ALGORTIMO BASEADO NO HARMONY SEARCH E NA REPRESENTAÇÃO PTbR PARA O PROBLEMA DE FLUXO DE CUSTO MÍNIMO COM INCERTEZAS

Rodolfo Luiz Mendonça Sales
Universidade Federal de Itajubá
Av. B P S, 1303 - Pinheirinho, Itajubá - MG, 37500-903
rodolfoluizms@unifei.edu.br

Edmilson Marmo Moreira
Universidade Federal de Itajubá
Av. B P S, 1303 - Pinheirinho, Itajubá - MG, 37500-903
edmarmo@unifei.edu.br

Anderson Tadeu de Oliveira Vicente
Universidade Federal de Itajubá
Av. B P S, 1303 - Pinheirinho, Itajubá - MG, 37500-903
ander.oliveira@ieee.org

RESUMO

Este artigo apresenta um estudo sobre o problema de fluxo de custo mínimo em Grafos com capacidades e custos Fuzzy. Descreve alguns dos fundamentos referentes à Teoria de Grafos e a Teoria de Conjuntos Fuzzy, necessários para a compreensão dos métodos abordados. Por fim, propõe dois algoritmos compostos pelo meta-heurístico *Harmony Search* em conjunto com um esquema de representação de grafos, chamado de Representação Probabilística baseada em Árvore (PTbR), a fim de verificar a aplicabilidade deste algoritmo no problema de fluxo de custo mínimo.

PALAVRAS CHAVE. Harmony Search, Representação Probabilística baseada em Árvore, Conjuntos Fuzzy.

MH – Metaheurísticas
TAG – Teoria e Algoritmos em Grafos

ABSTRACT

This article presents a study on the problem of minimum cost flow in Graphs with Fuzzy capacities and costs. It describes some of the fundamentals related to Graph Theory and Fuzzy Set Theory, necessary for the understanding of the methods covered. Finally, it is proposed two algorithms composed by the meta-heuristic Harmony Search together with a graph representation scheme, called Probabilistic tree-based representation (PTbR), in order to verify the applicability of this algorithm in the problem of minimum cost flow.

KEYWORDS. Harmony Search, Probabilistic tree-based representation, **Fuzzy Sets.**

MH – Metaheuristics
TAG – Theory and Algorithms in Graphs

1. Introdução

Uma das áreas da matemática aplicada é a Teoria dos Grafos, tendo como marco inicial o artigo de Leonhard Euler, publicado em 1736 [Euler 1736]. Esta teoria ganhou visibilidade pela quantidade de áreas de estudo que podem utilizar a estrutura de Grafos para representar os problemas existentes em suas respectivas áreas, por exemplo: Biologia, Economia, Linguística, Engenharia, Computação, dentre outras [Goldbarg e Goldbarg 2012]. Estes tipos de problemas podem ser organizados de diferentes formas, no qual cada um tem uma função objetivo diferente para ser otimizada. Um dos mais famosos é o problema de fluxo de custo mínimo (PFCM). Este tipo de problema é muito estudado justamente por suas derivações, como por exemplo, o problema de caminho mínimo, fluxo máximo, fluxo de multiprodutos e problemas de transporte [Takahashi 2004].

Existem vários algoritmos eficientes para o PFCM clássico, como o *Simplex* para redes e o *Out-of-Kilter*, dentre outros algoritmos descritos nos trabalhos de [Goldbarg e Luna 2000] e [Ahuja et al. 1993]. Contudo, usualmente, problemas reais têm parâmetros que são imprecisos, desde a própria estrutura da rede até os atributos que compõem sua estrutura, como a demanda dos nós e a capacidade dos arcos [Takahashi 2004]. Neste contexto, será discutido neste trabalho a versão *fuzzy* do PFCM, referenciado como PFCM-*fuzzy*. Existem diversos trabalhos que propõem algoritmos para resolver o PFCM-*fuzzy*, cada um com uma abordagem diferente, como em [Takahashi 2004], [Shih e Lee 1999], e dois dos trabalhos realizados por [Hernandes et al. 2008].

Dentre os trabalhos encontrados na literatura, o estudo desenvolvido em [Hernandes et al. 2008] resolve o PFCM-*fuzzy* utilizando o algoritmo *Simplex* BigM [Goldbarg e Luna 2000], com a "defuzzificação" dos custos, utilizando o primeiro índice de Yager e α-cortes nas capacidades *fuzzy*. Desta forma, cria-se um algoritmo com baixa complexidade matemática, todavia a utilização desta abordagem gera falta de precisão na obtenção da solução final. Mesmo com as perdas de informações causadas pelos métodos de "defuzzificação", a resposta encontrada por [Hernandes et al. 2008] se aproxima do valor modal apresentado por [Forbeck et al. 2008], sendo, então, uma abordagem válida.

Neste artigo será apresentada uma abordagem semelhante a de [Hernandes et al. 2008], realizando a troca do método de "defuzzificação" de Yager pela "defuzzificação" GMIR [Chen e Hsieh 1999], bem como a troca do algoritmo *Simplex* Big-M pelo *Harmony Search Algorithm*, descrito em [Mahdavi et al. 2007]. A troca dos algoritmos se dá pelo fato de que métodos heurísticos e meta-heurísticos podem ser utilizados para resolver problemas com múltiplas funções objetivos [Lu e Wang 2017] ou quando a função objetivo não é linear [Neumann et al. 2019].

2. Conceitos Preliminares

Um conjunto *fuzzy A* em *X* é caracterizado pela função de pertinência $\mu(x)$ que associa cada ponto de *X* a um número real no intervalo de [0, 1], com o valor de $\mu(x)$ em x representando o grau de pertinência de x em *A*, portanto, quanto mais a função de pertinência se aproxima da unidade, maior é o grau de pertinência de x em *A*, enquanto a função de pertinência de um conjunto ordinário recebe apenas dois valores, 0 ou 1, ou seja, se x pertence ou não ao conjunto *A*. As funções de pertinência podem ser definidas como quaisquer funções do tipo $\mu{:}X \rightarrow [0,1]$. Desta forma, cada função possui propriedades distintas e sua utilização é determinada pelo contexto do problema.

Definição 1. Um número *fuzzy* é dito triangular se sua função de pertinência é da forma:

$$\mu_a(x) = \begin{cases} 0, & se\ x < a \\ \frac{x-a}{m-a}, & se\ x \in [a, m] \\ \frac{b-x}{b-m}, & se\ x \in [m, b] \\ 0, & se\ x > b \end{cases} \qquad (1)$$

Sendo “a” e “b”, respectivamente, o limitante inferior e superior, e “m” o valor modal do número *fuzzy*. Este número pode ser escrito da seguinte forma $A(a, m, b)$ [Takahashi 2004].

Definição 2. Um número *fuzzy* é dito trapezoidal se sua função de pertinência é da forma:

$$\mu_a(x) = \begin{cases} 0, & se\ x < a \\ \frac{x-a}{m-a}, & se\ x \in [a, m] \\ 1, & se\ x \in [m, n] \\ \frac{b-x}{b-n}, & se\ x \in [n, b] \\ 0, & se\ x > b \end{cases} \qquad (2)$$

Sendo “a” e “b” a base maior do trapézio e “m” e “n” a base menor do trapézio. Este número pode ser escrito da seguinte forma $A(a, m, n, b)$ [Takahashi 2004].

Definição 3. Dado $\alpha \in [0, 1]$, um α-corte de um subconjunto *fuzzy* A é um conjunto ordinário $[A]_\alpha$, em que:

$$[A]_\alpha = \{x |\ \mu_a(x) \geq \alpha\} \qquad (3)$$

Definição 4. Sejam $\tilde{A}$ e $\tilde{B}$ dois números *fuzzy*, com $\tilde{A} \neq \tilde{B}$. O grau de possibilidade do número *fuzzy* $\tilde{A}$ ser menor que o número *fuzzy* $\tilde{B}$ é dado por:

$$Poss(A, B) = sup_{x \in X}[\min(\mu_A(x), \mu_B(x))] \qquad (4)$$

Definição 5. De acordo com Okada e Soper [Okada e Soper 2000] sejam $\tilde{a}$ e $\tilde{b}$ números *fuzzy* triangulares, de acordo com a definição 1, tal que $\tilde{a} = (a_1, m_1, b_1)$ e $\tilde{b} = (a_2, m_2, b_2)$, então $\tilde{a}$ domina $\tilde{b}$ se, e somente se, as seguintes inequações forem satisfeitas.

$$a_1 \leq a_2 \qquad (5)$$
$$b_1 \leq b_2 \qquad (6)$$
$$m_1 \leq m_2 \qquad (7)$$
$$\tilde{a} \neq \tilde{b} \qquad (8)$$

Definição 6. (Grafo *Fuzzy*) Um grafo *fuzzy* é um par $G = (V, A)$, sendo que V é um subconjunto *fuzzy* de S e A é uma relação *fuzzy* de S X S [Takahashi 2004].

Definição 7. O defuzzificador utilizado neste trabalho foi o GMIR (*Graded Mean Integration Representation*), as operações e a ordem entre um número *fuzzy* triangular A e um número ordinário k são definidas a seguir:

$$\Phi_g(A + k) = \frac{4m + a + b}{6} + k \qquad (16)$$

$$\Phi_g(A - k) = \frac{4m+a+b}{6} - k \qquad (17)$$

$$\Phi_g(kA) = k * \frac{4m + a + b}{6} \qquad (18)$$

$$k \leq A, \text{ se e somente se, } k \leq \frac{4m + a + b}{6} \qquad (19)$$

Comparações com outros índices de "defuzzificação" e propriedades do GMIR são amplamente formalizadas em [Chen e Hsieh 1999].

3 Problema do Fluxo de Custo Mínimo *Fuzzy*

O PFCM é um dos principais problemas de fluxo em redes e consiste em enviar um fluxo do vértice fornecedor ao vértice de demanda com o menor custo possível, respeitando os limites das capacidades de cada arco.

Seja uma rede $G = (V, A)$ com custo c_{ij} e fluxo x_{ij} associados ao arco $(i, j) \in A$. Cada arco tem uma capacidade mínima l_{ij} e máxima u_{ij} para a passagem de fluxo x_{ij}. O parâmetro b_i representa o somatório do fluxo que entra e sai de cada vértice. Se $b_i > 0$, então o vértice é um fornecedor, se $b_i = 0$, então este é um vértice de transbordo e se $b_i < 0$, então este é um vértice de demanda. Para este problema, o somatório dos parâmetros b_i deve ser igual a zero, ou seja, a quantidade de demanda é igual a quantidade de produto fornecido.

A modelagem do problema clássico é dada por:

$$Min\ z = \sum_{(i,j)\in A} c_{ij}x_{ij} \tag{9}$$

$$s.a. \sum_{(k,i)\in A} x_{ki} - \sum_{(i,k)\in A} x_{ik} = b_i, \forall i \in N \tag{10}$$

$$I_{ij} \leq x_{ij} \leq u_{ij}, (i,j) \in A \tag{11}$$

Como dito anteriormente, alguns algoritmos para o caso clássico são apresentados em Goldbarg e Luna [Goldbarg e Luna 2000] e Ahuja et al. [Ahuja et al. 1993]. Os trabalhos encontrados na literatura tratam o problema *fuzzy* utilizando um grafo de estrutura *crisp* (ordinária) com os parâmetros c_{ij} e u_{ij} como números *fuzzy* [Takahashi 2004]. Existem trabalhos que consideram o parâmetro b_i como um número *fuzzy*, como por exemplo em Lu e Wang [Lu e Wang 2017], entretanto nos resultados apresentados neste artigo serão apenas utilizados como números *fuzzy* apenas os custos e as capacidades dos arcos.

Alguns trabalhos que estudam o PFCM-*fuzzy* consideram que a capacidade está limitada apenas superiormente [Takahashi 2004], esta consideração foi utilizada neste trabalho. Logo, mantendo a ideia utilizada em [Takahashi 2004] e [Hernandes et al. 2008], a capacidade(u_{ij}) será um número *fuzzy* trapezoidal, enquanto o custo será um número triangular que será aplicado o GMIR para "defuzzificação".

Desta forma, a modelagem do caso *fuzzy* é dada por:

$$Min\ z = \sum_{(i,j)\in A} \tilde{c}_{ij}x_{ij} \tag{12}$$

$$s.a. \sum_{(k,i)\in A} x_{ki} - \sum_{(i,k)\in A} x_{ik} = b_i, \forall i \in N \tag{13}$$

$$0 \leq x_{ij} \leq \tilde{u}_{ij}, (i,j) \in A \tag{14}$$

4. Harmony Search Algorithm

O algoritmo utilizado neste trabalho foi o *Harmony Search Algorithm*, apresentado pela primeira vez em [Geem et al. 2001], sendo desenvolvido com o intuito de ser um novo algoritmo meta-heurístico que conseguisse ter um desempenho melhor do que os outros algoritmos existentes na época, comopor exemplo: o algoritmo genético [Geem et al. 2001]. O HS (*Harmony Search*) foi desenvolvido utilizando um fenômeno artificial do processo de improvisação, que os

músicos utilizam para achar uma harmonia que seja agradável. Os passos do algoritmo são descritos a seguir.

4.1. Inicialização do problema e dos parâmetros do algoritmo

Neste primeiro passo serão definidos a função objetivo, ou seja, a equação (9), o número de variáveis e o valor máximo e mínimo que cada variável irá receber. Neste trabalho, os valores máximos e mínimos serão definidos pelas capacidades inferiores e superiores dos arcos. Outros parâmetros necessários são o HMS (*harmony memory size*) que é o tamanho da matriz dos vetores solução; o HMCR (*Harmony memory considering rate*) que é a taxa de consideração da matriz dos vetores solução; o PAR (*pitch adjusting rate*) que é a taxa de ajuste; o bw (*bandwidth*) que é o tamanho de variação máxima para um novo vetor; e o NI que é o número de gerações ou também chamado de número de improvisações. *Harmony memory* (HM) é o local onde todos os vetores solução são guardados. Os parâmetros HMCR, PAR e bw (*bound wide*) são utilizados para improvisar o vetor solução.

4.2. Inicialização da *Harmony Memory*

Nesta etapa, a matriz HM, mostrada a seguir, é preenchida aleatoriamente.

$$\mathrm{HM} = \begin{bmatrix} x_1^1 & x_2^1 & \cdots & x_{N-1}^1 & x_N^1 \\ x_1^2 & x_2^2 & \cdots & x_{N-1}^2 & x_N^2 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ x_1^{\mathrm{HMS}-1} & x_2^{\mathrm{HMS}-1} & \cdots & x_{N-1}^{\mathrm{HMS}-1} & x_N^{\mathrm{HMS}-1} \\ x_1^{\mathrm{HMS}} & x_2^{\mathrm{HMS}} & \cdots & x_{N-1}^{\mathrm{HMS}} & x_N^{\mathrm{HMS}} \end{bmatrix}$$

Para esta pesquisa, o preenchimento da matriz foi feito de tal forma que os valores randomizados respeitassem os limites impostos pelas restrições das inequações (11). Além do preenchimento da matriz HM, calcula-se, para cada vetor, aleatoriamente formado, o valor da função de avaliação (*fitness*), a fim de futura comparação no processo de otimização.

4.3. Improvisação da nova harmonia

Como descrito em [Mahdavi et al. 2007] e [Geem et al. 2001], a improvisação da nova harmonia consiste em três regras: os vetores solução atuais da matriz HM, os ajustes realizados pelo PAR e pelo bw e a seleção randômica. O processo de formação do novo vetor pode ser visto na Listagem 1:

Algoritmo: Improvisação da nova harmonia

for j =1 to N **do**
  if rand()$\leqslant HMCR$
$x_j^{new} = x_j^r, (r \in (1,2,\cdots,HMS))$
    if rand()$\leqslant PAR$ **then**
      $x_j^{new} = x_j^{new} \pm bw \times rand$
    endif
  else $x_j^{new} = x_{jL} + rand \times (x_{jU} - x_{jL})$
  endif
endfor

Listagem 1 – Processo de improvisação [Li e Wang 2018].

Neste caso, foi utilizado o IHS (*Improved Harmony Search*), desenvolvido em [Mahdavi et al. 2007], e que tem um desempenho melhor do que o HS clássico, de acordo com [Mahdavi et al. 2007]. As alterações são dadas no cálculo do bw e do PAR, que são apresentados a seguir.

$$PAR(gn) = PAR_{min} + \frac{(PAR_{max} - PAR_{min})}{NI} * gn \quad (20)$$

$$bw(gn) = bw_{max} \exp\left(\frac{\ln(\frac{bw_{min}}{bw_{max}})}{NI} * gn\right) \quad (21)$$

Em que gn é o número da geração.

4.4. Atualizando a matriz HM

A atualização da matriz harmônica dependerá da função de avaliação (*fitness*), ou seja, se o valor de avaliação do novo vetor for melhor que o valor de avaliação do pior vetor já existente na matriz HM, caso esta condição seja satisfeita, o novo harmônico substitui o pior harmônico na matriz HM. Neste artigo, a função *fitness* é dada por:

$$fitness(\vec{x}) = f(\vec{x}) + \beta(gn) * \sum_{i=0}^{n-1} \left(\sum_{(k,i)\in A} x_{ki} - \sum_{(i,k)\in A} x_{ik} - b_i \right)^2 \quad (22)$$

O termo $f(\vec{x})$ é dado pela função objetivo, representado pela Equação 9. Já o segundo termo é formado pelo somatório dos erros quadráticos das equações de restrição (Equação 10) multiplicado por um ganho β, com o intuito de penalizar os vetores que não atenderem as restrições de igualdade dadas pelas equações da modelagem. O ganho β é uma função do número de gerações, onde inicialmente este ganho têm valores baixos e, conforme a progressão do número de gerações, o valor de β aumenta, com o intuito de penalizar as soluções que não atingem as limitações modeladas nas gerações finais.

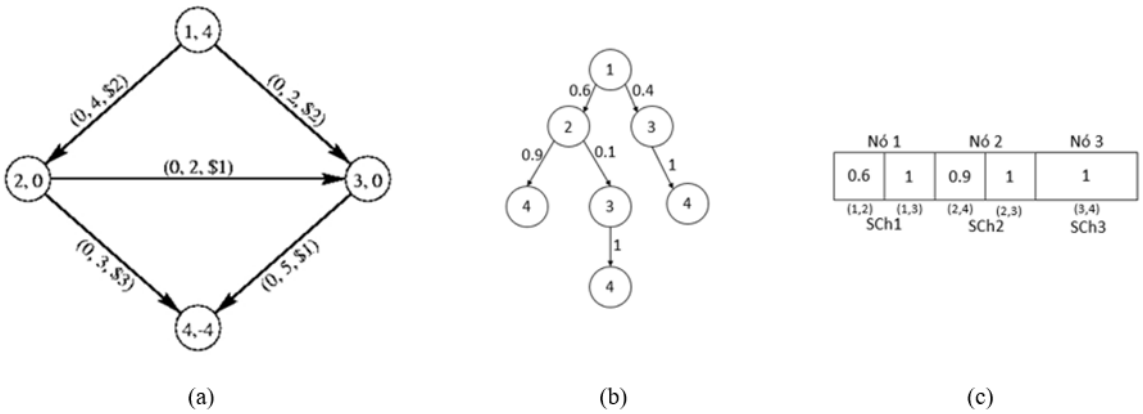
4.5. Checagem do critério de parada

O critério de parada é definido pelo número de improvisações que foi definido no começo do algoritmo. Caso o critério não seja atendido, o algoritmo voltará a repetir os passos 4.3 e 4.4.

5. Esquemas de representação

O *Harmony Search* é um algoritmo com uma estrutura genérica que visa otimizar uma função objetivo de forma combinatória, ou seja, as soluções encontradas pelo algoritmo em cada uma de suas iterações não são necessariamente viáveis para o problema proposto. Para resolver este problema utilizou-se um esquema de representação de Grafos chamado de *Probabilistic tree-based representation* (PTbR) que é uma variação da representação *Priority-based representation* (PbR), sendo a última muito utilizada para representar problemas de fluxo de custo mínimo [Neumann et al. 2019].

Para ilustrar como funciona o algoritmo proposto por [Neumann et al. 2019], utiliza-se a representação em árvore do Grafo da Figura 1, item (a). A representação e o cromossomo podem ser vistos nos itens (b) e (c) da Figura 1, respectivamente.



(a) (b) (c)

Figura 1 – Representação em árvore probabilística

No item (b), os ramos da árvore são probabilidades que o algoritmo utiliza para formar os caminhos para satisfazer a demanda. O Cromossomo (item c) é formado por outros sub-cromossomos em que o valor de cada gene representa a probabilidade do arco ser escolhido.

O algoritmo de decodificação do PTbR, proposto por [Neumann et al. 2019], funciona em duas etapas: na primeira, o caminho é construído tomando, como nó inicial, um dos nós de fornecimento. A partir deste nó, gera-se um número randômico entre 0 e 1 que será utilizado como base para tomar a decisão de qual arco o algoritmo irá seguir. Por exemplo, seja o sub-cromossomo 1 da Figura 1(c), e o número aleatório gerado com o valor de 0.7, neste caso o caminho a ser escolhido é o arco que leva ao nó 3, pois 0.7 está entre 0.6 e 1. Esta lógica é repetida até que seja possível atingir o nó de demanda.

A segunda parte do algoritmo é a de enviar um fluxo pelo caminho formado. O envio do fluxo funciona da seguinte forma, seja o caminho 1-2-3-4 gerado, define-se o menor valor para o conjunto de capacidades deste caminho, ou seja, $\min[u_{12}, u_{23}, u_{34}]$, com isto, gera-se um valor aleatório de acordo com a Equação 23. Este valor será o fluxo a ser enviado pelo caminho formado, desta forma garante-se que o fluxo é viável.

$$Fluxo = rand(1, min[\min[u_{12}, u_{23}, u_{34}], Src, |Dst|]) \quad (23)$$

Onde *Src* é valor do fornecimento e *Dst* é o valor da demanda, porém como a demanda é negativa, utiliza-se o módulo para realizar o cálculo.

Caso o primeiro fluxo fornecido não atenda a demanda da rede, ocorre outra iteração da parte 1 e 2 do algoritmo com a demanda e o fornecimento atualizados, ou seja, considera-se nesta nova iteração o fluxo do caminho que foi enviado anteriormente. Alguns exemplos das soluções geradas pelo algoritmo no grafo da Figura 1(a) podem ser vistas na Figura 2.

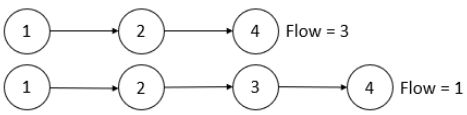


Figura 2 – Possível resultado do Algoritmo de [Neumann et al. 2019] para o Grafo da Figura 1(a).

É possível perceber que este esquema de representação tem certas limitações, como por exemplo, o somatório das demandas e dos fornecimentos tem que ser zero; só pode existir um nó de demanda, porém é possível ter vários nós de fornecimento e não pode existir mais que um arco com as mesmas extremidades. Estas limitações foram consideradas tanto no artigo de [Neumann et al. 2019], quanto neste artigo.

6. Algoritmos Propostos

Serão abordados nesta seção os algoritmos propostos que serão analisados no restante do artigo.

6.1. Algoritmo 1

Seguindo a linha de pensamento apresentado no algoritmo proposto em [Hernandes et al. 2008]. O algoritmo 1 contará com 3 passos.

Passo 1: A "defuzzificação" das capacidades utilizando a técnica de α-corte. O algoritmo irá executar os próximos passos para cada α-corte.

Passo 2: Na "defuzzificação" dos custos será aplicado o GMIR para transformar os números *fuzzy* triangulares em números ordinários.

Passo3: Aplicação do algoritmo IHS para cada α-corte e, por consequência, será obtido o vetor solução para cada α-corte.

6.2. Algoritmo 2

O algoritmo idealizado por [Neumann et al. 2019] foi chamado de PTbGA, pois utiliza, como cromossomo do algoritmo genético, a representação PTbR, este algoritmo foi idealizado a fim de resolver o problema do fluxo de custo mínimo com funções de custo não lineares e não convexas para instâncias de tamanhos pequenos, médios e grandes. No trabalho apresentado neste artigo, utilizou-se a representação PTbR nos indivíduos que compõem o *Harmony Memory do Harmony Search*, dito isto, o algoritmo 2 será chamado de PTbRIHS. A principal ideia é atribuir características inerentes ao PFCM no *Harmony Search* e verificar a diferença nas soluções obtidas. O algoritmo 2 contará com 3 passos.

Passo 1: A "defuzzificação" das capacidades utilizando a técnica de α-corte. O algoritmo irá executar os próximos passos para cada α-corte.

Passo 2: Na "defuzzificação" dos custos será aplicado o GMIR para transformar os números *fuzzy* triangulares em números ordinários.

Passo 3: Aplicação do algoritmo PTbRIHS para cada α-corte e, por consequência, será obtido o vetor solução para cada α-corte. Esta etapa difere do Algoritmo 1, pois o IHS é modificado para utilizar o PTbR em sua composição.

Uma exemplificação melhor no Passo 3 pode ser vista na Listagem 2.

```
Passo 3
    1 Entrada     Grafo e Parâmetros do algoritmo
    2 Inicializar a Harmony Memory com cada indivíduo no formato da figura 6
    3 Enquanto    a condição de parada do IHS não for satisfeita faça
    4             Improvisão da nova Harmonia
    5             Realize    o processo de decodificação (listagem 9) "N" vezes
    6             Avalie     todos os indivíduos
    7             Substituir o pior individuo pela nova Harmonia, caso a nova Harmonia tenha um fitness melhor
    8 Continue    até que a condição de parada seja satisfeita
    9 Pare
```

Listagem 2 – Passo 3

No passo 3 tem-se a adição da variável *N* que deve ser definida antes do início do algoritmo, esta variável é o número de vezes que será realizado o algoritmo de decodificação. Outra mudança realizada é na avaliação das soluções obtidas pelo algoritmo de decodificação, ou seja, a equação de *fitness* (22) terá mais um termo neste algoritmo, como pode ser visto na Equação 24.

$$Novo\ fitness = fitness + \gamma * (\max [0, x - u\ ])^2 \quad (24)$$

Onde x é o vetor solução, obtido do algoritmo de decodificação, e u é o vetor dos limitantes superiores da capacidade de cada arco. Faz-se necessário realizar esta modificação, pois diferente do IHS padrão, o vetor solução não será proveniente da improvisação da nova Harmonia (Listagem 1), mas sim do algoritmo de decodificação [Neumann et al. 2019], portanto, é provável que alguns fluxos gerados ultrapassem os limitantes superiores. Na análise realizada, foi utilizado o valor 20000 para γ, ou seja, garante-se uma alta penalização desde a primeira geração.

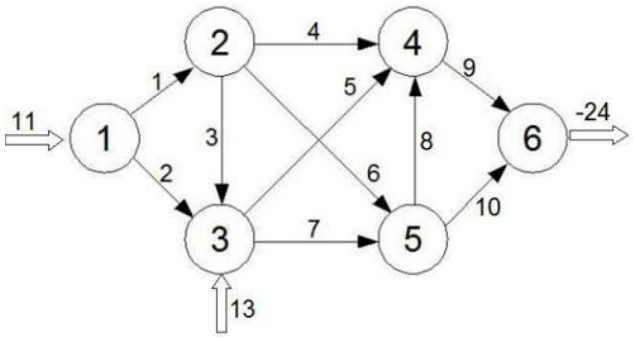
7. Análise

As análises foram realizadas na linguagem Python, utilizando como IDE o PyCharm em uma máquina com o sistema operacional Ubuntu 20.04 com um processador I5-5200U CPU 2.20GHz. Os parâmetros utilizados no *Harmony Search* (Algoritmo 1) e no PTbIHS (Algoritmo 2), durante as análises, podem ser vistos na Tabela 1.

Tabela 1 – Parâmetros utilizados nos algoritmos propostos

Algoritmo	PARmin	PARmax	BWmin	BWmax	Número de gerações	HMS	HMCR	Número de iterações	N de decodificação
IHS	0.01	0.99	0.000001	$\frac{u_{ij}}{2}$	20000	5	0.99	30	--
PTbRIHS	0.01	0.99	0.000001	0.05	300	5	0.99	30	20

Para efeito de análise, primeiro comparou-se os resultados obtidos no trabalho em [Hernandes et al. 2008], utilizando o grafo da Figura 3 com seus respectivos custos e capacidades *fuzzy*.



Arco	Custo	Capacidade
1	(3,6,8)	(0,0,12,14)
2	(1,3,8)	(0,0,13,14)
3	(3,6,8)	(0,0,8,10)
4	(5,6,8)	(0,0,9,13)
5	(7,8,10)	(0,0,6,18)
6	(8,10,12)	(0,0,15,18)
7	(5,7,8)	(0,0,12,17)
8	(9,11,12)	(0,0,17,20)
9	(7,10,12)	(0,0,18,20)
10	(10,12,14)	(0,0,15,16)

Figura 3 – Grafo estudado em [Hernandes et al. 2008]

A primeira análise foi feita com o primeiro índice de Yager como "defuzzificador" [Hernandes et al. 2008]. Na Tabela 2 estão dispostas as informações estatísticas produzidas, respectivamente, pelo Algoritmo 1 e Algoritmo 2 em cada uma de suas iterações.

Tabela 2 – Descrição das soluções encontradas pelo Algoritmo 1 (a) e o Algoritmo 2 (b) com o "defuzzificador" de Yager.

Alfa corte	Pior	Melhor	Média	Desvio Padrão
0.5	484.38	476.47	479.65	2.01
0.6	483.80	476.22	479.50	2.09
0.7	482.95	476.26	479.12	1.91
0.8	483.01	476.54	479.09	1.96
0.9	482.77	477.73	479.97	1.58
1	481.91	478.93	479.92	1.02

(a)

Alfa corte	Pior	Melhor	Média	Desvio Padrão
0.5	484.00	475.67	478.52	3.18
0.6	488.67	477.00	480.04	2.89
0.7	501.00	476.67	480.08	5.24
0.8	584.00	477.67	492.81	27.33
0.9	567.33	478.33	493.52	18.51
1	591.67	479.00	495.76	24.95

(b)

É possível perceber-que o Algoritmo 1 tem uma dispersão, nos valores de solução, menor do que o Algoritmo 2, entretanto para os α-cortes de 0.5 e 1 o Algoritmo 2 obteve como sua melhor resposta a mesma solução encontrada pelo Simplex, como pode ser visto na Tabela 3, que compara as respostas produzidas pelo algoritmo de [Hernandes et al. 2008] com a melhor resposta dos Algoritmos 1 e 2.

Tabela 3 – Fluxos obtidos para o Grafo da Figura 3 utilizando o primeiro índice de Yager.

	α = 0.5			α = 0.6			α = 0.7			α = 0.8			α = 0.9			α = 1.0		
Arcos	IHS	PTbRIHS	Simplex	IHS	PTbRIHS	Simplex	IHS	PTbRIHS	Simplex	IHS	PTbRIHS	Simplex	IHS	PTbRIHS	Simplex	IHS	PTbRIHS	Simplex
1	8.41	11	11	9.57	9	10.6	9.84	10	10.2	9.81	9	9.8	9.41	9	9.4	9.01	9	9
2	2.59	0	0	1.43	2	0.4	1.16	1	0.8	1.19	2	1.2	1.59	2	1.6	1.99	2	2
3	0.00	0	0	0.00	0	0	0.00	0	0	0.00	0	0	0.00	0	0	0.00	0	0
4	8.40	11	11	9.57	9	10.6	9.84	10	10.2	9.80	9	9.8	9.40	9	9.4	9.00	9	9
5	10.60	8	8	9.23	9	8.2	8.76	8	8.4	8.40	8	8.4	7.20	7	7.2	6.00	6	6
6	0.00	0	0	0.00	0	0	0.00	0	0	0.01	0	0	0.01	0	0	0.01	0	0
7	4.99	5	5	5.19	6	5.2	5.40	6	5.4	5.79	7	5.8	7.39	8	7.4	8.99	9	9
8	0.00	0	0	0.00	0	0	0.00	0	0	0.00	0	0	0.00	0	0	0.00	0	0
9	19.00	19	19	18.80	18	18.8	18.60	18	18.6	18.20	17	18.2	16.60	16	16.6	15.01	15	15
10	5.00	5	5	5.20	6	5.2	5.40	6	5.4	5.79	7	5.8	7.39	8	7.4	8.99	9	9
Custo Final	476.47	475.67	475.6667	476.22	477.00	475.9333	476.26	476.67	476.2	476.54	477.67	476.6	477.73	478.33	477.8	478.93	479.00	479

De forma geral, a resposta do Algoritmo 1 se aproxima do resultado obtido pelo Simplex. Enquanto o Algoritmo 2 (PTbRIHS) obtém os mesmos resultados do Simplex para os α-cortes de 0.5 e 1, percebe-se que a representação PTbR [Neumann et al. 2019] obtêm soluções de fluxos com valores inteiros, ou seja, diferente do Algoritmo 1, o Algoritmo 2 resolve o problema considerando fluxos inteiros, tal fato explica o motivo pelo qual os α-cortes de 0.6, 0.7, 0.8 e 0.9 não tiveram como melhor resultado a mesma resposta obtida pelo algoritmo de [Hernandes et al. 2008].

Realizando a mesma análise, porém, desta vez, utilizando o "defuzzificador" GMIR, obtêm-se os resultados apresentados nas Tabelas 4.

Tabela 4 – Descrição das soluções encontradas pelo Algoritmo 1 (a) e o Algoritmo 2 (b) com o "defuzzificador" GMIR

Alfa corte	Pior	Melhor	Média	Desvio Padrão
0.5	484.52	477.00	480.26	2.08
0.6	483.68	477.48	480.13	1.94
0.7	484.19	478.02	480.67	1.89
0.8	483.71	478.54	480.86	1.47
0.9	483.64	479.73	481.44	1.07
1	482.98	480.92	481.61	0.69

(a)

Alfa corte	Pior	Melhor	Média	Desvio Padrão
0.5	506.50	477.00	480.10	5.10
0.6	505.17	478.17	481.55	5.19
0.7	539.67	478.50	486.36	13.05
0.8	535.33	479.33	490.85	14.02
0.9	566.17	480.17	490.62	17.28
1	636.00	481.00	504.85	38.62

(b)

A mesma análise das Tabelas 2(a) e 2(b) pode ser feita para as Tabelas 4(a) e 4(b), portanto, resta a comparação realizada na Tabela 3, porém, desta vez, para o "defuzzificador" GMIR. A comparação pode ser vista na Tabela 5.

Tabela 5 – Fluxos obtidos para o Grafo da Figura 3 utilizando o GMIR.

	α = 0.5			α = 0.6			α = 0.7			α = 0.8			α = 0.9			α = 1.0		
Arcos	IHS	PTbRIHS	Simplex	IHS	PTbRIHS	Simplex	IHS	PTbRIHS	Simplex	IHS	PTbRIHS	Simplex	IHS	PTbRIHS	Simplex	IHS	PTbRIHS	Simplex
1	6.93	7	7	8.13	8	8	9.25	9	9	9.81	9	9.8	9.41	9	9.4	9.01	9	9
2	4.07	4	4	2.87	3	3	1.74	2	2	1.19	2	1.2	1.59	2	1.6	1.99	2	2
3	0.00	0	0	0.00	0	0	0.00	0	0	0.00	0	0	0.00	0	0	0.00	0	0
4	6.93	7	7	8.13	8	8	9.25	9	9	9.80	9	9.8	9.40	9	9.4	9.00	9	9
5	12.00	12	12	10.67	10	10.8	9.35	9	9.6	8.40	8	8.4	7.20	7	7.2	6.00	6	6
6	0.00	0	0	0.00	0	0	0.00	0	0	0.01	0	0	0.01	0	0	0.01	0	0
7	5.07	5	5	5.19	6	5.2	5.39	6	5.4	5.79	7	5.8	7.39	8	7.4	8.99	9	9
8	0.00	0	0	0.00	0	0	0.00	0	0	0.00	0	0	0.00	0	0	0.00	0	0
9	18.93	19	19	18.80	18	18.8	18.60	18	18.6	18.20	17	18.2	16.60	16	16.6	15.00	15	15
10	5.07	5	5	5.20	6	5.2	5.40	6	5.4	5.79	7	5.8	7.39	8	7.4	8.99	9	9
Custo Final	477.00	477.00	477	477.48	478.17	477.5	478.02	478.50	478	478.54	479.33	478.6	479.73	480.17	479.8	480.92	481.00	481

Os algoritmos se comportam da mesma forma independentemente do "defuzzificador" utilizado, considerando, obviamente, o grafo da Figura 3 e as condições utilizadas neste trabalho. Portanto, o que pode ser avaliado é que a mudança de "defuzzificador" altera o valor da solução final, sendo que os valores obtidos pela utilização do GMIR são maiores do que aqueles obtidos com a utilização do primeiro índice de Yager.

Percebe-se que as respostas de ambos os "defuzzificadores" se aproximam do valor modal encontrado por [Hernandes et al. 2008] utilizando o algoritmo de [Forbeck et al. 2008], que pode ser visto na Tabela 6, para o mesmo grafo com os mesmos parâmetros.

Tabela 6 – Resultado obtido por [Forbeck et al. 2008]

Arco	Custo	Capacidade	Fluxo
1	(3,6,8)	(0,0,12,14)	11
2	(1,3,8)	(0,0,13,14)	0
3	(3,6,8)	(0,0,8,10)	0
4	(5,6,8)	(0,0,9,13)	11
5	(7,8,10)	(0,0,6,18)	13
6	(8,10,12)	(0,0,15,18)	0
7	(5,7,8)	(0,0,12,17)	0
8	(9,11,12)	(0,0,17,20)	0
9	(7,10,12)	(0,0,18,20)	24
10	(10,12,14)	(0,0,15,16)	0
Custo Total Final			(395, 500, 595)

No caso deste trabalho, os resultados obtidos pelo GMIR são mais próximos do valor modal do que os resultados obtidos pelo primeiro índice de Yager, porém a real vantagem do GMIR é a possibilidade de trabalhar com números *fuzzy* do formato $A = (c, a, b; w)$, fazendo com que possa ser adicionado um nível de confiança "w" perante o número *fuzzy*. Isto significa que um número *fuzzy* [90, 100, 110] pode ter uma confiança de 90%, portanto, o número *fuzzy* seria escrito como [90, 100, 110; 0.9], desta forma é possível aumentar as possibilidades de representar diferentes situações.

8. Conclusão

Neste artigo foi proposto dois algoritmos para o PFCM-fuzzy com números fuzzy nos custos e nas capacidade. Os algoritmos propostos apresentam uma adaptação do *Harmony Search Algorithm*, ou seja, um meta-heurístico, em conjunto com uma representação baseada em árvore probabilística a fim de melhorar o desempenho do meta-heurístico. Além disso, foi analisado qual seria o impacto na resposta quando se utiliza dois "defuzzificadores" diferentes.

Verificou-se, portanto, que os algoritmos propostos aproximam-se da resposta encontrada na Tabela 6, existindo uma pequena diferença na alteração dos "defuzzificadores e obtendo um maior desvio padrão nas iterações do Algoritmo 2.

Como trabalho futuro, pretende-se adaptar este problema a redes que possuam parâmetros de maiores dimensões. Outra proposta é realizar a modelagem de sistemas reais, como por exemplo malhas rodoviárias, e aplicar o algoritmo proposto.

Referências

Ahuja, R., Magnati, T. e Orlin, J. (1993). Network flows: theory, algorithms, and applications, Prentice-Hall.

Chen, S. H. e Hsieh, C. H. (1999). Graded Mean Integration Representation of Generalized Fuzzy Number, *Journal of the Chinese Fuzzy System Association*, vol. 5, no.2, pp.1-7.

Euler, L. (1736). Solutio problematis ad geometriam situs pertinentis. Comment. Acad. Sci. U. Petrop8, 128-40.

Geem, Z. W., Kim, J. H. e Loganathan, G. V. (2001). A new heuristic optimization algorithm: harmony search, Simulation 76 (2) 60–68.

Forbeck, F. R., Hernandes, F. E. e Katayama, J. P. M. K. (2008). Um algoritmo para o problema de fluxo de custo mínimo com incertezas, *Anais do XL Simpósio Brasileiro de Pesquisa Operacional*.

Goldbarg, M. C. e Goldbarg, E. (2012). Grafos: conceitos, algoritmos e aplicações, 1a. ed. Rio de Janeiro: Elsevier.

Goldbarg, M. C. e Luna, H. P. L. (2000). Otimização Combinatória e Programação Linear, Editora Campus, Rio de Janeiro.

Hernandes, F., Katayama, J. P. M. K. e Forbeck, F. R. (2008). Um algoritmo baseado no método Simplex para redes aplicado no problema de fluxo de custo mínimo com incertezas, *Hífen Uruguaiana. Online*, v. 32, p. 59-66.

Li, G. e Wang, H. (wa). Improved harmony search algorithm for global optimization, 2018 *Chinese Control And Decision Conference (CCDC)*, Shenyang, pp. 864-867.

Lu, S. e Wang, X. (2017). A fuzzy Bi-objective multi-commodity minimum cost flow-based perishable food transportation problem and its harmony search algorithm, 2017 *IEEE 2nd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*, Chengdu, pp. 725-728.

Mahdavi, M., Fesanghary, M. e Damangir, E. (2007). An improved harmony search algorithm for solving optimization problems, *Applied Mathematics and Computation*, vol.188, no.2, pp.1567–1579.

Neumann, F., Ozlen, M., Li, X. e Ghasemishabankareh, B. (2019). Probabilistic tree-based representation for solving minimum cost integer flow problems with nonlinear non-convex cost functions. Applied Soft Computing Journal, Melbourne, p. 1-14. Disponível em: `https://doi.org/10.1016/j.asoc.2019.105951` .Acesso em: 1 out. 2020.

Okada, S. e Soper, T. (2000). A Shortest Path Problem on a Network with Fuzzy Arc Lengths. *Fuzzy Sets and Systems*, 109, 129-140.

Shih, H. S. e Lee, E. S (1999). Fuzzy Multi-Level Minimum Cost Flow Problems, *Fuzzy Sets and Systems*, 107, 159-17.

Takahashi, M.T.(2004) Contribuições ao Estudo de Grafos Fuzzy: Teoria e Algoritmos, *Tese de doutorado*, Faculdade de Engenharia Elétrica e de Computação, Universidade Estadual de Campinas.