definido para argmax $_{eu \in (S \cup eu_2) \cap Eu\, j\}} \{g_{eu\, j}\},$ o que significa que cada cliente coberto é alocado ao seu local preferido entre todos os cátions em $S \cup eu_2$ A fase construtiva termina quando p instalações são selecionadas. O procedimento ganancioso aleatório do GRASP proposto heurística é dada em Algoritmo 3.1 .

Algoritmo 3.1 Procedimento aleatório ganancioso.

```
1: função GreedyRandomized (α)
2:     SolValue ← 0
3:     selecionado ← 0
4:     S ← ∅
5:     Co V ered ← J2
6:     para todos j ∈ J2 Faz
7:         A(j) ← argmax_{eu ∈ eu2} { g_{eu j} : j ∈ J(i) }
8:     fim para
9:     repetir
10:        para todos eu ∈ eu1 \ S Faz
11:            para todos j ∈ J(i) Faz
12:                β_{eu j} ← { D_j   E se j ∉ Co V ered ou (g_{ij} > g_{A(j), j} e A(j) ∈ eu2)
                              { 0     por outro lado
13:            fim para
14:            C_eu ← 0
15:            para todos j ∈ J1 Faz
16:                C_eu ← C_i + β_{eu j}
17:            fim para
18:        fim para
19:        C_min ← min_{eu ∈ eu1 \ S{C_eu}}
20:        C_max ← max_{eu ∈ eu1 \ S{C_eu}}
21:        T ← C_min + α (C_max - C_min)
22:        RCL ← { eu ∈ eu1 \ S: w_eu ≥ T}
23:        selecionar eu* ra
24:        S ← S ∪ { eu* } raramente de RCL
25:        selecionado ← selecionado + 1
26:        Co V ered ← ∪_{eu ∈ (S ∪ eu2)} J(i) {}
27:        A(j) ← argmax_{eu ∈ (S ∪ eu2) ∩ Eu j)} g_{eu j}
28:        DemandCo V ered ← DemandCo V ered + w_{eu*}
29:    até selecionado = p
30:    Retorna SolValue
31: função final
```

3.1.2. Procedimento de busca local

Cada solução inicial gerada pelo procedimento guloso aleatório da heurística GRASP é aprimorada por um procedimento de busca local. Desta forma, o procedimento de busca local em uma heurística GRASP é usado como uma estratégia de intensificação (Resende & GonzálezVelarde, 2003) Em nosso caso, o procedimento de busca local explora a vizinhança onde uma instalação aberta é trocada por um instalação fechada, ou seja, a bolsa abre uma instalação fechada $eu_1 \in eu_1 \setminus S$ e fecha uma instalação aberta $eu_2 \in S.$ Uma solução para o problema de localização de cobertura máxima com pedido de preferência do cliente é repre-

enviado por um par (*S, A),* Onde *S* é o conjunto de instalações abertas e *UMA: J* → *S* ∪ *eu* 2, conforme definido na seção anterior. Portanto, a vizinhança *N (S, A)* de uma determinada solução (*S, A)* é expresso da seguinte forma.

$$N(S,A) = \{(S', UMA'): S' = S \setminus \{i_2\} \cup \{eu_1\}, eu_2 \in S, eu_1 \in eu_1 \setminus S, \forall j \in \cup_{eu \in (S' \cup eu_2)} J(i), UMA'(j) = \text{argmax}_{eu \in (S' \cup eu_2) \cap Eu\,j)} g_{eu\,j}, UMA'(j) = nulo, \forall j \notin \cup_{eu \in (S' \cup eu_2)} J(i)\}$$

Aqui, *N (S, A)* é o conjunto de todas as soluções que podem ser obtidas abrindo uma instalação fechada e fechando uma instalação aberta.

Deixar

$$\delta_{j\,eu_1,\,eu_2} = \begin{cases} -D_j, & \text{E se } A(j) = i_2 \text{ e } (j \not\in \cup_{eu \in Si_2) \cup \{eu_1)} J(i) \\ & \text{ou argmax}_{eu \in (S \cup eu_2) \setminus \{eu_2) \cup \{eu_1) \{} g_{eu\,j)} \in eu_2) \\ & \text{E se } j \in J(eu_1) \text{ e } (j \in \cup_{eu \in (S \cup eu_2)} J(i)) \\ D_j, & \text{ou } (A(j) \in eu_2 \text{ e } g_{A(j),\,j} < g_{eu_1,\,j)}) \\ 0, & \text{por outro lado} \end{cases}$$

para todos *j* ∈ *J,* denotam a mudança de função objetivo dos líderes associados atado com o cliente *j* quando trocamos instalações *eu* 1 com facilidade *eu* 2 Como pode ser visto, o valor da função objetivo diminui em D_j se a demanda do cliente *j* é coberto pela instalação *eu* 2 em solução (*S, A),* ou seja, *A (j) = i* 2 e

- a demanda do cliente *j* não pode ser coberto pelo conjunto do instalações abertas do líder *S* {*i* 2} ∪ { *eu* 1} após a troca ser realizada, ou
- a demanda do cliente *j* será alocado para alguma instalação em *eu* 2 de acordo com as preferências (ou seja, argmax $_{eu \in (S \cup eu_2) \setminus \{eu_2) \cup \{eu_1) \{} g_{eu\,j)} \in eu_2)$.

Por outro lado, o valor da função objetivo aumentará em D_j quando a demanda do cliente *j* é coberto pela instalação *eu* 1 (ou seja, *j* ∈ *J (eu* 1)) e

- a demanda do cliente *j* não é coberto pela solução atual (ou seja, *j* ∈ ∪ *eu* ∈ (*S* ∪ *eu* 2) *J (i)),* ou
- a demanda do cliente *j* é alocado para alguma instalação em *eu* 2; no entanto, de acordo com as preferências do cliente, será real citado para instalação *eu* 1 Porque $g_{A(j),\,j} < g_{eu\,1,\,j}$.

Em seguida, a mudança de função objetivo do líder associada à troca de instalações *eu* 1 com facilidade *eu* 2 é expresso da seguinte forma.

$$_{eu_1,\,eu_2} = \sum_{j \in J} \delta_{j\,eu,\,eu_2}$$

Aqui, $_{eu_1,\,eu_2}$ mede o impacto na função objetivo do líder valor de ção calculando a mudança de demanda do cliente quando fa- cilidade *eu* 1 ∈ *eu* 1 *S* é aberto e o impacto na demanda do cliente mudar quando facilidade *eu* 2 ∈ *S* é encerrado considerando a reação do seguidor em relação às preferências do cliente.

A busca local da heurística GRASP proposta usa a melhor estratégia de melhoria (ou seja, a melhor solução em *N (S, A)* é selecionado) e termina quando o valor da função objetivo não pode ser melhorado mais (todas as soluções em *N (S, A)* são piores do que a solução atual). O procedimento de busca local da heurística GRASP proposta é dado em Algoritmo 3.2 .

A heurística GRASP proposta é fornecida em Algoritmo 3.3 . O O critério de término é um número predeterminado de iterações sem melhorias.

3.2. Heurística Híbrida GRASP-Tabu

Algoritmos híbridos têm sido amplamente usados para resolver problemas difíceis (Blum, 2010; Lozano e García-Martínez, 2010; Talbi, 2002) Essas combinações aumentam as vantagens de usar uma única metodologia para obter melhores soluções. A combinação de GRASP com pesquisa Tabu foi estudada pela primeira vez em Laguna e González-Velarde (1991) . Métodos híbridos usando GRASP com Tabu search foram usados para encontrar limites viáveis para muitos problemas (Abdinnour-Helm & Hadley, 2000; Colomé e Serra, 2001; Delmaire, Díaz, Fernández, & Ortega, 1999; Díaz, Luna e Zetina, 2013; Duarte & Martí, 2007; Laguna e González-Velarde, 1991; Lim e Wang, 2004) Neste estudo, propomos um procedimento que combina pesquisa GRASP e pesquisa Tabu. O procedimento de busca Tabu é usado no lugar da busca local na fase de aprimoramento do algoritmo.

A pesquisa tabu é comumente usada para resolver problemas difíceis. Foi proposto pela primeira vez em Glover (1986) e é descrito em detalhes em Glover (1989) . Nesta metodologia, as memórias de curto e longo prazo são usadas para escapar dos ótimos locais a fim de explorar melhor o espaço de solução. Três elementos devem ser definidos para implementar a pesquisa tabu: atributos tabu, tabu-tenure (um parâmetro que indica o número de iterações nas quais um atributo estará ativo) e o critério de aspiração (critério para substituir atributos tabu-ativos). Atributos ativos de tabu são aqueles proibidos em uma solução. A memória de curto prazo de uma pesquisa tabu contém uma lista de atributos selecionados que ocorrem em soluções visitadas recentemente para evitar a revisitação dessas soluções.

Em nossa implementação, os atributos tabu são definidos de maneira simples. Uma vez que uma instalação *eu* ∈ *S* estiver fechado, ele deve permanecer fechado por várias iterações. Para implementar isso, uma lista de tamanho fixo com uma disciplina de primeiro acesso é mantida. Portanto, se a facilidade *eu* ∈ *S* está fechado na iteração *t,* o índice de facilidade *eu* será anexado para a lista de tabu. Se o tamanho fixo da lista de tabu for *r,* instalação de abertura *eu* será proibido em iterações *t* + 1, *t* + 2,. . . , *t* + *r.* Também usamos o critério de aspiração padrão toda vez que uma solução com um atributo tabu-ativo é melhor do que a solução atual, ou seja, as restrições tabu são ignoradas. Portanto, em qualquer iteração do procedimento, as soluções vizinhas candidatas serão aquelas que não contêm atributos tabu-ativos ou satisfazem o critério de aspiração (linha 14 em Algoritmo 3.4) A heurística híbrida GRASP-Tabu proposta é fornecida em Algoritmo 3.5 . O critério de encerramento é um número predeterminado de iterações sem melhorias.

4. Experimentação computacional

Para avaliar o desempenho dos métodos heurísticos propostos, foram utilizadas 60 instâncias geradas aleatoriamente. As instâncias foram geradas pelo procedimento descrito em Resende (1998) . O procedimento utilizado neste estudo pode ser resumido da seguinte forma: (1) *t* os pontos são gerados em uma unidade quadrada e a matriz das distâncias euclidianas é calculada, (2) *m* do *t* os pontos são selecionados aleatoriamente como locais de instalações potenciais e os restantes *n* = *m* - *t* pontos formam o conjunto de clientes, (3) 10% do *m* potencial instalação lo- cátions são selecionados aleatoriamente para formar *eu* 2, e (4) a preferência do cliente em relação às instalações é gerada usando o procedimento descrito dentro Cánovas et al. (2007) . Com este procedimento, dez instâncias dos seguintes tamanhos *n* × *m* foram gerados: 225 × 25, 450 × 50, 675 × 75, 900 × 100, 1350 × 150 e 1800 × 200. Um raio de cobertura diferente foi usado ao gerar cada conjunto de instâncias:
0,80, 0,70, 0,50, 0,30, 0,25 e 0,20, respectivamente. O *p* os valores dos parâmetros foram definidos como 3, 6, 10, 13, 20 e 27 para instâncias de tamanho 225 × 25, 450 × 50, 675 × 75, 900 × 100, 1350 × 150 e 1800 × 200, respectivamente.

Para avaliar a qualidade das soluções obtidas pelas heurísticas propostas, a reformulação em nível único foi codificada e executada no software comercial FICO XPRESS 7.8. Ambas as heurísticas (a heurística GRASP e a heurística híbrida GRASP-Tabu) foram codificadas em C ++. A reformulação de nível único e ambas as heurísticas foram executadas em um processador Intel Xenon (R) a 3,10 GHz com 8 GB de RAM. A experimentação computacional para ambas as heurísticas e os resultados obtidos são discutidos a seguir.

O FICO XPRESS Optimization Suite fornece uma estrutura de programação matemática com diferentes algoritmos para resolver problemas LP e problemas LP inteiros mistos (MIP). Para problemas de LP,

Algoritmo 3.2 Procedimento de busca local.

função LocalSearch (*SolValue)*
  Melhor ← *SolValue*
  Pare ← **falso**
  repetir
    para todos $eu_2 \in S$ e $eu_1 \in eu_1 \setminus S$ **Faz**
      para todos $j \in J$ **Faz**
        $\delta_{j,eu_1,eu_2} = \begin{cases} -D_j, & \text{E se } A(j) = i_2 \text{ e } (j / \in \cup_{eu \in Si_2) \cup \{eu_1\}} J(i) \text{ ou argmax}_{eu \in (S \cup eu_2) \setminus \{eu_2\} \cup \{eu_1\}\{g_{eu\,j}\}} \in eu_2) \\ D_j, & \text{E se } j \in J(eu_1) \text{ e } (j \in \cup_{eu \in (S \cup eu_2)} J(i)) \text{ ou } (A(j) \in eu_2 \text{ e } g_{A(j),j} < g_{eu_1,j})) \\ 0, & \text{por outro lado} \end{cases}$
      fim para
      $_{eu_1,eu_2} \leftarrow 0$
      para todos $j \in J$ **Faz**
        $_{eu_1,eu_2} \leftarrow {}_{eu_1,eu_2} + \delta_{j,eu_2}$
      fim para
    fim para
    $\leftarrow \max\{{}_{eu_1,eu_2} : eu_2 \in S \text{ e } eu_1 \in eu_1 \setminus S\}$
    E se > 0 **então**
      $(eu^*_{1,eu_2}) \in \text{argmax}_{eu_2 \in Si_1 \in eu_1 \setminus S\{i_1,eu_2}\}$
      $S \leftarrow S \cup \{eu_1\} \setminus \{eu_2\}$
      Melhor ← *Melhor* +
      para todos $j \in J$ **Faz** {}
        $A(j) \leftarrow \text{argmax}_{eu \in (S \cup eu_2) \cap Eu\,j)} g_{eu\,j}$
      fim para
    senão
      Pare ← **verdadeiro**
    fim se
  até *Pare*
  Retorna *Melhor*
função final

Algoritmo 3.3 Heurística GRASP.

função APERTO(*Número de iterações,* $\alpha_{inicial}$, α_{final}, *ItersF ixed, Incremento)*
  IterCount ← 1
  $\alpha \leftarrow \alpha_{inicial}$
  Melhor ← 0
  enquanto (*IterCount* ≤ *Número de Iterações)* **Faz**
    E se (*IterCount* **mod** *Iter f ixado* = 0) **então**
      E se ($\alpha = \alpha_{final}$ **)então**
        $\alpha \leftarrow \alpha_{inicial}$
      senão
        $\alpha \leftarrow \alpha +$ *Incremento*
      fim se
    fim se
    SolValue ← *GreedyRandomized (* α*)*
    SolValue ← *LocalSearch (SolValue)*
    E se (*SolValue> Best)* **então**
      Melhor ← *SolValue*
      IterCount ← 0
    fim se
    IterCount ← *IterCount* + 1
  terminar enquanto
  Retorna *Melhor*
função final

inclui os algoritmos primal simplex, dual simplex e de barreira de Newton. Para problemas de MIP, o solucionador fornece um poderoso framework de ramificação e limite que usa métodos heurísticos para determine boas soluções e planos de corte para fortalecer os relaxamentos LP. O uso de heurísticas e planos de corte para fornecer limites primários e duais pode permitir uma redução considerável do esforço enumerativo. Além disso, o solucionador MIP usa procedimentos de pré-solução que podem ajudar a reduzir a matriz do problema e, por sua vez, a dimensão do problema, tornando-o mais fácil de resolver. Em testes preliminares com FICO XPRESS, observou-se que, para as instâncias maiores, a execução da reformulação de nível único para cada instância poderia demorar mais de 15 h. Além disso, em testes preliminares, observou-se que a execução das heurísticas propostas para cada instância nunca ultrapassou 250 s. Desse modo,

Aqui, descrevemos uma avaliação da heurística GRASP proposta. A heurística GRASP tem dois parâmetros que precisam ser ajustados: o α valor e o critério de término (número de iterações sem melhoria). A seleção do α parâmetro na fase de construção de uma heurística GRASP pode afetar a qualidade da solução; portanto, é muito importante selecionar um valor apropriado para α. No entanto, devido à dificuldade em definir um valor para

α que é adequado para todas as instâncias de um determinado problema, várias opções foram usadas em trabalhos anteriores. Uma estratégia de tentativa e erro é freqüentemente usada para encontrar o valor de parâmetro mais adequado para cada instância de dados. Outras implementações (Delmaire et al., 1999; Prais e Ribeiro, 2000; Ríos-Mercado & Fernández, 2009) sintonizar automaticamente o α valor usando um procedimento reativo. Esses procedimentos são geralmente baseados em probabilidades de selecionar α valores de um conjunto discreto de valores, ou seja, o α valor usado em cada iteração. Para que esses procedimentos funcionem corretamente, o algoritmo heurístico deve realizar

Algoritmo 3.4 Procedimento de pesquisa tabu.

função TabuSearch (*SolValue)*
  Melhor ← *SolValue*
  StopCriterion ← **falso**
  repetir
    para todos $eu_2 \in S$ e $eu_1 \in eu_1 \setminus S$ **Faz**
      para todos $j \in J$ **Faz**
        $\delta_{j,eu_1,eu_2} = \begin{cases} -D_j, & \text{E se } A(j) = i_2 \text{ e } (j \notin \cup_{eu \in S \cup \{eu_1\}} J(i) \text{ ou } \text{argmax}_{eu \in (S \cup eu_2) \setminus \{eu_1\}} g_{eu,j} \in eu_2) \\ D_j, & \text{E se } j \in J(eu_1) \text{ e } (j \in \cup_{eu \in (S \cup eu_2)} J(i)) \text{ ou } (A(j) \in eu_2 \text{ e } g_{A(j),j} < g_{eu_1,j})) \\ 0, & \text{por outro lado} \end{cases}$
      fim para
      $_{eu_1,eu_2} \leftarrow 0$
      para todos $j \in J$ **Faz**
        $_{eu_1,eu_2} \leftarrow {}_{eu_1,eu_2} + \delta_{ij,eu_2}$
      fim para
    fim para
    Candidatos ← $\{(eu_1, eu_2): eu_1 \in eu_1 \setminus S, eu_2 \in S \text{ e } (eu_2 \in TabuList \text{ ou } SolValue + {}_{eu_1,eu_2} > Melhor)\}$
    ← max $\{eu_{1,eu_2} : (eu_1, eu_2) \in$ *adicionar uma*$\}$
    $(eu^*_{1,eu_2}) \in \text{argmax}\{eu_{1,eu_2} : (eu_1, eu_2 \in Candidatos\}$
    $S \leftarrow S \cup \{eu_1\} \setminus \{eu_2\}$
    para todos $j \in J$ **Faz**
      A (j) ← $\text{argmax}_{eu \in (S \cup eu_2) \cap Eu_j)} g_{eu,j}$ {}
    fim para
    Atualizar *TabuList*
    SolValue ← *SolValue +*
    E se *SolValue> Best* **então**
      Melhor ← *SolValue*
    fim se
    Atualizar *StopCriterion*
  até (não *StopCriterion)*
  Retorna *Melhor*
função final

muitas iterações. Neste estudo, a qualidade da solução dos algoritmos propostos é suficiente sem exigir muitas iterações; assim, um procedimento reativo não é necessário. Em vez disso, em todos os testes, o α parâmetro foi variado de 0,75 a 0,95, começando em 0,75 e aumentando α por 0,05 a cada cinco iterações.

O critério usado para encerrar a heurística GRASP é um número fixo de iterações sem melhorias. Dois conjuntos de testes foram realizados para avaliar a qualidade dos resultados obtidos com a heurística GRASP proposta e a robustez heurística. No primeiro conjunto de testes, o critério de término foi de 50 iterações sem melhorias, e no segundo conjunto de testes, o critério de término foi de 100 iterações sem melhorias. Cada conjunto de testes consistia em executar cada instância de teste cinco vezes. Para avaliar a qualidade da solução, a melhor solução das cinco execuções foi comparada com a solução ótima ou mais conhecida de cada instância e, para avaliar a robustez heurística, o desvio entre a melhor e a pior solução foi medido.

Tabelas 1 e 2 descrever os resultados obtidos com a heurística GRASP proposta. A primeira coluna mostra o tamanho das instâncias, e as próximas três colunas mostram o desvio percentual médio das melhores soluções obtidas em cada instância com relação à solução ótima ou mais conhecida (Avg. Best%), o desvio percentual médio do média da solução em relação à solução ótima ou mais conhecida (Avg. Mean%), e o desvio percentual médio das piores soluções obtidas em relação à solução ótima ou mais conhecida (Avg. Worst%). Finalmente, as duas últimas colunas mostram o tempo médio de CPU em segundos exigido pela heurística GRASP e XPRESS, respectivamente.

tabela 1
Resultados heurísticos GRASP (50 iterações).

Tamanho	Média Melhor %	Média Mau %	Média Pior %	CPU (seg)	XPRESS CPU (s)
225 × 25	0,000	0,000	0,000	0,0	1,9
450 × 50	0,000	0,001	0,005	0,0	6,4
675 × 75	0,000	0,018	0,031	1,5	21,1
900 × 100	0,000	0,002	0,010	4,5	65,3
1350 × 150	0,000	0,004	0,018	24,1	2562,6
1800 × 200	0,000	0,011	0,032	79,1	> 10.800,0

mesa 2
Resultados heurísticos GRASP (100 iterações).

Tamanho	Média Melhor %	Média Mau %	Média Pior %	CPU (seg)	XPRESS CPU (s)
225 × 25	0,000	0,000	0,000	0,0	1,9
450 × 50	0,000	0,000	0,000	0,2	6,4
675 × 75	0,000	0,018	0,031	3,1	21,1
900 × 100	0,000	0,002	0,010	9,0	65,3
1350 × 150	0,000	0,001	0,003	44,3	2562,6
1800 × 200	0,000	0,005	0,015	142,6	> 10.800,0

Observe que, para instâncias de tamanho 1800 × 200, XPRESS não era capaz de encontrar a solução ótima no limite de tempo de três horas para nove das dez instâncias. Além disso, para essas nove instâncias, o melhor valor de solução viável obtido pelo XPRESS foi pior do que a solução obtida pela heurística GRASP. Em ambos os conjuntos de testes, o algoritmo encontrou a solução ideal ou mais conhecida

Algoritmo 3.5 Heurística híbrida GRASP-Tabu.

```
1: função Híbrido( Número de iterações, α inicial , α final , ItersF ixed,
   Incremento)
2:     IterCount ← 1
3:     α ← α inicial
4:     Melhor ← 0
5:     enquanto ( IterCount ≤ Número de Iterações) Faz
6:         E se ( IterCount mod Iter f ixado = 0) então
7:             E se ( α = α final ) então
8:                 α ← α inicial
9:             senão
10:                α ← α + Incremento
11:            fim se
12:        fim se
13:        SolValue ← GreedyRandomized ( α)
14:        SolValue ← TabuSearch (SolValue)
15:        E se ( SolValue> Best) então
16:            Melhor ← SolValue
17:            IterCount ← 0
18:        fim se
19:        IterCount ← IterCount + 1
20:    terminar enquanto
21:    Retorna Melhor
22: função final
```

Tabela 3
Resultados híbridos GRASP-Tabu.

Tamanho	Média Melhor %	Média Mau %	Média Pior %	CPU (seg)	XPRESS CPU (s)
225 × 25	0,000	0,000	0,000	0,0	1,9
450 × 50	0,000	0,000	0,000	1,4	6,4
675 × 75	0,000	0,006	0,031	7,6	21,1
900 × 100	0,000	0,000	0,000	13,9	65,3
1350 × 150	0,000	0,000	0,000	58,8	2562,6
1800 × 200	0,000	0,001	0,001	172,8	> 10.800,0

para todas as instâncias em pelo menos uma das cinco execuções. Além disso, o desvio percentual médio das piores soluções obtidas em relação à solução ótima ou mais conhecida nunca excedeu
0,032% para o experimento com critério de término de 50 iterações sem melhoria e 0,031% para o experimento com 100 iterações. Claramente, o algoritmo é mais robusto ao aumentar o número de iterações sem melhora no critério de terminação, visto que o desvio entre a melhor e a pior solução é reduzido.

Tabela 3 mostra os resultados da heurística híbrida GRASP-Tabu proposta. Tal como acontece com os testes heurísticos GRASP, o α parâmetro foi variado de 0,75 a 0,95, começando em 0,75 e aumentando em 0,05 a cada cinco iterações. Além disso, para este teste, o número de iterações sem melhorias foi definido para 50 e a posse do Tabu foi definida para sete iterações. Novamente, o teste consistiu em cinco execuções para cada instância de teste. Para avaliar a qualidade da solução, a melhor solução das cinco execuções foi comparada com a solução ótima ou mais conhecida de cada instância e, para avaliar a robustez heurística, foi medido o desvio entre a melhor e a pior solução. Claramente, a heurística híbrida GRASP-Tabu fornece os melhores resultados de qualidade. O desvio percentual médio da média da solução em relação à solução ótima ou mais conhecida no pior caso foi 0,006%, que é muito menor em comparação com 0,018% da heurística GRASP. Além disso, × 75. Em apenas três das 60 instâncias, o algoritmo não encontrou a solução ótima ou mais conhecida nas cinco execuções, mostrando claramente que o algoritmo é robusto. Finalmente, al-

Tabela 4
Resultados detalhados do híbrido GRASP-Tabu.

Nome	Ótimo ou mais conhecido	Melhor %	Média %	Pior %	CPU
mcb225 × 25-01	20.870	0,000	0,000	0,000	<1,00
mcb225 × 25-02	18.562	0,000	0,000	0,000	<1,00
mcb225 × 25-03	19.546	0,000	0,000	0,000	<1,00
mcb225 × 25-04	21.236	0,000	0,000	0,000	<1,00
mcb225 × 25-05	20.160	0,000	0,000	0,000	<1,00
mcb225 × 25-06	19.458	0,000	0,000	0,000	<1,00
mcb225 × 25-07	20.243	0,000	0,000	0,000	<1,00
mcb225 × 25-08	20.511	0,000	0,000	0,000	<1,00
mcb225 × 25-09	19.815	0,000	0,000	0,000	<1,00
mcb225 × 25-10	20.210	0,000	0,000	0,000	<1,00
mcb450 × 50-11	73.623	0,000	0,000	0,000	1,80
mcb450 × 50-12	79.903	0,000	0,000	0,000	1,20
mcb450 × 50-13	76.999	0,000	0,000	0,000	2,20
mcb450 × 50-14	68.361	0,000	0,000	0,000	1,20
mcb450 × 50-15	73.454	0,000	0,000	0,000	1,60
mcb450 × 50-16	72.540	0,000	0,000	0,000	1,20
mcb450 × 50-17	79.528	0,000	0,000	0,000	1,20
mcb450 × 50-18	75.316	0,000	0,000	0,000	1,20
mcb450 × 50-19	74.286	0,000	0,000	0,000	1,00
mcb450 × 50-20	80.620	0,000	0,000	0,000	1,20
mcb675 × 75-21	182.743	0,000	0,000	0,000	7,40
mcb675 × 75-22	176.540	0,000	0,061	0,306	10,40
mcb675 × 75-23	183.008	0,000	0,000	0,000	7,00
mcb675 × 75-24	177.188	0,000	0,000	0,000	8,20
mcb675 × 75-25	185.892	0,000	0,000	0,000	8,00
mcb675 × 75-26	176.562	0,000	0,000	0,000	6,00
mcb675 × 75-27	201.912	0,000	0,000	0,000	6,80
mcb675 × 75-28	188.272	0,000	0,000	0,000	6,20
mcb675 × 75-29	203.018	0,000	0,000	0,000	8,80
mcb675 × 75-30	183.625	0,000	0,000	0,000	7,20
mcb900 × 100-31	300.826	0,000	0,000	0,000	12,60
mcb900 × 100-32	325.846	0,000	0,000	0,000	12,60
mcb900 × 100-33	328.118	0,000	0,000	0,000	15,00
mcb900 × 100-34	309.788	0,000	0,000	0,000	11,60
mcb900 × 100-35	318.625	0,000	0,000	0,000	15,20
mcb900 × 100-36	319.178	0,000	0,000	0,000	13,40
mcb900 × 100-37	314.034	0,000	0,000	0,000	15,80
mcb900 × 100-38	327.188	0,000	0,000	0,000	12,00
mcb900 × 100-39	317.017	0,000	0,000	0,000	17,80
mcb900 × 100-40	298.779	0,000	0,000	0,000	13,20
mcb1350 × 150-41	693.053	0,000	0,000	0,000	59,20
mcb1350 × 150-42	723.185	0,000	0,000	0,000	65,80
mcb1350 × 150-43	742.178	0,000	0,000	0,000	61,20
mcb1350 × 150-44	688.609	0,000	0,000	0,000	66,00
mcb1350 × 150-45	684.277	0,000	0,000	0,000	61,80
mcb1350 × 150-46	695.860	0,000	0,000	0,000	44,00
mcb1350 × 150-47	690.537	0,000	0,000	0,000	48,60
mcb1350 × 150-48	693.398	0,000	0,000	0,000	61,60
mcb1350 × 150-49	731.687	0,000	0,000	0,000	50,60
mcb1350 × 150-50	723.746	0,000	0,000	0,000	69,00
mcb1800 × 200-51	1.281.284	0,000	0,000	0,000	144,40
mcb1800 × 200-52	1.356.246	0,000	0,006	0,010	211,80
mcb1800 × 200-53	1.258.479	0,000	0,000	0,000	163,80
mcb1800 × 200-54	1.314.660	0,000	0,001	0,004	168,60
mcb1800 × 200-55	1.300.838	0,000	0,000	0,000	145,20
mcb1800 × 200-56	1.256.017	0,000	0,000	0,000	157,60
mcb1800 × 200-57	1.290.808	0,000	0,000	0,000	206,20
mcb1800 × 200-58	1.280.134	0,000	0,000	0,000	180,20
mcb1800 × 200-59	1.287.646	0,000	0,000	0,000	201,00
mcb1800 × 200-60	1.331.171	0,000	0,000	0,000	149,20

embora o tempo médio de CPU tenha sido maior para a heurística GRASPTabu híbrida, o tempo médio aumentou apenas ligeiramente do que o tempo de CPU da heurística GRASP e permaneceu muito menor do que o tempo de CPU do modelo matemático executado no FICO XPRESS.

Os resultados detalhados para cada instância são mostrados em Tabela 4 . A primeira coluna mostra o nome de cada instância e a segunda coluna mostra a solução ideal ou mais conhecida. As três colunas a seguir mostram o melhor, a média e o pior desvios percentuais das soluções em relação à solução ótima ou mais conhecida. A última coluna mostra o tempo médio de CPU em segundos.

5. Conclusões

Neste estudo, o problema de localização de cobertura máxima com a ordem de preferência do cliente foi investigado. Dois modelos diferentes para o problema foram apresentados, ou seja, um modelo de dois níveis e um modelo equivalente de um único nível. A equivalência entre esses dois modelos também foi discutida. Dois algoritmos heurísticos foram propostos para obter limites inferiores para a solução ótima do problema: uma heurística GRASP e uma heurística híbrida GRASP-Tabu que substitui a fase de busca local da heurística GRASP por um procedimento de busca Tabu.

Ambos os algoritmos foram testados com um conjunto de 60 instâncias geradas aleatoriamente, e diversos testes computacionais foram realizados para avaliar as heurísticas propostas. Em primeiro lugar, a reformulação de nível único foi resolvida para todas as instâncias utilizando software de programação matemática (FICO XPRESS) com duas finalidades: (1) avaliar a qualidade das soluções obtidas pelas heurísticas propostas e (2) avaliar a eficiência de uma exata método conforme o tamanho da instância aumenta.

As soluções heurísticas foram comparadas com soluções de reformulação do problema em um único nível. De acordo com os resultados dos testes computacionais, as duas heurísticas propostas forneceram soluções de boa qualidade. Em 51 das 60 instâncias de teste, a heurística proposta encontrou a solução ótima em pelo menos uma das cinco execuções. Além disso, para as nove instâncias restantes, para as quais a otimalidade das soluções não pôde ser veri fi cada no prazo de três horas com o FICO XPRESS, as heurísticas forneceram melhores limites inferiores do que os obtidos com o FICO XPRESS. No entanto, a heurística híbrida GRASP-Tabu proposta superou a heurística GRASP porque é mais robusta sem aumentar significativamente o tempo de CPU.

Os resultados indicam que o esforço enumerativo exigido pelo FICO XPRESS aumenta consideravelmente à medida que o tamanho da instância aumenta. Em testes preliminares, para as maiores instâncias, o FICO XPRESS exigiu mais de 15 h. Observe que, no pior caso, o tempo requerido pela heurística proposta nunca ultrapassa 250 s, demonstrando assim a eficiência das heurísticas propostas.

Métodos metaheurísticos e métodos de plano de corte são muito úteis para projetar métodos exatos para fins especiais. Metaheurísticas podem fornecer limites primários e incumbentes de alta qualidade nos métodos Branch & Bound, e os métodos de plano de corte melhoram os limites duplos. Juntos, eles podem reduzir o esforço enumerativo de um método exato. Uma direção futura desta pesquisa pode ser a combinação de heurísticas e planos de corte para o projeto de um método exato para o problema de localização de cobertura máxima com pedido de preferência do cliente.

Outra direção de pesquisa futura pode ser considerar métodos heurísticos para a formulação de dois níveis do problema com base em um jogo estratégico de dois jogadores, duas fases com informações perfeitas, semelhantes aos propostos em Robbins e Lunday (2016) .

Referências

Abdinnour-Helm, S., & Hadley, S. (2000). Heurística baseada em pesquisa tabu para vários andares layout das instalações. *International Journal of Production Research, 38 (* 2), 365-383.

Bard, J. (1998). *Otimização prática de dois níveis: Aplicações e algoritmos.* Kluwer Aca- Editores dmicos.

Belotti, P., Labbé, M., Ma ffi oli, F., & Ndiaye, M. (2007). Um método de ramificação e corte para o detestável *p*- problema mediano. *4OR, 5 (* 4), 299-314.

Blum, C. (2010). Metaheurísticas híbridas. *Computadores e Pesquisa Operacional, 37 (* 3), 430–431.

Camacho-Vallejo, J., Cordero-Franco, A., & González-Ramírez, R. (2014). Resolvendo o problema de localização de instalações de dois níveis sob preferências por um algoritmo evolucionário de Stackelberg. *Mathematical Problems in Engineering, 2014,* 1-14.

Cánovas, L., García, S., Labbé, M., & Marín, A. (2007). Uma formulação reforçada para o simples problema de localização da fábrica com o pedido. *Cartas de Pesquisa Operacional, 35 (* 2), 141-150.

Church, R. & ReVelle, C. (1974). O problema de localização de cobertura máxima. *Artigos em Ciência Regional, 32 (* 1), 101-118.

Colomé, R., & Serra, D. (2001). Escolha do consumidor em modelos de localização competitivos: Formulações e heurísticas. *Artigos em Ciência Regional, 80 (* 4), 439–464.

Colson, B., Marcotte, P., & Savard, G. (2007). Uma visão geral da otimização de dois níveis. *Annals of Operations Research, 153 (* 1), 235–265.

Delmaire, H., Díaz, JA, Fernández, E., & Ortega, M. (1999). GRASP reativo e tabu heurísticas baseadas em pesquisa para o problema de localização de planta capacitada de fonte única. *INFOR: Sistemas de Informação e Pesquisa Operacional, 37 (* 3), 194–225.

Díaz, JA, Luna, DE, & Zetina, CA (2013). Um algoritmo híbrido para o manufac- problema de formação de células. *Journal of Heuristics, 19 (* 1), 77-96.

Duarte, A., & Martí, R. (2007). Tabu search e GRASP para a máxima diversidade problema. *European Journal of Operational Research, 178 (* 1), 71–84.

Feo, T., & Resende, M. (1995). Procedimentos de busca adaptativa aleatória gananciosa. *Jour- final de Otimização Global, 6 (* 2), 109–133.

Glover, F. (1986). Caminhos futuros para programação inteira e links para inteligência artificial ligência. *Computadores e Pesquisa Operacional, 13 (* 5), 533-549.

Glover, F. (1989). Pesquisa tabu - parte I. *ORSA Journal of Computing, 1 (* 3), 190–206. Glover, F. (1990). Pesquisa Tabu parte II. *ORSA Journal on Computing, 2 (* 1), 4-32. Hakimi, SL (1983). Sobre a localização de novas instalações em um ambiente competitivo. *Euro- pean Journal of Operational Research, 12 (* 1), 29–35.

Hanjoul, P., & Thill, JC (1987). Elementos de análise planar em competição espacial. *Ciência Regional e Economia Urbana, 17 (* 3), 423–439.

Hansen, P., Jaumard, B., & Savard, G. (1992). Novas regras de ramificação e limite para linear programação de dois níveis. *SIAM Journal on Scienti fi c and Statistical Computing, 13 (* 5), 1194–1217.

Hansen, P., Kochetov, Y., & Mladenovi, N. (2004). *Limites inferiores para os não capacitados problema de localização da instalação com as preferências do usuário.* Groupe d'études et de recherche en analise des décisions, HEC Montreal.

Jeroslow, R. (1985). A hierarquia polinomial e um modelo simples para análise. *Programação Matemática, 32 (* 2), 146-164.

Kalashnikov, V., Dempe, S., Pérez-Valdéz, G., Kalashnikova, V., & Camacho– Vallejo, J. (2004). Programação e aplicativos de dois níveis. *Problemas Matemáticos em Engenharia, 31,* 1515–1526.

Kress, D., & Pesch, E. (2012). Localização competitiva sequencial nas redes. *europeu Journal of Operational Research, 217 (* 3), 483–499.

Laguna, M. e González-Velarde, JL (1991). Uma heurística de pesquisa para just-in– programação de tempo em máquinas paralelas. *Journal of Intelligent Manufacturing, 2 (* 4), 253–260.

Lee, JM, & Lee, YH (2012). Localização da instalação e problema de decisão de escala com clientes preferência do cliente. *Computadores e Engenharia Industrial, 63 (* 1), 184–191.

Lim, A., & Wang, F. (2004). Uma pesquisa tabu dinâmica suavizada GRASP incorporado para m-VRPTW. Dentro *Ferramentas com inteligência artificial, 2004. ICTAI 2004. 16ª conferência internacional IEEE sobre (* pp. 704–708).

Lozano, M., & García-Martínez, C. (2010). Metaheurísticas híbridas com evolução ários algoritmos especializados em intensi fi cação e diversi fi cação: Visão geral e relatório de progresso. *Computadores e Pesquisa Operacional, 37 (* 3), 481–497.

Maric´, M., Stanimirovic´, Z., & Milenkovic´, N. (2012). Métodos metaheurísticos para solv- lidar com o problema de localização de instalação não capacitada de dois níveis com as preferências dos clientes. *Notas Eletrônicas em Matemática Discreta, 39,* 43–50.

Ognjanovic´, I., GaševiÄ, D., & Bagheri, E. (2013). Uma estrutura estratificada para lidar com preferências condicionais: uma extensão do processo de hierarquia analítica. *Sistemas especialistas com aplicativos, 40 (* 4), 1094-1115.

Plastria, F. (2001). Localização estática de instalações competitivas: uma visão geral da otimização abordagens. *European Journal of Operational Research, 129 (* 3), 461–470.

Prais, M., & Ribeiro, C. (2000). GRASP reativo: Um aplicativo para uma decomposição de matriz problema de posição na atribuição de tráfego TDMA. *INFORMS Journal on Computing, 12 (* 3), 164-176. doi: 10.1287 / ijoc.12.3.164.12639.

Resende, MGC (1998). Calculando soluções aproximadas da cobertura máxima problema com GRASP. *Journal of Heuristics, 4 (* 2), 161–177.

Resende, MGC e González-Velarde, JL (2003). GRASP: procedimientos de busqueda miopes aleatorizados y adaptativos. *Inteligencia Arti fi cial, Revista Iberoamericana de Inteligencia Arti fi cial, 7 (* 19), 61-76.

Ríos-Mercado, E., & Fernández, R. (2009). Um reativo {GRASP} para um ter- problema de projeto de ritório com múltiplos requisitos de balanceamento. *Computadores e Pesquisa Operacional, 36 (* 3), 755–776.

Robbins, M., & Lunday, B. (2016). Uma formulação de dois níveis do preço da vacina pediátrica problema ing. *European Journal of Operational Research, 264,* 634–645. Russell, S., & Norvig, P. (2010). *Inteligência artificial: uma abordagem moderna.* Nova Jersey, Estados Unidos: Prentice Hall.

Talbi, EG (2002). Uma taxonomia de metaheurísticas híbridas. *Journal of Heuristics, 8 (* 5), 541-564.

Vasil'ev, IL, Klimentova, KB, & Kochetov, YA (2009). Novos limites inferiores para o problema de localização das instalações com as preferências dos clientes. *Matemática Computacional e Física Matemática, 49 (* 6), 1010–1020.

Vasilyev, IL, & Klimentova, KB (2010). O método de ramificação e corte para a instalação problema de localização com as preferências do cliente. *Journal of Applied and Industrial Mathematics, 4 (* 3), 441–454.