LI Simpósio Brasileiro de Pesquisa Operacional
Limeira–SP, 02 a 06 de setembro de 2019

GRASP HÍBRIDO PARA RESOLUÇÃO DO PROBLEMA DE LOCALIZAÇÃO DE FACILIDADES CAPACITADAS EM DOIS NÍVEIS

Pedro Henrique González[1,4], Gabriel Souto[1]
DEPIN [1] - Centro Federal de Educação Tecnológica Celso Suckow da Fonseca, Rio de Janeiro - RJ
pegonzalez@eic.cefet-rj.br, gabriel.augusto@aluno.cefet-rj.br

Geraldo Regis Mauri[2]
DCOMP [2] - Universidade Federal do Espírito Santo, Alegre - ES
geraldo.mauri@ufes.br

Glaydston Mattos Ribeiro[3], Luidi Simonetti[4]
{PET [3],PESC [4]}/COPPE - Universidade Federal do Rio de Janeiro, Rio de Janeiro - RJ
glaydston@pet.coppe.ufrj.br,luidi@cos.ufrj.br

RESUMO

O Problema de Localização de Facilidades Capacitadas em Dois Níveis (PLFC2n) é um problema importante na classe de problemas de cadeia de suprimentos. O problema consiste em definir locais para instalação de fábricas e depósitos visando o atendimento da demanda de um conjunto de clientes. Neste problema, um único tipo de produto deve ser transportado das fábricas para os clientes, passando por depósitos intermediários. O objetivo do problema é a redução dos custos de operação (abertura de instalações e o fluxo de produtos fábrica - depósito - cliente) respeitando as restrições de capacidade das instalações, capacidade dos depósitos e demanda dos clientes. Neste trabalho é apresentado um método híbrido que combina a meta-heurística GRASP com uma formulação de programação linear inteira mista do PLFC2n. Experimentos computacionais foram realizados utilizando instâncias base da literatura e mostram a eficácia do método proposto em relação aos demais métodos existentes na literatura.

PALAVRAS CHAVE. **Localização de Facilidades. Método Híbrido. GRASP. Local Branching.**

MH - Meta-heurísticas.

ABSTRACT

The Two-Stage Capacited Facility Location Problem (TSCFL) is a major problem in the supply chain class. The problem consists of defining locations for installing factories and warehouses in order to meet the demand of a group of customers. In this problem, a single type of product must be transported from the factories to the customers, through intermediate warehouses. The objective of the problem is to reduce the operating costs, ensuring restrictions on the capacity of the facilities, on the capacity of the warehouses and that the demand of the customers are fullfiled. In this work is presented a hybrid method that combines the GRASP meta-heuristic with a mixed integer linear programming formulation of TSCFL. Computational experiments were performed using benchmark instances. The results showed the efficency of the proposed method in relation to other methods presented in the literature.

KEYWORDS. **Facility Location. Hybrid Method. GRASP. Local Branching.**

MH - Metaheuristics.

https://proceedings.science/p/106904?lang=pt-br

1. Introdução

A chave do sucesso de uma organização pode estar diretamente relacionada à sua eficiência operacional em relação à distribuição geográfica de suas instalações físicas. Isso porque, ao localizar-se em locais estratégicos com incentivos fiscais e com boa malha rodoviária/ferroviária, pode-se reduzir os custos de distribuição da cadeia de suprimentos, aumentar o lucro e atender a demanda (os clientes) de forma rápida e eficiente.

A definição de um melhor sistema de distribuição para uma organização pode ser obtida por meio de um problema de localização de facilidades, cujo objetivo é definir os locais para instalação de facilidades (fábricas, depósitos, pontos de venda, etc.) com o objetivo de atender um conjunto de clientes com um custo mínimo, por exemplo. Um caso específico desse problema é o Problema de Localização de Facilidades Capacitadas em Dois Níveis (PLFC2n) que, além de considerar restrições de capacidade das facilidades, considera também um nível intermediário de facilidades a serem instaladas.

Diversas situações práticas podem ser representadas como um PLFC2n, por exemplo: um centro de distribuição nacional dos correios (facilidades no nível 1) que deve satisfazer os centros regionais (facilidades no nível 2) para atender as agências locais (clientes); ou fábricas (facilidades no nível 1) que desejam escoar a sua produção por meio de centros de distribuição (facilidades no nível 2) para atender a demanda de lojistas (clientes) [Louzada et al., 2016].

O PLFC2n é capaz de representar casos comuns nas indústrias modernas, nas quais são definidos locais para instalação de fábricas e depósitos que devem atender a demanda de um conjunto de clientes. O fluxo de produtos flui das fábricas passando pelos depósitos até chegar ao cliente, nesta, e somente, nesta ordem (fábrica $\rightarrow$ depósito $\rightarrow$ cliente). O objetivo, em geral, consiste em minimizar o custo total de distribuição, definido pelo custo de abertura de facilidades (fábricas e depósitos) mais o custo de transporte entre fábricas e depósitos e entre depósitos e clientes.

Diferentes abordagens para o PLFC2n podem ser encontradas na literatura, como pode ser visto em Klose [2000], Tragantalerngsak et al. [2000] e Litvinchev e Ozuna [2012]. Klose [2000] aplicou uma relaxação Lagrangiana baseada em *cut-and-relax* como alternativa de resolução do problema considerando que as fábricas são fixas, ou seja, já estão alocadas previamente. Fábricas fixas também foram consideradas em Tragantalerngsak et al. [2000], que apresentaram um método exato para resolução do problema. Já Litvinchev e Ozuna [2012] propuseram o uso de relaxação Lagrangiana para resolver o problema considerando a localização tanto de fábricas quanto de depósitos. Mais detalhes sobre algumas das diferentes abordagens para o PLFC2n podem ser obtidos em Klose e Drexl [2005].

Neste trabalho foi utilizada a mesma abordagem apresentada por Litvinchev e Ozuna [2012], ou seja, a localização tanto de depósitos quanto de fábricas deve ser determinada. Essa mesma abordagem foi considerada nos trabalhos mais recentes sobre o PLFC2n, descritos brevemente a seguir.

Fernandes et al. [2014] apresentaram um Algoritmo Genético (AG) para resolver o problema. Os autores propuseram um conjunto de instâncias com diferentes características, e o desempenho do AG foi comparado com um *solver* comercial. Esse mesmo conjunto de instâncias foi utilizado por Louzada et al. [2016], que propuseram um método híbrido baseado na combinação do método *Clustering Search* (CS), para a definição das fábricas e depósitos a serem instalados, com um algoritmo exato de fluxo de redes, utilizado para definir o fluxo de produtos entre fábricas, depósitos e clientes. O CS foi capaz de encontrar soluções melhores em relação ao AG em tempos computacionais inferiores.

Um algoritmo evolutivo híbrido com aprendizado de máquina (HEA/FA) foi aplicado por

Guo et al. [2017], e os resultados obtidos foram competitivos em relação ao AG e ao CS descritos anteriormente. Por fim, Biajoli et al. [2019] propuseram a aplicação de um *Biased Random-Key Genetic Algorithm* (BRKGA) para resolução do PLFC2n, apresentando os resultados mais recentes para o problema. Os resultados obtidos foram competitivos em relação aos apresentados até então.

Neste trabalho, é proposto um método híbrido baseado na aplicação da meta-heurística *Greedy Randomized Adaptive Search Procedure* (GRASP) com um procedimento de *Local Branching* para resolução do PLFC2n. Essa combinação busca tirar vantagens das boas características do GRASP (mais especificamente a sua fase de construção) e ao mesmo tempo usar um *solver* comercial de programação linear inteira mista para a fase de refinamento. Os resultados obtidos foram comparados com os apresentados pelos trabalhos mais recentes sobre o problema, e o desempenho do método proposto foi competitivo em relação ao estado-da-arte.

O restante do artigo apresenta o problema em detalhes na Seção 2, seguida pela descrição do método proposto para sua resolução na Seção 3. Os experimentos computacionais são descritos na Seção 4 e as conclusões são apresentadas na Seção 5.

2. Descrição do Problema e Formulação Matemática

O Problema de Localização de Facilidades Capacitadas em Dois Níveis (PLFC2n) consiste em definir um conjunto de fabricas e depósitos a serem abertos, além de planejar o fluxos de produtos das fábricas para os depósitos (primeiro nível) e, em seguida, dos depósitos para os clientes (segundo nível). Considerando que, de modo geral, a indústria costuma estar sempre interessada em reduzir seus gastos, o PLFC2n tem como objetivo determinar o planejamento descrito acima, cujo custo seja o menor possível.

Assim, o objetivo geral do problema é minimizar os custos de operação, desde que seja possível satisfazer todas as demandas de clientes, e que as restrições de capacidade das fábricas e depósitos sejam respeitadas. Os custos podem ser classificados em dois tipos: custos de abertura e custos de transporte. Os custos de abertura são definidos como o custo para abrir (construir) uma fábrica ou depósito, e os custos de transporte consideram o custo para transportar certa quantidade de produto de uma fábrica para um depósito, e de um depósito para um cliente.

Do ponto de vista matemático, uma cadeia de suprimento pode ser representada por um grafo $\mathcal{G}(\mathcal{V}, \mathcal{A})$, onde $\mathcal{V}$ é o conjunto de vértices e $\mathcal{A}$ é o conjunto de arcos. Para um melhor tratamento dos vértices, particiona-se o conjunto $\mathcal{V}$ em três subconjuntos disjuntos, onde $\mathcal{I}$ é o conjunto de todas as fábricas, $\mathcal{J}$ é o conjunto de todas os depósitos e $\mathcal{K}$ é o conjunto de todos os clientes. Para cada fábrica $i \in \mathcal{I}$ é definida sua capacidade b_i, seu custo de abertura f_i e os custos c_{ij} de enviar produtos para todos os depósitos $j \in \mathcal{J}$. Analogamente, para cada depósito $j \in \mathcal{J}$ é definida sua capacidade p_j, seu custo de abertura g_j e os custos d_{jk} de enviar produtos para todos os clientes $k \in \mathcal{K}$. Para cada cliente $k \in \mathcal{K}$, define-se q_k como sendo a demanda a ser atendida. No PLFC2n o conjunto de arcos $\mathcal{A} = \mathcal{A}_1 \cup \mathcal{A}_2$ é definido como:

$$\mathcal{A}_1 = \{(a_1, a_2) | (a_1, a_2) \in \mathcal{I} \times \mathcal{J}\} \tag{1}$$

$$\mathcal{A}_2 = \{(a_1, a_2) | (a_1, a_2) \in \mathcal{J} \times \mathcal{K}\} \tag{2}$$

Define-se y_i, $i \in \mathcal{I}$, como variáveis de decisão que indicam se a fábrica i será aberta ou não. Defini-se também z_j, $j \in \mathcal{J}$, como variáveis de decisão que indicam se o depósito j será aberto ou não. Em seguida, definem-se as variáveis de decisão x_{ij}, $(i, j) \in \mathcal{A}_1$, e s_{jk}, $(j, k) \in \mathcal{A}_2$, que indicam o quanto de fluxo está sendo enviado da fábrica i para o depósito j e do depósito j para o cliente k, respectivamente.

Após as definições acima, têm-se todos os componentes necessários para representar o PLFC2n como um problema de programação linear inteira mista [Fernandes et al., 2014]:

$$\min \quad \sum_{i\in\mathcal{I}} f_i y_i + \sum_{j\in\mathcal{J}} g_j z_j + \sum_{(i,j)\in\mathcal{A}_1} c_{ij}x_{ij} + \sum_{(j,k)\in\mathcal{A}_2} d_{jk}s_{jk}$$

$$\text{s.t:} \quad \sum_{j\in\mathcal{J}|(j,k)\in\mathcal{A}_2} s_{jk} \geq q_k \qquad \forall k \in \mathcal{K} \qquad (3)$$

$$\sum_{i\in\mathcal{I}|(i,j)\in\mathcal{A}_1} x_{ij} \geq \sum_{k\in\mathcal{K}|(j,k)\in\mathcal{A}_2} s_{jk} \qquad \forall j \in \mathcal{J} \qquad (4)$$

$$\sum_{j\in\mathcal{J}|(i,j)\in\mathcal{A}_1} x_{ij} \leq b_i y_i \qquad \forall i \in \mathcal{I} \qquad (5)$$

$$\sum_{k\in\mathcal{K}|(j,k)\in\mathcal{A}_2} s_{jk} \leq p_j z_j \qquad \forall j \in \mathcal{J} \qquad (6)$$

$$x_{ij} \in \mathbb{R}^+ \qquad \forall (i,j) \in \mathcal{A}_1 \qquad (7)$$

$$s_{jk} \in \mathbb{R}^+ \qquad \forall (j,k) \in \mathcal{A}_2 \qquad (8)$$

$$y_i \in \{0,1\} \qquad \forall i \in \mathcal{I} \qquad (9)$$

$$z_j \in \{0,1\} \qquad \forall j \in \mathcal{J} \qquad (10)$$

As restrições (3) garantem que o cliente tenha sua demanda atendida. Já as restrições (4) garantem que o somatório da quantidade de produtos que chega em um depósito precisa ser maior ou igual a quantidade que este depósito envia para os clientes. As restrições (5) e (6) relacionam o fluxo enviado por uma fábrica (ou depósito) e o fato de estarem abertas ou não. Por último, as restrições (7) - (10) definem o domínio de cada uma das variáveis de decisão.

3. Método de Solução

Nesta seção são apresentadas as componentes desenvolvidas que compõem o método GRASPH, proposto neste artigo. A seção esta organizada em três subseções. Na primeira é apresentado o algoritmo construtivo utilizado para obter uma solução viável. Em seguida, na segunda subseção, tem-se uma rápida explicação do técnica de *Local Branching*, utilizada para realizar a busca local. Por último, é apresentado um panorama geral do GRASPH, combinando os elementos apresentados anteriormente.

3.1. Algoritmo Construtivo

A heurística construtiva aqui proposta pode ser divida em de quatro etapas: 1 - Escolha de um subconjunto de fábricas a serem abertas; 2 - Escolha de um subconjunto de depósitos a serem abertos; 3 - Cálculo do fluxo na rede; 4 - Pós-processamento.

Para realização das Etapa 1, tem-se um laço de repetição que escolhe fábricas aleatoriamente de uma lista restrita de candidatos, aqui chamada de CL_1, até que o somatório das capacidades das fábricas seja maior ou igual ao somatório das demandas do clientes. Para construir a CL_1, utilizou-se como função de benefício a equação a seguir:

$$h_i^1 = \frac{f_i + \frac{1}{|\mathcal{J}|}\sum_{j\in\mathcal{J}} c_{ij}}{b_i}, \forall i \in \mathcal{I} \qquad (11)$$

Uma vez definida a função de beneficio, é possível construir a CL_1. A CL_1 é construída segundo a regra a seguir:

$$CL_1 = \{i \in \mathcal{I} \setminus \mathcal{F} | h^1_{min} \leq h^1_i \leq h^1_{max} + \alpha(h^1_{min} - h^1_{max})\} \tag{12}$$

onde $h^1_{min} = \underset{i \in \mathcal{I}}{\text{argmin}}\{h^1_i\}$, $h^1_{max} = \underset{i \in \mathcal{I}}{\text{argmax}}\{h^1_i\}$, $\alpha \in [0,1]$ e $\mathcal{F}$ é o conjunto das fábricas já abertas.

Analogamente, para a Etapa 2 cria-se um laço de repetição em que a cada iteração é escolhido um depósito, até que o somatório das capacidades dos depósitos seja maior ou igual a soma das demandas dos clientes. Para isso, utiliza-se uma lista de candidatos restrita, CL_2, utilizando a seguinte função de beneficio:

$$h^2_j = \frac{f_j + \frac{1}{|\mathcal{K}|}\sum_{k \in \mathcal{K}} d_{jk}}{p_j}, \forall j \in \mathcal{J} \tag{13}$$

Usando a função de beneficio acima a CL_2 é formada utilizando a seguinte regra:

$$CL_2 = \{j \in \mathcal{J} \setminus \mathcal{D} | h^2_{min} \leq h^2_j \leq h^2_{max} + \beta(h^2_{min} - h^2_{max})\} \tag{14}$$

onde $h^2_{min} = \underset{j \in \mathcal{J}}{\text{argmin}}\{h^2_j\}$, $h^2_{max} = \underset{j \in \mathcal{J}}{\text{argmax}}\{h^2_j\}$, $\beta \in [0,1]$ e $\mathcal{D}$ e o conjunto dos depósitos já abertos.

Já a Etapa 3 consiste em fixar as fábricas e depósitos abertos nas etapas 1 e 2 no modelo matemático, descrito na Seção 2, e resolvê-lo para assim obter o fluxo e o custo total da solução. Uma vez concluída a Etapa 3, passa-se a Etapa 4, que consiste em fechar todas as fábricas e depósitos que não enviam nenhuma unidade de fluxo. Essa sequência de etapas é apresentada em detalhes no Algoritmo 1.

Algorithm 1: Construtivo

Entrada: $\mathcal{I}$,$\mathcal{J}$,$\mathcal{K}$, α, β
Início
  $\mathcal{F} \leftarrow \emptyset$;
  enquanto $\sum_{i \in \mathcal{F}} b_i < \sum_{k \in \mathcal{K}} q_k$ **faça**
    $CL_1 \leftarrow GeraListaCandidatos1(\mathcal{I}, \mathcal{F}, \alpha)$;
    $\mathcal{F} \leftarrow \mathcal{F} \cup rand(CL_1)$;
  fim
  $\mathcal{D} \leftarrow \emptyset$;
  enquanto $\sum_{j \in \mathcal{D}} p_j < \sum_{k \in \mathcal{K}} q_k$ **faça**
    $CL_2 \leftarrow GeraListaCandidatos2(\mathcal{J}, \mathcal{D}, \beta)$;
    $\mathcal{D} \leftarrow \mathcal{D} \cup rand(CL_2)$;
  fim
  $s_{best} \leftarrow CalcFluxo(\mathcal{F}, \mathcal{D})$;
  $s_{best} \leftarrow PosProcessamento(s_{best})$;
  retorna s_{best}
Fim

3.2. Local Branching

A técnica *Local Branching* (LB), introduzida por Fischetti e Lodi [2003], pode ser usada de forma que modelos de programação inteira mista possam ser usados para melhorar uma dada solução viável. Dito isso, pode-se dizer que a LB usa um *solver* de programação linear inteira mista para explorar subespaços de soluções.

Esse procedimento pode ser correlacionado a um procedimento de busca local, na qual as vizinhanças são definidas através da introdução de desigualdades lineares no modelo de programação linear inteira mista.

Especificamente para o PLFC2n, é considerada a solução $s = \langle \bar{y}, \bar{z} \rangle \in P$, em que P é o poliedro formado pelas restrições (3)-(10). A ideia geral consiste em adicionar as seguintes restrições de LB:

$$\sum_{i \in I | \bar{y}_i = 0} y_i + \sum_{i \in I | \bar{y}_i = 1} (1 - y_i) \leq \Delta_1, \tag{15}$$

$$\sum_{j \in J | \bar{z}_j = 0} z_j + \sum_{j \in J | \bar{z}_j = 1} (1 - z_j) \leq \Delta_2, \tag{16}$$

em que Δ_1 e Δ_2 recebem um valor inteiro positivo, indicando o número de variáveis y_i, $i \in \mathcal{I}$ e z_j, $j \in \mathcal{J}$, que podem ter seu valor alterado de um para zero e vice-versa.

3.3. GRASPH

A meta-heurística GRASP tem como diferencial para outros métodos a geração de soluções iniciais que se baseiam em três premissas básicas: gulosa (*Greedy*), aleatória (*Randomized*) e adaptativa (*Adaptive*) [Resende e Ribeiro, 2014]. O GRASP é uma meta-heurística *multi-start*, o que significa que a cada iteração ele executa sua componente construtiva e sua busca local. As componentes principais do GRASPH (Construtivo + *Local Branching*) podem ser observadas no Algoritmo 2:

Algorithm 2: GRASPH

Entrada: $\mathcal{I}$,$\mathcal{J}$,$\mathcal{K}$, α, β, Δ_1, Δ_2, $MaxIterGR$
Início
  para $NumIterGR$ *de* $1 \ldots MaxIterGR$ **faça**
    $s \leftarrow$ Construtivo$(\mathcal{I}, \mathcal{J}, \mathcal{K}, \alpha, \beta)$;
    $s \leftarrow$ LocalBranching$(\Delta_1, \Delta_2, s, s_{best})$;
    AtualizaSol(s, s_{best})
  fim
  retorna s_{best}
Fim

A componente construtiva do GRASP utilizada neste trabalho consiste na utilização do algoritmo construtivo apresentado na Subseção 3.1. Para compor a etapa de busca local, utiliza-se o procedimento *Local Branching*, descrito na Subseção 3.2, para refinar a solução encontrada na fase de construção. Esses procedimentos são executados sequencialmente $MaxIterGR$ vezes.

4. Experimentos Computacionais

Dois conjuntos de instâncias propostos por Fernandes et al. [2014] foram utilizados para avaliar o desempenho do método proposto. O primeiro conjunto é formado por 25 instâncias com 50 fábricas, 100 depósitos e 200 clientes. Já no segundo conjunto, também com 25 instâncias, o tamanho do problema é dobrado, ou seja, 100 fábricas, 200 depósitos e 400 clientes. Ambos os conjuntos são divididos em cinco classes de instâncias com diferentes características.

Antes de realizar os experimentos computacionais foi necessário calibrar os parâmetros do GRASPH, sendo considerados para tal fim os intervalos de valores apresentados na Tabela 1.

Tabela 1: Valores testados na calibração do GRASPH.

Parâmetro	Intervalo
α	{0,25;0,35;0,45;0,55;0,65;0,75;0,85;0,95}
β	{0,25;0,35;0,45;0,55;0,65;0,75;0,85;0,95}
Δ_1	$\{ 2;3;\left\lceil\frac{\lvert\mathcal{I}\rvert}{10}\right\rceil;\left\lceil\frac{2\lvert\mathcal{I}\rvert}{10}\right\rceil;\left\lceil\frac{3\lvert\mathcal{I}\rvert}{10}\right\rceil\}$
Δ_2	$\{ 2;3;\left\lceil\frac{\lvert\mathcal{J}\rvert}{10}\right\rceil;\left\lceil\frac{2\lvert\mathcal{J}\rvert}{10}\right\rceil;\left\lceil\frac{3\lvert\mathcal{J}\rvert}{10}\right\rceil\}$
$MaxIterGR$	{ 5,10,20,50,100 }

Para determinar a melhor combinação de parâmetros foi utilizado o pacote *irace* [López-Ibáñez et al., 2016], que determinou como melhor combinação os valores apresentados na Tabela 2.

Tabela 2: Parâmetros utilizados nos experimentos computacionais.

Parâmetro	α	β	Δ_1	Δ_2	$MaxIterGR$
Valor	0,95	0,95	$\left\lceil\frac{3\lvert\mathcal{I}\rvert}{10}\right\rceil$	$\left\lceil\frac{3\lvert\mathcal{J}\rvert}{10}\right\rceil$	5

O método proposto (GRASPH) foi codificado na linguagem XPRESS Mosel 4.8.4 e executado em um computador com um processador Intel Core i7 de 4.20GHz com 12 GB de memória RAM. O GRASPH foi aplicado 10 vezes para cada uma das 50 instâncias, assim como os demais métodos utilizados para comparação. As Tabelas 3 e 4 apresentam os resultados obtidos pelo GRASPH e pelos métodos apresentados nos trabalhos mais recentes sobre o PLFC2n. Nessas tabelas, as primeiras colunas indicam a classe da instância, o identificador e o melhor limitante inferior conhecido (LB), apresentado por Guo et al. [2017]. Já as colunas Avg(%) e Bst(%) indicam, respectivamente, o percentual do desvio das soluções médias e melhores obtidas por cada método em relação ao limitante inferior (LB). O desvio é calculado como $((sol - LB)/LB) \times 100$, em que sol é o valor da solução média ou melhor. Os melhores desvios obtidos estão destacados em negrito. Por fim, as colunas T(s) indicam o tempo médio de execução dos métodos em segundos.

Como pode ser observado na Tabela 3, o desvio médio apresentado pelo GRASPH tanto para as soluções médias foi pior em relação ao AG, CS+CPLEX e BRKGA+LS, porém melhor em relação ao HEA/FA. O mesmo pode ser observado para o desvio médio para as melhores soluções, exceto pelo AG cujas melhores soluções não são apresentadas em Fernandes et al. [2014]. O tempo de processamento do GRASPH foi o maior entre os métodos. Já na Tabela 4, é possível observar que o GRASPH apresentou resultados piores em relação ao CS+CPLEX e ao BRKGA+LS, porém melhores em relação ao AG e ao HEA/FA, e o tempo de processamento apresentado pelo GRASPH foi o menor entre os métodos.

https://proceedings.science/p/106904?lang=pt-br

Tabela 3: Resultados obtidos para o primeiro conjunto de instâncias (50 fábricas, 100 depósitos e 200 clientes).

			GA[a]		CS+CPLEX[b]			HEA/FA[c]			BRKGA+LS[d]			GRASPH		
Classe	Id	LB	Avg(%)	T(s)	Bst(%)	Avg(%)	T(s)	Bst(%)	Avg(%)	T(s)	Bst(%)	Avg(%)	T(s)	Bst(%)	Avg(%)	T(s)
1	1	721209,60	**0,13**	581,41	**0,13**	0,31	69,92	**0,13**	0,24	416,17	0,14	0,14	23,33	**0,13**	**0,13**	3,81
	2	730451,60	0,40	564,71	**0,24**	0,28	140,93	0,31	0,36	492,78	**0,24**	**0,24**	140,00	**0,24**	**0,24**	18,07
	3	731885,30	0,24	578,28	**0,22**	0,28	108,82	**0,22**	0,24	316,08	**0,22**	**0,22**	47,45	**0,22**	**0,22**	25,79
	4	721515,00	0,81	533,44	1,19	1,33	175,91	0,54	0,59	455,55	**0,50**	0,51	140,44	**0,50**	**0,50**	24,62
	5	713633,80	0,82	552,90	**0,81**	0,84	62,16	0,86	0,97	463,69	**0,81**	**0,81**	71,37	**0,81**	**0,81**	135,49
2	1	479860,20	**2,69**	317,05	**2,69**	2,71	44,54	2,74	2,82	230,84	**2,69**	**2,69**	20,34	**2,69**	**2,69**	324,54
	2	483072,20	**2,30**	316,55	**2,30**	2,42	95,77	2,34	2,41	199,11	**2,30**	2,31	73,16	**2,30**	2,34	312,10
	3	486018,50	2,14	330,70	**1,87**	**1,87**	35,55	**1,87**	1,89	167,28	**1,87**	**1,87**	141,54	1,88	1,94	500,39
	4	482374,60	2,04	312,35	**2,02**	2,03	115,23	2,07	2,12	162,13	**2,02**	2,05	71,96	**2,02**	**2,02**	309,55
	5	474803,30	3,14	276,85	**3,12**	3,23	75,96	**3,12**	**3,12**	170,57	**3,12**	**3,12**	13,09	**3,12**	**3,12**	500,33
3	1	2608800,00	**3,07**	276,45	**3,07**	**3,07**	43,61	3,14	3,30	117,09	3,11	3,11	153,30	3,22	3,30	504,70
	2	2616252,00	3,12	285,95	**3,11**	**3,11**	57,57	3,30	3,36	116,67	3,17	3,17	48,62	3,37	3,39	503,60
	3	2598277,00	3,11	271,31	**3,10**	**3,10**	64,78	3,30	3,39	161,80	**3,10**	**3,10**	88,09	3,23	3,32	500,72
	4	2612534,00	3,07	236,55	**3,06**	**3,06**	77,92	3,18	3,22	148,86	3,10	3,10	9,96	3,18	3,29	502,71
	5	2568856,00	**3,01**	242,34	**3,01**	**3,01**	37,27	3,17	3,19	158,15	3,13	3,13	48,58	3,14	3,22	502,36
4	1	525294,10	**3,14**	303,57	**3,14**	3,48	115,50	3,36	3,54	216,81	**3,14**	3,22	70,58	3,29	3,29	500,90
	2	526911,70	**2,33**	307,04	**2,43**	2,50	107,58	2,74	2,92	199,39	**2,43**	2,51	97,80	2,65	2,80	501,45
	3	532592,30	2,66	318,14	2,45	2,45	148,90	2,59	2,62	247,29	**2,33**	**2,42**	133,19	2,45	2,92	500,74
	4	529372,00	2,53	279,54	**2,36**	**2,48**	156,22	2,63	2,81	261,75	2,44	2,51	90,39	**2,36**	2,50	500,70
	5	521470,10	**3,13**	264,10	**3,13**	3,29	89,83	3,18	3,18	182,24	3,14	3,16	47,79	3,15	3,23	329,16
5	1	2743547,00	1,20	361,04	1,18	**1,19**	193,13	**1,16**	1,20	199,23	1,18	1,22	145,41	1,24	1,31	500,72
	2	2752021,00	**1,07**	344,12	**1,07**	1,10	160,30	1,11	1,16	259,72	**1,07**	1,09	43,13	1,15	1,17	500,70
	3	2737769,00	1,10	420,73	**1,09**	**1,09**	235,62	1,22	1,28	202,71	1,13	1,15	157,52	1,29	1,30	501,10
	4	2748216,00	**1,07**	300,55	**1,06**	1,11	234,59	1,10	1,13	203,32	1,10	1,10	183,09	**1,06**	**1,07**	501,01
	5	2702350,00	**1,25**	318,55	**1,23**	**1,25**	62,26	1,31	1,34	182,20	1,26	1,27	89,61	1,29	1,34	501,71
		Média	1,98	355,77	1,96	2,02	108,39	2,03	2,10	237,26	1,95	1,97	85,99	2,00	2,06	380,28

[a] Fernandes et al. [2014] - Pentium Intel 2,3GHz, 24GB de RAM
[b] Louzada et al. [2016] - Intel Core i5 2.67GHz, 4GB de RAM
[c] Guo et al. [2017] - Intel Core i7 3.6GHz, 8GB de RAM
[d] Biajoli et al. [2019] - Intel Core i5 1.7GHz, 16GB de RAM

https://proceedings.science/p/106904?lang=pt-br

Tabela 4: Resultados obtidos para o segundo conjunto de instâncias (100 fábricas, 200 depósitos e 400 clientes).

			GA[a]		CS+CPLEX[b]			HEA/FA[c]			BRKGA+LS[d]			GRASPH		
Classe	Id	LB	Avg(%)	T(s)	Bst(%)	Avg(%)	T(s)	Bst(%)	Avg(%)	T(s)	Bst(%)	Avg(%)	T(s)	Bst(%)	Avg(%)	T(s)
1	1	1475952	0,55	2784,60	0,11	0,18	400,20	0,29	0,33	1341,67	**0,10**	0,11	320,99	**0,10**	**0,10**	322,80
	2	1462736	1,01	2745,02	0,37	0,68	655,95	0,27	0,43	1514,37	**0,12**	0,13	661,34	**0,12**	**0,12**	110,81
	3	1492163	0,34	3001,83	0,70	0,91	523,17	0,23	0,48	1481,83	**0,15**	0,17	232,45	**0,15**	**0,15**	237,97
	4	1459076	0,49	2823,39	**0,22**	0,30	740,68	0,25	0,28	1122,33	**0,22**	**0,23**	344,14	**0,22**	0,28	409,95
	5	1490742	0,67	2863,24	**0,12**	0,42	409,80	0,17	0,24	1501,11	**0,12**	**0,12**	622,30	**0,12**	**0,12**	70,10
2	1	970908,5	0,89	1483,25	**0,27**	0,39	884,59	0,63	0,70	944,83	**0,27**	**0,28**	481,18	**0,27**	0,36	494,68
	2	965908,5	0,74	1455,77	**0,28**	0,46	611,62	0,47	0,56	908,30	0,29	**0,29**	553,57	**0,28**	0,34	473,55
	3	975499,7	1,42	1427,72	**0,14**	0,20	236,58	0,20	0,25	922,00	**0,14**	0,15	108,15	**0,14**	**0,14**	412,71
	4	973019,1	0,56	1444,07	**0,28**	0,38	715,60	0,56	0,59	1083,61	**0,28**	**0,29**	717,48	0,35	0,41	501,97
	5	941567	1,12	1419,02	**0,60**	0,68	427,82	0,86	0,96	1159,33	0,61	**0,61**	612,25	0,86	1,06	501,47
3	1	5213566	1,63	1355,38	**1,61**	**1,62**	1049,04	1,91	2,03	855,68	1,64	1,65	851,26	1,79	1,92	506,99
	2	5191321	1,67	1320,83	**1,65**	**1,65**	1217,64	1,96	1,99	1220,48	1,68	1,71	283,62	1,84	1,94	504,95
	3	5145991	**1,58**	1311,86	**1,58**	**1,58**	989,07	1,78	1,84	1193,12	1,63	1,63	109,46	1,77	1,84	503,33
	4	5225601	1,74	1365,90	**1,71**	**1,73**	1469,17	1,98	2,01	968,64	1,74	1,75	378,38	2,01	2,09	508,38
	5	5163182	1,72	1383,04	**1,68**	**1,69**	1404,04	1,99	2,08	942,29	1,72	1,75	495,11	2,02	2,03	503,30
4	1	1052172	0,82	1269,01	**0,60**	0,76	807,73	0,91	1,04	878,64	0,64	**0,69**	528,51	0,73	0,74	503,09
	2	1043553	0,93	1230,08	**0,68**	0,83	462,41	0,82	0,89	672,60	**0,68**	**0,74**	600,17	0,77	0,85	502,68
	3	1050683	1,88	1283,26	**0,63**	**0,80**	734,64	1,00	1,29	877,26	0,73	1,04	683,77	1,20	1,77	501,70
	4	1044571	0,96	1301,32	**0,75**	**0,85**	749,02	1,36	1,62	797,26	0,81	0,88	692,03	0,98	1,01	503,30
	5	1053869	0,64	1334,96	**0,53**	0,73	381,18	0,94	1,06	843,02	0,58	**0,61**	597,65	0,56	0,65	503,08
5	1	5486098	0,48	1551,16	**0,39**	**0,40**	873,59	0,61	0,62	967,57	**0,39**	0,41	433,52	0,43	0,54	502,63
	2	5461680	0,47	1499,94	**0,39**	**0,42**	336,05	0,41	0,44	1017,10	0,41	0,43	604,48	0,40	**0,42**	502,57
	3	5425391	0,62	1477,01	0,50	0,52	478,28	0,64	0,73	1371,96	**0,42**	**0,45**	671,41	0,59	0,59	503,50
	4	5494811	0,52	1513,78	**0,41**	**0,44**	436,70	0,58	0,58	1084,43	0,43	**0,44**	438,47	0,50	0,51	502,27
	5	5442621	0,47	1472,05	**0,39**	**0,40**	1080,57	0,43	0,47	1162,63	**0,39**	**0,40**	607,44	0,46	0,48	501,81
		Média	0,96	1684,70	0,66	0,76	723,01	0,85	0,94	1073,28	0,65	0,68	505,17	0,75	0,82	443,58

[a] Fernandes et al. [2014] - Pentium Intel 2,3GHz, 24GB de RAM
[b] Louzada et al. [2016] - Intel Core i5 2.67GHz, 4GB de RAM
[c] Guo et al. [2017] - Intel Core i7 3.6GHz, 8GB de RAM
[d] Biajoli et al. [2019] - Intel Core i5 1.7GHz, 16GB de RAM

https://proceedings.science/p/106904?lang=pt-br

Uma análise conjunta dos resultados apresentados nas Tabelas 3 e 4 permite observar a escalabilidade do método proposto, pois o tempo médio de processamento teve um aumento de apenas 16,6% para resolução das instâncias do segundo conjunto em relação àquelas do primeiro conjunto, ou seja, mesmo dobrando o tamanho das instâncias, o tempo de processamento não aumentou significativamente. Já para o AG, CS+CPLEX, HEA/FA e BRKGA+LS, o aumento foi de 373,5%, 567,0%, 352,4% e 487,5%, respectivamente.

5. Conclusões

Este trabalho apresentou um método híbrido GRASP com *Local Branching* para resolução do Problema de Localização de Facilidades Capacitadas em Dois Níveis (PLFC2n). O algoritmo construtivo foi utilizado para obter soluções iniciais de qualidade, enquanto que o *Local Branching* foi aplicado para na fase de refinamento, buscando melhorar as soluções iniciais, ou seja, esses dois métodos foram organizados na estrutura de um GRASP.

O desempenho do método proposto foi comparado diretamente com aqueles apresentados nos trabalhos mais recentes sobre o problema. Os resultados obtidos foram próximos aos melhores conhecidos, piores em alguns casos e melhores em outros. Além disso, é importante destacar a escalabilidade apresentada pelo método proposto, que foi capaz de resolver as instâncias maiores sem um aumento significativo no tempo de processamento em relação às instâncias menores.

Portanto, o método proposto demonstrou ser uma alternativa para resolução do PLFC2n compatível com os métodos que definem o atual estado-da-arte para o problema.

Referências

Biajoli, F. L., Chaves, A. A., e Lorena, L. A. N. (2019). A biased random-key genetic algorithm for the two-stage capacitated facility location problem. *Expert Systems With Applications*, 115: 418–426.

Fernandes, D. R. M., Rocha, C., Aloise, D., Ribeiro, G. M., Santos, E. M., e Silva, A. (2014). A simple and effective genetic algorithm for the two-stage capacitated facility location problem. *Computers & Industrial Engineering*, 75:200–208.

Fischetti, M. e Lodi, A. (2003). Local branching. *Mathematical Programming*, 98:23–47.

Guo, P., Cheng, W., e Wang, Y. (2017). Hybrid evolutionary algorithm with extreme machine learning fitness function evaluation for two-stage capacitated facility location problems. *Expert Systems With Applications*, 71:57–68.

Klose, A. (2000). A lagrangian relax-and-cut approach for the two-stage capacitated. *European Journal of Operational Research*, 126:185–198.

Klose, A. e Drexl, A. (2005). Facility location models for distribution system design. *European Journal of Operational Research*, 162:4–29.

Litvinchev, I. e Ozuna, E. L. (2012). Lagrangian bounds and a heuristic for the two-stage capacitated facility location problem. *International Journal of Energy Optimization and Engineering*, 1:59–71.

López-Ibáñez, M., Dubois-Lacoste, J., Cáceres, L. P., Birattari, M., e Stützle, T. (2016). The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, 3:43–58.

Louzada, R. R., Mauri, G. R., e Ribeiro, G. M. (2016). Método heurístico híbrido para resolução do problema de localização de facilidades capacitadas em dois níveis. In *Anais do XLVIII SBPO - Simpósio Brasileiro de Pesquisa Operacional*. SOBRAPO.

Resende, M. G. C. e Ribeiro, C. C. (2014). Grasp: Greedy randomized adaptive search procedures. In Burke, E. K. e Kendall, G., editors, *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques*, p. 287–312. Springer US, Boston, MA.

Tragantalerngsak, S., Holt, J., e Ronnqvist, M. (2000). An exact method for the two-echelon, single-source, capacitated facility location problem. *European Journal of Operational Research*, 123: 473–489.