UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Câmpus de Presidente Prudente

Beatriz de Barros Montini

ALGORITMO GENÉTICO PARA PROBLEMA GENERALIZADO DE ATRIBUIÇÃO

PRESIDENTE PRUDENTE
2021/2022

BEATRIZ DE BARROS MONTINI

ALGORITMO GENÉTICO PARA PROBLEMA GENERALIZADO DE ATRIBUIÇÃO

Relatório Final apresentado para aproveitamento na disciplina “Trabalho de Conclusão de Curso” da Graduação em Estatística da FCT/Unesp.

Orientadora: Prof. Dra. Silvely Nogueira de Almeida Salomão Néia

PRESIDENTE PRUDENTE

2021/2022

M792a

Montini, Beatriz de Barros

Algoritmo Genético para Problema Generalizado de Atribuição / Beatriz de Barros Montini. -- Presidente Prudente,2022

49 p. : il., tabs.

Trabalho de conclusão de curso (Bacharelado - Estatística) - Universidade Estadual Paulista (Unesp), Faculdade de Ciências e Tecnologia, Presidente Prudente

Orientadora: Silvely Nogueira de Almeida Salomão Néia

1. Algoritmo Genético. 2. Problema Generalizado de Atribuição. I. Título.

Sistema de geração automática de fichas catalográficas da Unesp. Biblioteca da Faculdade de Ciências e Tecnologia, Presidente Prudente. Dados fornecidos pelo autor(a).

Essa ficha não pode ser modificada.

TERMO DE APROVAÇÃO

Beatriz de Barros Montini

ALGORITMO GENÉTICO PARA PROBLEMA GENERALIZADO DE ATRIBUIÇÃO

Relatório de Final de Trabalho de Conclusão de Curso aprovado como requisito para obtenção de créditos na disciplina Trabalho de Conclusão do curso de graduação em Estatística da Faculdade de Ciências e Tecnologia da Unesp, pela seguinte banca examinadora:

Orientador: _____

Prof. Dra. Silvely Nogueira de Almeida Salomão Néia

Departamento de Estatística

_____

Prof. Dr. EDILSON FERREIRA FLORES

Departamento de Estatística

_____

Prof. Dr. KLAUS SCHLUNZEN JÚNIOR

Departamento de Estatística

Presidente Prudente, 26 de março de 2022.

RESUMO

Este trabalho aborda o Problema Generalizado de Atribuição. Visto que é um problema clássico de otimização, cujo objetivo é minimizar os custos de atribuir n tarefas a m agentes, a resolução do problema a ser estudada será pelo Algoritmo Genético. O desenvolvimento deste, consiste em construir soluções através do princípio da seleção natural e sobre os trabalhos de Mendel sobre a genética. A teoria da evolução natural consiste na ideia onde o mais apto sobreviva e este se reproduza, resultando na obtenção de uma nova geração. A resolução será testada nas instâncias de Yagiura (2021) e a implementação foi utilizando o software R, obtendo a melhor solução para as instâncias testadas.

Palavras-chaves: Problema Generalizado de Atribuição, Problema de Otimização e Algoritmo Genético.

ABSTRACT

This work addresses the Generalized Attribution Problem. Since it is a classical optimization problem, whose objective is to minimize the costs of assigning n tasks to m agents, the solution of the problem to be studied will be by the Genetic Algorithm. The development of this consists of building solutions through the principle of natural selection and on Mendel's work on genetics. The theory of natural evolution consists of the idea where the fittest survives and it reproduces, resulting in a new generation. The resolution will be tested in the instances of Yagiura (2021) and the implementation was using the R software, obtaining the best solution for the tested instances.

Keywords: Generalized Assignment Problem, Optimization Problem and Genetic Algorithm.

LISTA DE FIGURAS

LISTA DE SIGLAS

AG- Algoritmo Genético

B&B- Branch-and-Bound

B&P- Branch-and-Price

PCC- Problema de Cobertura dos Conjuntos

PGA- Problema Generalizado de Atribuição

PMR- Problema Mestre Restrito

PPC- Problema Particionamento dos Conjuntos

SUMÁRIO

1. Introdução

A Pesquisa Operacional desenvolve métodos quantitativos para a resolução de problemas de otimização de processos (GOLDBARG e LUNA, 2005), auxiliando na tomada de decisão do gerenciamento de uma organização. Em todas as empresas tomar decisões é de suma importância, as quais implicarão em lucro ou prejuízo. Um tipo de problema presente nas empresas é o de atribuição, como por exemplo determinar como se dará a divisão de tarefas, e/ou a distribuição de horários.

O Problema Generalizado de Atribuição é um problema clássico de otimização combinatória que consiste em, dadas n tarefas e m agentes, determinar qual, dentre todas as combinações de atribuições possíveis, oferece menor custo, ou seja, tem seu custo de atribuição total minimizado. Este é um problema de grande importância prática podendo ser encontrado em empresas que precisam distribuir tarefas diárias, cujo objetivo é otimizar recursos humanos e materiais, até tarefas mais complexas como a distribuição de processadores de um sistema de computação paralela (BOKHARI, 1987) e a distribuição de pacientes em voos médicos (RULAND, 1999) envolvendo prioridades.

Neste trabalho, será estudado a geração de soluções para o Problema Generalizado de Atribuição através do Algoritmo Genético.

O Algoritmo Genético é um método evolutivo de otimização trazida por John Holland em 1975. Esse método tem como referência o princípio da seleção natural e com base nos trabalhos de Mendel sobre a genética. A evolução natural consiste em que o mais apto sobreviva e se reproduza, assim obtém-se uma nova geração. O princípio de Mendel diz que um indivíduo de uma população pode ser formado por um ou mais cromossomos.

Este trabalho tem como objetivo principal realizar um estudo sobre o Problema Generalizado de Atribuição e sua resolução usando a geração de soluções através do algoritmo genético, abordando o método de geração de colunas. Também realizar uma implementação do Algoritmo Genético para resolver o Problema Generalizado de Atribuição usando instâncias do tipo A de

Yagiura(2021). As quais são dados usados por pesquisadores para implementações de algoritmo para solução de problemas.

O estudo sobre Pesquisa Operacional é muito útil para auxiliar tomadas de decisões, principalmente, em grandes organizações empresariais. A meta é minimizar custos e maximizar lucros, ou seja, essa técnica assessora a decisão, com grande probabilidade de ser a escolha certeira.

Analisando que o Problema Generalizado de Atribuição abrange vários casos práticos, como por exemplo planejamento de tarefas do telescópio espacial, ROSAT (NOWAKOVSKI *et al*. At 1999) e atribuição de frequência em redes de comunicação de celulares (FISCHETTI *et al*., 2000; WANG e GU, 2004), consequentemente, é muito utilizado. Com isso, vemos muitos casos desses problemas práticos e até mesmo pode ser considerado um subproblema de outra questão, por isso tem diversos métodos e algoritmos para resolver o problema.

Desse modo é de suma importância entender o Problema Generalizado de Atribuição para resolver da melhor maneira. E como o Algoritmo Genético tem obtido ótimo resultado em problema de otimização (GEN e CHENG, 1997). Devido ao fato de o Algoritmo Genético ser mais abstrato e geral que os outros métodos usados, ele é considerado mais flexível e com isso alcança maior variedade de problemas de otimização (ASLLANI e LARI, 2007).

A pesquisa será apresentada nos seguintes capítulos: no primeiro, a introdução ao tema, no segundo, uma apresentação de uma revisão expondo um breve estado da arte do Problema Generalizado de Atribuição. Já no terceiro capítulo será apresentado uma teoria mais aprofundada sobre o algoritmo genético. A parte experimental, em que serão mostrados os resultados obtidos com suas respectivas análises, estão no quarto capítulo e, a conclusão final com considerações sobre o trabalho realizado e possibilidades de trabalhos futuros estão no último capítulo.

2. O Problema Generalizado de Atribuição

De forma que já foi mencionado o Problema Generalizado de Atribuição é um problema clássico de otimização. Existe em diversas áreas práticas e há vários métodos para solucionar este problema.

A solução do Problema Generalizado de Atribuição consiste em atribuir um conjunto de máquinas para atender um conjunto de tarefas a um custo mínimo, sem sobrecarregar as máquinas além da sua capacidade (MARTELLO e TOTH, 1981). Não há um método que resolva o problema em tempo polinomial e é considerado NP-Difícil, por ter que explorar todas as combinações possíveis que respeitem as restrições do problema, exigindo muito esforço computacional.

2.1 Breve revisão bibliográfica

Para esse trabalho foram selecionados alguns artigos a partir do ano de 2017 até neste momento.

Observa-se que o tema estudado encontra diversas áreas práticas como a logística de entrada na indústria da cana-de-açúcar. Para a cana-de-açúcar entrar na indústria necessita passar por pelo menos três processos (cultivo, colheita e transporte). Muitos dos agricultores têm dificuldade de administrar esses processos. A fim de facilitar o gerenciamento desses procedimentos foi realizado um estudo utilizando o Problema Generalizado de Atribuição pela Unidade de Pesquisa em Modelagem de Sistemas para a Indústria, Departamento de Engenharia Industrial, Faculdade de Engenharia, Khon Kaen Universidade, Khon Kaen, Tailândia feita por Phattaraphorn Phudphad, Kanchana Sethanan, e Thitipong Jamrus (2018).

Inclusive, há estudos sobre novos algoritmos como o estudo de algoritmo de fluxo de rede para resolver o Problema Generalizado de Atribuição. Aqui um trabalho pode ser atribuído a muitos, e diferentes, agentes e um agente pode

realizar várias, e diferentes, trabalhos. Pesquisa realizada por Yongwen Hu, and Qunpo Liu (2021)

Estudos mais recentes apontam que um novo algoritmo para o problema de atribuição generalizada de multi-robôs com consumo de recursos estocásticos. No problema apresentado cada robô tem limitações como, por exemplo, a vida útil da bateria. O objetivo do problema é encontrar uma atribuição dos robôs para tarefas que maximizem a recompensa da equipe de robôs. E ainda é apresentado resultados de simulação para demonstrar que nosso algoritmo é escalável com o número de robôs e tarefas. Estudo feito por F. Yang e N. Chakraborty (2021).

2.2 Modelagem Matemática

De forma geral, temos dois conjuntos de dados I e J, sendo uma atribuição de elementos $i \in I$ a elementos $j \in J$. Dessa maneira o objetivo da função é minimizar o custo dessa atribuição.

Considerando a variável de decisão dada por:

$$x_{ij} = \begin{cases} 1 \text{ se a tarefa } j \text{ for atribuída ao agente } i \\ 0, \text{caso contrário} \end{cases} \quad (1)$$

para $\mathrm{i} = 1,2, \ldots, \mathrm{m},$

$\mathrm{j} = 1,2, \ldots, \mathrm{n}$.

Observe que ao considerar n=m, não há perda de generalidade, pois basta inserir tarefas ou agentes fantasmas para igualar n a m.

O modelo matemático do problema generalizado de atribuição é dado por:

$$\min z = \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij} \quad (2)$$

Sujeito a:

$$\sum_{i=1}^{m} x_{ij} = 1, \text{ com } j = 1,2, \ldots, n \quad (3)$$

$\sum_{j=1}^{n} r_{ij}x_{ij} \leq R_i$, com $i = 1,2,\ldots,m$ (4)

$x_{ij} \in \{0,1\}$ qualquer que sejam i e j. (5)

A equação (2) é a restrição que evita a possibilidade de a tarefa j ser atribuída a mais de um agente i. A equação (3) corresponde às restrições que garantem que a capacidade R_i do agente i seja respeitada e a última restrição faz com que a variável de decisão assuma somente valores 0 ou 1. Esta formulação foi usada por Cattrysse e Van Wassenhove (1994) para desenvolverum algoritmo heurístico para o problema.

O problema de atribuição em alguns casos pode ser fácil de trabalhar como é o problema linear de atribuição, cujo número de agente será igual ao número de tarefas. O Problema Linear de Atribuição, tem um caso especial de otimização conhecido como o Problema de Transporte. Com isso o Problema de Transporte pode ser resolvido por métodos polinomiais e outros diversos métodos (KUHN, 1955; WRIGHT, 1990)

2.3 Decomposição do Problema Generalizado de Atribuição

Inicialmente, temos que decompor o Problema Generalizado de Atribuição para conseguir através do método de geração de colunas dar início ao Algoritmo Genético gerando a população inicial.

Então para gerar colunas para o Problema Generalizado de Atribuição necessita-se de uma reformulação do problema, como por exemplo o Problema de Cobertura de Conjuntos.

O Problema de Cobertura de Conjuntos é achar o menor subconjunto de um conjunto principal cuja união é igual ao conjunto principal. Vamos supor um subconjunto representado por vértices a direita e o conjunto principal vértices a esquerda e as arestas são representadas a inclusão de elementos em subconjuntos. O objetivo é encontrar o subconjunto com o mínimo de vértices direito que cubra todos os vértices esquerdo.

Temos o Problema de Cobertura de Conjuntos formulado como:

$$\sum_{i \in M} \sum_{j \in N} cij\ xij \tag{6}$$

Sujeito a:

$$\sum_{j \in N} rij\ xij \leq bi\ \ \forall\ i\ \in M \tag{7}$$

$$\sum_{i \in M} xij = 1,\ \ \forall\ j\ \in N \tag{8}$$

$$xij \in \{0,1\}, \forall\ i\ \in M, j \in N \tag{9}$$

Para gerar colunas no Problema Generalizado de Atribuição usando como base a reformulação do Problema de Cobertura dos Conjuntos. Temos que, $Ki = \{x^i_{1,\ldots,\ ki}\ x^i\}$ os conjuntos de atribuições que podem ser viáveis para máquina i, onde $x^i_{1k,\ldots,\ nk}\ x^i\}$ é uma solução viável e segue as restrições do Problema de Cobertura dos Conjuntos. Sendo assim, $y^i_k, i \in M$ e $k \in Ki$, uma variável binária (zero ou um) que está associado a atribuição x^i_kcom a máquina i.

$$y^{\ i}_k \begin{cases} 1, se\ atribuição\ x^i_k\ está\ associada\ à\ máquina\ i \\ 0, caso\ contrário \end{cases} \tag{10}$$

Então o Problema Generalizado de Atribuição pode ser reformulado como o seguinte Problema de Cobertura de Conjuntos:

$$\sum_{i=1}^{m} \sum_{k=1}^{ki} \left(\sum_{j=1}^{n} cij\ x^i_{jk}\right) y^i_k \tag{11}$$

Sujeito a:

$$\sum_{i=1}^{m} \sum_{k=1}^{ki} x^i_{jk}\ y^i_k = 1, \quad j \in N \tag{12}$$

$$\sum_{i \in M} y^i_k \leq 1,\ \ i\ \in M \tag{13}$$

$y_k^i \in \{0,1\},\ i\ \in M$ (14)

Essa formulação é usada para desenvolver um algoritmo para o Problema Generalizado de Atribuição. Agora com reformulação pode ser resolvido a partir da programação linear.

$\sum_{i=1}^{m} \sum_{k=1}^{ki} (\sum_{j=1}^{n} cij\ x_{jk}^{i}\ )\ y_{k}^{i}$ (15)

Sujeito a:

$\sum_{i=1}^{m} \sum_{k=1}^{ki} x_{jk}^{i}\ y_k^i = 1, \quad j \in N$ (16)

$\sum_{i\in M}\ y_k^i \leq 1,\ i\ \in M$ (17)

$y_k^i \in [0,1],\ i\ \in M$ (18)

Então com essa nova formulação matemática conseguimos resolver como um problema de programação linear. E consequentemente as equações (15)-(18) são conhecidas como Problema Mestre Restrito.

A princípio após definir um conjunto inicial de colunas, o Problema Mestre Restrito é resolvido e os custos das restrições são insumos para gerar novas colunas. Para isso acontecer precisamos do problema da mochila para resolver as atribuições viáveis da mochila para resolver as atribuições viáveis x^i para cada máquina i.

O problema da mochila é um problema de otimização combinatória. Tem esse nome por causa analogia feita pela situação de preencher uma mochila com diversos objetos (diferentes pesos e valores). Com isso tem o objetivo de armazenar o maior valor possível sem ultrapassar o peso máximo na mochila.

Concluindo que cada solução viável x^i corresponde a uma coluna. Essa coluna consiste em n valores pertencentes a zero ou um, ou seja, representa se a tarefa j está atribuída a máquina i.

3. O Algoritmo Genético

Algoritmo genético é a tradução do inglês do termo *Genetic Algorithm.* É algoritmo de meta heurística para solucionar o problema de otimização. Segundo (Bealey, 2002) o Algoritmo Genético produzem melhores resultados para problemas combinatória quando comparados com heurísticas baseados na melhorias de uma única solução.

A base desse algoritmo é nos princípios de seleção natural e genética evolutiva de Darwin. Ou seja, os indivíduos que são mais adaptados ao ambiente são selecionados naturalmente e assim sobrevivem, consequentemente, reproduzindo e gerando uma nova população. Segundo (Beasley, 2002) o algoritmo tem uma busca probabilística sagaz, baseados no processo de evolução biológica, dispondo uma população de soluções e assim gerar novas populações a partir da combinação da população inicial.

3.1 Teoria da Evolução

Charles Robert Darwin escreveu, o famoso, livro “A origem das espécies” em 1859. Neste, delata a evolução das espécies, cuja apresenta evolução a partir de ramificação padronizadas. Ressaltando os princípios da seleção natural (DARWIN,1859).

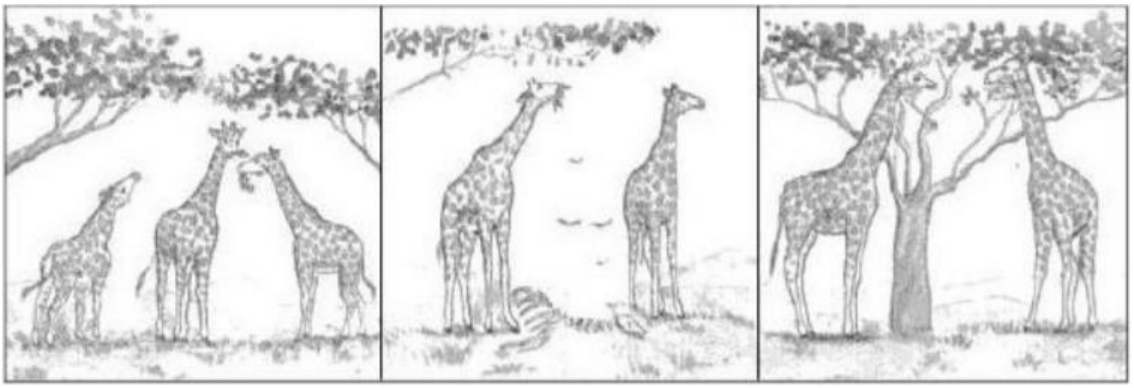
3.1.1 Seleção Natural

A teoria da seleção natural foi proposta por Charles Darwin, consiste em explicar a evolução pelo processo em que os mais aptos sobrevivam, pois há uma luta de sobrevivência no ambiente. Sendo assim, o organismo mais apto consegue sobreviver a esse ambiente.

Um exemplo clássico para explicar essa seleção é o processo evolutivo das girafas. É de conhecimento geral a característica mais marcante da girafa seu pescoço longo. Contudo, antigamente não era exatamente assim, existiam as girafas com pescoço médio, essas conseguiam obter comida através de vegetação em áreas mais baixa. Com o passar dos anos foi ficando cada vez

mais difícil a competição dos alimentos devido a vegetação escassa, de modo que as girafas de pescoço médio morriam por falta de alimento enquanto as girafas de pescoço longo sobreviviam por conseguir se alimentar da vegetação em áreas mais altas. E assim com passar do tempo só sobreviviam as girafas de pescoço longo.

Figura 1 – Evolução das girafas



Fonte: BRUN, 2007

Na figura 1 temos, a esquerda as girafas de pescoço médio não conseguindo se alimentar pois a vegetação é mais alta. Já no meio temos as girafas de pescoço longo se alimentando e girafa de pescoço médio morta. E na direita temos somente as girafas de pescoço longo que foram as mais aptas a esse ambiente.

3.2 Conceitos Básico do Algoritmo Genético

Antes de explicar o Algoritmo Genético necessita compreender uns conceitos iniciais.

Sabe-se que na biologia um cromossomo é composto por milhares de genes, os quais carregam características do indivíduo e podendo ser trocados ou mutados durante o processo de reprodução. Agora no caso do algoritmo fazemos uma analogia o qual o cromossomo corresponde um vetor e os genes são as características do vetor, no caso sendo binária, representando se foi atribuída a tarefa i para o agente j ou se não foi atribuída. E os alelos desses genes seriam os valores do custo dessa atribuição.

Figura 2 – Terminologia

Linguagem Natural	Algoritmo Genético
Cromossomo	Indivíduo, string
Gene	Característica
Alelo	Valor
Locus	Posição
Genótipo	Estrutura
Fenótipo	Conjuntos de parâmetros

Fonte: Elaborada pela autora (2021)

Pela Figura 2 observa-se mais analogia de terminologia. Numa estrutura binária vamos ter um cromossomo de zeros e uns, então os genes são: se foi ou não atribuído a tarefa j para o agente j. Os alelos são o custo do agente realizar a tarefa, caso for atribuída essa tarefa para esse agente. A estrutura desse cromossomo é como os genótipo, por exemplo se esse vetor é binário ou não.

Então o Algoritmo Genético parte de uma população inicial de possíveis soluções e pelo processo de seleção natural sobrevivem os mais aptos, os quais se cruzam e mutam geneticamente gerando uma nova população mais evoluída, ou seja, melhorando as soluções para o problema. E isto repete até achar uma solução ótima que satisfaça todos os critérios de parada. Basicamente o Algoritmo Genético é constituído por 3 fases principais *gerar* a população inicial, *o cruzamento* e mutação dessa população inicial e assim reproduzir uma nova população, *avaliar* essa nova população. Pensando em sempre melhorar as novas populações a fim de no final do algoritmo obter uma solução ótima.

Uma forma de codificar o Problema Generalizado de Atribuição é a posição do vetor são as tarefas a serem realizadas. E os elementos do vetor, que neste caso, será binário será por zeros ou um. Portanto, indica se foi ou não atribuído a tarefa j para agente i. Como por exemplo:

Figura 3 – Exemplo de cromossomo

1	2	3	4	5	....	n-1	n
1	0	0	1	0	...	0	1

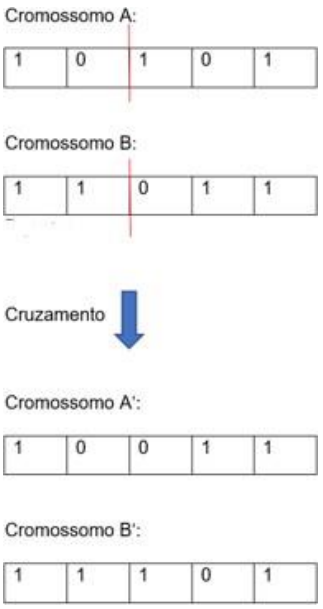
Fonte: Elaborada pela autora (2021)

Então na Figura 3 temos que foi atribuída a tarefa 1 para o agente i além de atribuir a tarefa 4 e tarefa n. E assim podemos gerar várias soluções. E até o momento essa é a melhor maneira de codificação para o Problema Generalizado de Atribuição.

Portanto, cada proposta de solução pode ser facilmente calculada a função objetivo. A função objetivo (fitness) calcula e armazena e assim usa de insumo para escolher uma solução melhor na próxima iteração.

Para gerar novas populações usamos a substituição por meio de descendentes. Assim preserva-se algumas soluções, de melhor qualidade, melhorando a cada iteração. Com a finalidade de realizar o cruzamento, precisa-se de um par de população, quer dizer dois cromossomos. Aleatoriamente escolha um ponto entre 1 a n e troca os elementos gerando novos cromossomos. Como na figura abaixo:

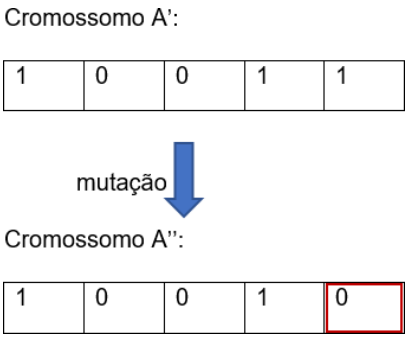
Figura 4 – Cruzamento do cromossomo



Fonte: Elaborada pelo autor (2021)

A fim de evitar codificações com elementos muito semelhantes podemos realizar mutações nos genes (elementos do vetor). O gene é escolhido aleatoriamente em um cromossomo e é mudado. Como a figura a seguir:

Figura 5 – Mutação do cromossomo



Fonte: Elaborada pela autora (2021)

Assim a próxima iteração é gerar uma nova população baseada pelo cruzamento e seleção.

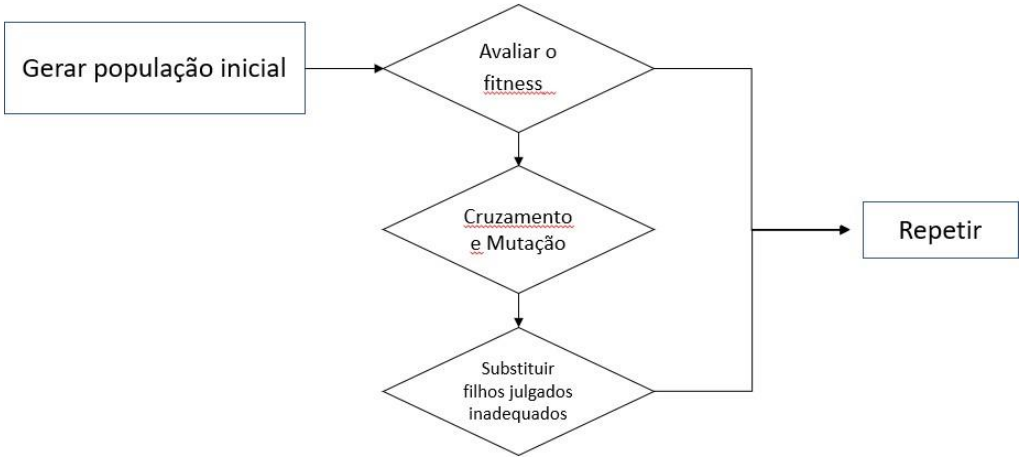
O processo de seleção é verificar se os novos elementos são saudáveis e estão aptos a reprodução. Ou seja, ele segue todos os critérios do Problema Generalizado de Atribuição e não fere nenhuma restrição. E consequentemente a função objetivo melhorará até gerar a melhor solução para o problema, onde para cada tarefa i será realizado pelo agente de menor custo.

Resumindo o Algoritmo Genético segue esses passos:

1) Gerar a população inicial.
2) Avaliar o fitness dessa população.
3) Aplicar o cruzamento e mutação na população inicial.
4) Obter os filhos gerados da reprodução.
5) Avaliar o fitness desses filhos.
6) Repetir os passos 3), 4) e 5) até que o critério de parada seja alcançado.

Podemos visualizar o algoritmo pelo fluxograma:

Figura 6 – Fluxograma do Algoritmo Genético



Fonte: Elaborada pela autora (2021)

E repetimos o processo até obter a melhor solução que seria quando o critério de parada foi alcançado. O critério de parada pode ser o número de iterações máximas que o algoritmo pode realizar.

4. Implementação e Resultados

Inicialmente, vamos apresentar os dados. Os dados que serão usados foram obtidos do Laboratório de Negamochi da Universidade de Kyoto, as instâncias utilizadas são do tipo A de Yagiura(2021) O formato consiste em dispor: o número de agentes (m), número de empregos (n) para cada agente i (i = 1, ..., m) por sua vez: custo de alocar o trabalho j ao agente i (j = 1, ..., n) para cada agente i (i = 1, ..., m) por sua vez: recurso consumido na alocação da tarefa j para o agente i (j = 1, ..., n) capacidade de recursos do agente i (i = 1, ..., m).

4.1 Instâncias

Para melhor compreendimento, usaremos como exemplo um caso com 3 agentes (n=3) e 5 tarefas (m=5).

Aqui temos o custo de alocação de cada agente para as 10 tarefas.

Agente1 = [36, 46, 34, 25, 29]

Agente2 = [12, 33, 26, 10, 32]

Agente3 = [34, 23, 36, 34, 21]

Assim, no exemplo dado o custo de alocar o agente1 na primeira tarefa é de 36, já de alocar na segunda tarefa é de 46.

Agora temos os recursos usados por cada agente para cada das 10 tarefas.

Agente1 = [8, 10, 7, 23, 20]

Agente2 = [22, 9, 13, 12, 22]

Agente3 = [6, 5, 24, 24, 24]

Assim temos os recursos usados que podem ser ,por exemplo, o tempo de execução da tarefa, logo o Agente1 leva 8 minutos para executar a primeira tarefa e o Agente2 para executar a primeira tarefa gasta 12 minutos.

E o recurso máximo que os agentes podem carregar são:

Agente1 = [35]

Agente2 = [35]

Agente3 = [35]

Um agente pode fazer mais de uma tarefa, mas não pode ultrapassar do seu recurso máximo, ou seja, o Agente1 pode fazer mais de uma tarefa, mas não vai ultrapassar o limite de 35. Se usarmos o exemplo anterior, o tempo para realizar uma tarefa, temos que o agente não pode passar de 35 min realizando as tarefas. Depois desse tempo ele não pode realizar mais nenhuma tarefa.

O modelo matemático desse problema, em particular é dado por:

$$Min\ 36x_{1,1} + 46x_{1,2} + 34x_{1,3} + 25x_{1,4} + 29x_{1,5} + 12x_{2,1} + 33x_{2,2} + 26x_{2,3} + 10x_{2,4} + 32x_{2,5} + 34x_{3,1} + 23x_{3,2} + 36x_{3,3} + 34x_{3,4} + 21x_{3,5}$$

Sujeito a:

$$x_{1,1} + x_{2,1} + x_{3,1} = 1$$

$$x_{1,2} + x_{2,2} + x_{3,2} = 1$$

$$x_{1,3} + x_{2,3} + x_{3,3} = 1$$

$$x_{1,4} + x_{2,4} + x_{3,4} = 1$$

$$x_{1,5} + x_{2,5} + x_{3,5} = 1$$

$$8x_{1,1} + 10x_{1,2} + 7x_{1,3} + 23x_{1,4} + 20x_{1,5} \leq 35$$

$$22x_{2,1} + 9x_{2,2} + 13x_{2,3} + 12x_{2,4} + 22x_{2,5} \leq 35$$

$$6x_{3,1} + 5x_{3,2} + 24x_{3,3} + 24x_{3,4} + 24x_{3,5} \leq 35$$

Sabendo que $x_{i,j} \in [0,1]$, um diz que foi atribuída a tarefa ao agente e zero caso contrário.

Após a leitura dos problemas dados por Yagiura (2021) serão realizados testes com o Algoritmo Genético para gerar soluções para o Problema Generalizado de Atribuição.

Usando a decomposição Dantzig-Wolfe, temos como subproblema encontrar soluções $\vec{x}$ satisfazendo:

$\vec{x} \in X = \{[x_{i,j}] \in [0,1] / \; x_{1,1} + x_{2,1} + x_{3,1} = 1, \;\; x_{1,2} + x_{2,2} + x_{3,2} = 1, \; x_{1,3} + x_{2,3} + x_{3,3} = 1, \; x_{1,4} + x_{2,4} + x_{3,4} = 1$ e $x_{1,5} + x_{2,5} + x_{3,5} = 1 \}$

E o Problema Master é dado por:

$$Min \; 36x_{1,1} + 46x_{1,2} + 34x_{1,3} + 25x_{1,4} + 29x_{1,5} + 12x_{2,1} + 33x_{2,2} + 26x_{2,3} + 10x_{2,4} + 32x_{2,5} + 34x_{3,1} + 23x_{3,2} + 36x_{3,3} + 34x_{3,4} + 21x_{3,5}$$

Sujeito à:

$$8x_{1,1} + 10x_{1,2} + 7x_{1,3} + 23x_{1,4} + 20x_{1,5} \leq 35$$

$$22x_{2,1} + 9x_{2,2} + 13x_{2,3} + 12x_{2,4} + 22x_{2,5} \leq 35$$

$$6x_{3,1} + 5x_{3,2} + 24x_{3,3} + 24x_{3,4} + 24x_{3,5} \leq 35$$

$$x_{i,j} \in X$$

No exemplo dado poderíamos ter as seguintes soluções do subproblema:

$$\vec{x}^1 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

No vetor $\vec{x}^1$ a tarefa 1 corresponde aos três primeiros elementos do vetor, já a tarefa 2 está entre o quarto e sexto elemento desse vetor e assim sucessivamente.

Agora para resolver o problema master multiplicamos o vetor da solução do subproblema.

$$Min\ 36x_{1,1} + 12x_{2,1} + 34x_{3,1} + 46x_{1,2} + 23x_{3,2} + 33x_{2,2} + 34x_{1,3} + 26x_{2,3} + 36x_{3,3} + 25x_{1,4} + 10x_{2,4} + 34x_{3,4} + 29x_{1,5} + 32x_{2,5} + 21x_{3,5}$$

Substituindo pelo vetor solução do subproblema:

$$Min\ 36(1) + 12(0) + 34(0) + 46(0) + 23(1) + 33(0) + 34(1) + 26(0) + 36(0) + 25(0) + 10(1) + 34(0) + 29(0) + 32(0) + 21(1)$$

Então temos o seguinte custo minimizado:

$$Min\ 36 + 23 + 34 + 10 + 21 = 124$$

Para validar essa solução precisa ser verificada nas condições abaixo

$$8(1) + 10(0) + 7(1) + 23(0) + 20(0) \leq 35$$

$$22(0) + 9(0) + 13(0) + 12(1) + 22(0) \leq 35$$

$$6(0) + 5(1) + 24(0) + 24(0) + 24(1) \leq 35$$

Resolvendo

$$15 \leq 35$$

$$12 \leq 35$$

$$29 \leq 35$$

Como a solução satisfaz todas as restrições é uma possível solução para o Problema Generalizado de Atribuição. Neste exemplo não pode afirmar que é a melhor solução, porém podemos afirmar que é uma solução possível para esse problema.

4.2 Implementação

A implementação do Algoritmo Genético nesse estudo foi realizada pelo programa R.

O programa R é um software de análise de dados. Foi desenvolvido por Ross Ihaka e Robert Gentleman em 1996 e elaboraram uma nova linguagem. O software é gratuito e tem uma linguagem de fácil entendimento, está sempre se atualizando com novos pacotes para facilitar o uso, muito flexível, tem capacidade gráfica e está disponível para diferentes plataformas: Windows, Linux e Mac.

Nesse programa existem pacotes que oferecem o resultado basicamente completo, ou seja, não necessita realizar todos os passos para chegar no resultado. Contudo, a fim de enriquecer mais o estudo foi realizado uma implementação sem o uso de bibliotecas ou pacotes do programa.

Deve ressaltar a importância do uso dos algoritmos, pois ao usar dados reais tem uma extensa quantidade de dados para analisar ficando inviável de resolver a mão. Por isso é de extrema importância o estudo e conhecimento dos algoritmos.

Para a implementação devemos seguir e ideia do fluxograma. A implementação do Algoritmo Genético é fluxograma em uma outra linguagem, uma linguagem computacional.

Lembrando que os dados usados para a implementação são dados usados por pesquisadores. São instâncias do tipo A Yagira (2021), os quais forem apresentados acima.

Assim começamos a implementação construindo uma classe com os custos e recursos de cada agente aplicados a cada tarefa.

Figura 7 – Código Algoritmo Genético

```
setClass (“Tarefa”){

c=(Agente, Recursos Usados,Custo)

Lista = ("Tarefa", nome = "Agente11", Recurso Usados= 8, Custo = 8),

("Tarefa ", nome = "Agente12", Recurso Usados = 10, Custo = 10),

("Tarefa ", nome = "Agente13", Recurso Usados = 7, Custo = 7),

("Tarefa ", nome = "Agente14", Recurso Usados = 23, Custo = 23),

("Tarefa ", nome = "Agente15", Recurso Usados = 20, Custo = 20),

("Tarefa ", nome = "Agente21", Recurso Usados = 22, Custo = 12),

("Tarefa ", nome = "Agente22", Recurso Usados = 9, Custo =33),

("Tarefa ", nome = "Agente23", Recurso Usados = 13, Custo = 26),

("Tarefa ", nome = "Agente24", Recurso Usados = 12, Custo = 10),

("Tarefa ", nome = "Agente25", Recurso Usados = 22, Custo = 32),

("Tarefa ", nome = "Agente31", Recurso Usados = 6, Custo = 34),

("Tarefa ", nome = "Agente32", Recurso Usados = 5, Custo = 23),

("Tarefa ", nome = "Agente33", Recurso Usados = 24, Custo = 36),

("Tarefa ", nome = "Agente34", Recurso Usados = 24, Custo = 34),

("Tarefa ", nome = "Agente35", Recurso Usados = 24, Custo = 21))
```

Fonte: Elaborada pela autora (2022)

Posteriormente entramos com o conceito de indivíduo, o qual no código que seria o cromossomo, ou seja,uma representação de uma solução. Neste usaremos solução binária o número um representa que foi atribuído a tarefa i para o agente j e zero caso contrário. Estabelecendo assim uma função a qual gera cromossomos de forma aleatória.

Figura 8 – Código Algoritmo Genético

```
cromossomo = sample(x = c("0","1"), size = tamanhoCromossomo, replace = FALSE)
```

Fonte: Elaborada pela autora (2022)

Dessa maneira temos que criar uma função para avaliar a solução gerada, ou seja, analisar o cromossomo gerado pela função anterior. Esse modo de analisar pode ser chamado de fitness, que é o valor usado para podemos escolher a melhor solução. Logo cria-se uma medida de qualidade para identificar qual cromossomo é o melhor, e se é uma solução aceitável para evolução.

Figura 9 – Código Algoritmo Genético

```
for (i in 1:tamanhoCromossomo {
   if (cromossomo[i] == '1'){
     nota = nota + Custos[i]
     somaRecusoUsados = somaRecusoUsados + RecursoUsados[i]
   }
 }
 if (somaRecusoUsados < limite RecursoUsados){
   continua
else {
gerar novos cromossomos}
 }
```

Fonte: Elaborada pela autora (2022)

Logo temos uma função que avalia os custos dos indivíduos, portanto agora cria-se uma função para fazer o crossover entre os indivíduos gerados.

Seguindo o fluxo do código foi gerado dos indivíduos iniciais e irão introduzir para função de crossover.

Figura 10 – Código Algoritmo Genético

```
corte = sample (indices,1)
if (corte == tamanhoCromossomoA){
   filho1 = CromossomoA [1:corte]
   filho2 = CromossomoB[1:corte]
 } else{

   filho1 = c(CromossomoB [1:corte], CromossomoA [(corte+1):length(CromossomoA)])
   filho2 = c(CromossomoA [1:corte], CromossomoB [(corte+1):length(CromossomoB)])
 }
```

Fonte: Elaborada pela autora (2022)

Lembrando que a ideia do crossover é a reprodução. Combina-se pedações de cromossomo gerando filhos mais aptos e consequentemente ao passar das gerações a população tende a evoluir.

Neste caso foi usado o ponto de corte para aplicar o crossover, o ponto de corte é gerado de forma aleatória e assim dividimos os cromossomos formando outros com intuito de ser melhores que a geração anterior.

Além da reprodução foi criado a função de mutação, a qual tem o objetivo de diversidade e mudar aleatoriamente os genes e é aplicada de forma menos frequente do que a reprodução, como na natureza. Nessa função possui uma taxa associada a probabilidade extremamente baixa para alterar o gene.

Figura 11 – Código Algoritmo Genético

```
for (i in 1: ( tamanhoCromossomo)){
  if (runif(n = 1, min = 0, max =1)< taxaMutacao){
    if (cromossomo[i]=='1'){

cromossomo[i] = '0'
    } else {
      cromossomo[i] = '1'
    }
  }
}
```

Fonte: Elaborada pela autora (2022)

Agora que temos essas funções criadas podemos inicializar uma população e assim introduzir o Algoritmo Genético.

Figura 12 – Código Algoritmo Genético

```
for (i in 1: tamanhoPopulacao){
    populacao = gerarCromossomo(tamanhoCromossomo))
```

Fonte: Elaborada pela autora (2022)

Após gerar a população precisa avaliar essa população para isto ocorrer elabora-se uma função que ordena a população assim conseguimos visualizar quais indivíduos possuem o menor custo.

Figura 13 – Código Algoritmo Genético

```
for (Individuo in populacao){
  notas = notaAvaliacao
 }
 listaPosicao = order(notasAvaliacao, decreasing = TRUE)
 for (i in 1:length(listaPosicao)){
  populacaoOrdenada = c(populacaoOrdenada,populacao[[listaPosicao[i]]])
 }
```

Fonte: Elaborada pela autora (2022)

Seguimos com a função adicional para dar apoio a função anterior para selecionar os melhores indivíduos.

Figura 14 – Código Algoritmo Genético

```
if (notaAvaliacao > melhorSolucao){
  melhorSolucao = Individuo
```

Fonte: Elaborada pela autora (2022)

Após a escolha do melhor indivíduo forma-se um ciclo para escolha da melhor solução. Esse ciclo interrompe quando encontra a melhor solução ou o critério de parada seja alcançado.

Deselvovendo o código no software R temos as seguintes possíveis solução:

Figura 15 – Soluções do Algoritmo Genético para o problema

	Tarefa 1	Tarefa 2	Tarefa 3	Tarefa 4	Tarefa 5
Cromossomo	1 1 1	1 1 1	1 1 1	1 1 1	1 1 1
Cromossomo	1 1 1	1 1 1	1 1 1	1 1 1	1 1 1

Cromossomo	1 1 1	1 1 1	1 1 1	1 1 1	1 1 1
Cromossomo	1 1 0	0 1 1	1 0 1	1 1 1	1 1 0
Cromossomo	0 0 1	0 1 0	0 0 1	0 1 1	0 1 0
Cromossomo	0 0 1	0 1 0	0 0 1	0 0 1	0 1 1
Cromossomo	0 0 1	0 1 0	0 0 1	0 0 1	0 1 0
Cromossomo	0 0 1	0 1 0	0 0 1	0 1 1	1 1 0
Cromossomo	0 0 1	1 0 1	0 0 1	0 0 1	1 0 1
Cromossomo	0 0 1	0 1 0	0 0 1	1 0 1	1 0 1
Cromossomo	0 0 1	0 1 0	0 0 1	0 1 1	0 1 0
Cromossomo	0 0 1	0 1 0	0 0 1	0 1 1	0 1 0
Cromossomo	0 0 1	0 1 0	0 0 1	0 0 1	0 0 1
Cromossomo	0 1 0	1 1 0	0 0 1	0 1 1	0 1 0
Cromossomo	0 0 1	0 1 0	0 0 1	0 0 1	0 1 0
Cromossomo	0 1 1	0 1 0	0 0 1	0 0 1	0 1 0
Cromossomo	0 0 1	0 1 0	0 1 1	0 0 1	0 1 0
Cromossomo	0 0 1	0 1 0	0 0 1	0 0 1	0 1 0
Cromossomo	0 0 1	0 1 0	0 0 1	0 0 1	0 1 1
Cromossomo	1 0 1	0 1 0	0 0 1	0 0 1	0 1 0
Cromossomo	0 0 1	0 1 0	0 0 1	0 0 1	0 1 0
Cromossomo	1 0 1	0 1 0	0 0 1	0 0 1	0 1 0
Cromossomo	0 0 1	1 1 0	0 0 1	0 0 1	0 1 0
Cromossomo	0 0 1	0 1 0	0 0 1	0 1 1	1 0 1
Cromossomo	0 1 1	0 1 0	0 0 1	0 0 1	0 1 0
Cromossomo	0 0 1	1 1 0	0 0 1	1 0 1	1 0 1
Cromossomo	0 0 1	1 0 1	0 1 1	0 0 1	0 1 0
Cromossomo	0 0 1	0 1 0	0 0 1	0 0 1	0 1 0

Cromossomo	0 0 1	0 1 1	0 1 1	0 0 1	0 1 0
Cromossomo	1 0 1	0 1 0	1 0 1	1 0 1	0 1 0
Cromossomo	0 0 1	1 1 0	0 0 1	0 0 1	1 1 0
Cromossomo	1 0 1	1 1 0	0 0 1	1 0 1	0 1 0
Cromossomo	0 1 1	0 1 0	0 0 1	0 0 1	0 1 0
Cromossomo	1 0 1	0 1 0	0 0 1	0 0 1	1 1 0
Cromossomo	0 0 1	0 1 1	0 0 1	0 0 1	0 1 0
Cromossomo	0 1 1	0 1 0	0 0 1	0 0 1	0 1 0
Cromossomo	0 0 1	0 1 0	0 0 1	0 0 1	0 1 0
Cromossomo	0 0 1	0 1 1	0 0 1	0 0 1	0 1 0
Cromossomo	1 0 1	0 1 0	0 0 1	1 0 1	0 1 0
Cromossomo	0 1 1	0 1 0	1 1 1	0 0 1	0 1 0
Cromossomo	0 0 1	0 1 0	0 0 1	0 0 1	1 0 1
Cromossomo	1 0 1	0 1 0	0 0 1	0 0 1	1 1 0
Cromossomo	0 1 1	1 1 0	0 0 1	0 0 1	1 1 0
Cromossomo	0 0 1	0 1 1	0 0 1	0 0 1	0 1 0
Cromossomo	0 0 1	0 1 0	0 0 1	0 1 1	1 1 0
Cromossomo	0 0 1	1 1 0	0 0 1	0 0 1	0 1 0
Cromossomo	0 1 1	0 1 0	0 0 1	1 0 0	0 1 1
Cromossomo	0 0 1	0 1 0	1 0 1	0 0 1	1 0 1
Cromossomo	0 0 1	0 1 0	0 0 1	0 0 1	0 1 0
Cromossomo	0 1 1	0 1 1	0 1 0	0 1 1	0 1 1
Cromossomo	0 0 1	1 0 1	0 0 1	0 0 1	0 1 0
Cromossomo	0 0 1	1 1 0	0 0 1	0 1 1	0 1 0
Cromossomo	0 0 1	0 1 0	0 0 1	0 0 1	0 1 1
Cromossomo	0 0 1	0 1 0	0 0 1	1 0 1	1 1 0

Cromossomo	1 1 1	0 1 0	0 0 1	1 0 1	0 1 0
Cromossomo	1 0 1	0 1 0	0 0 1	0 1 1	0 1 0
Cromossomo	0 0 1	0 1 1	0 0 1	0 0 1	1 1 0
Cromossomo	0 0 1	0 1 0	0 0 1	0 0 1	0 1 0
Cromossomo	0 0 1	0 1 0	0 1 1	0 0 1	0 1 0
Cromossomo	1 1 1	0 1 0	0 1 1	0 0 1	1 1 0
Cromossomo	0 0 1	0 1 0	0 0 1	0 0 1	0 1 0
Cromossomo	0 0 1	0 1 1	1 0 1	0 0 1	1 0 1
Cromossomo	0 0 1	0 1 0	0 0 1	0 0 1	0 1 0
Cromossomo	0 0 1	1 0 1	0 1 1	1 0 1	0 1 0
Cromossomo	1 0 1	0 1 0	0 0 1	1 0 1	0 1 0
Cromossomo	0 0 1	0 1 1	0 0 1	0 0 1	1 1 0
Cromossomo	0 0 1	0 1 0	0 0 1	1 0 1	1 0 1
Cromossomo	0 0 1	0 1 0	0 1 1	0 0 1	0 1 1
Cromossomo	0 0 1	1 1 0	0 0 1	1 0 1	1 0 1
Cromossomo	0 0 1	0 1 0	0 0 1	0 0 1	0 1 0
Cromossomo	0 0 1	0 1 1	0 0 1	0 0 1	0 1 0
Cromossomo	0 0 1	0 1 0	0 0 1	0 0 1	1 1 0
Cromossomo	0 0 1	1 1 1	0 0 1	0 0 1	1 1 0
Cromossomo	0 0 1	0 0 1	0 0 1	0 0 1	0 1 0
Cromossomo	0 0 1	0 1 0	1 0 1	0 0 1	0 1 0
Cromossomo	0 0 1	0 1 0	0 0 1	0 0 1	0 1 0
Cromossomo	0 0 1	0 1 0	0 1 1	1 0 1	1 1 0
Cromossomo	0 0 1	1 0 1	0 1 1	0 0 1	0 1 0
Cromossomo	0 0 1	0 1 0	0 0 1	0 0 1	0 1 0
Cromossomo	0 0 1	0 0 1	1 0 1	1 0 1	0 1 0

Cromossomo	0 1 1	0 1 0	0 1 1	0 0 1	0 1 1
Cromossomo	0 0 1	0 1 0	0 0 1	0 0 1	0 1 0
Cromossomo	0 0 1	1 1 0	0 0 1	0 0 1	0 1 0
Cromossomo	0 0 1	1 1 0	0 0 1	0 0 1	0 1 0
Cromossomo	0 0 1	0 1 1	1 0 1	0 0 1	1 0 1
Cromossomo	0 0 1	1 1 0	0 0 1	1 0 1	1 0 1
Cromossomo	0 1 1	0 1 0	0 0 1	0 0 1	0 1 0
Cromossomo	1 0 1	0 1 0	0 1 1	0 0 1	0 1 0
Cromossomo	0 1 1	0 1 0	0 1 1	0 0 1	0 1 0
Cromossomo	0 1 1	0 1 0	0 0 1	0 1 1	1 0 1
Cromossomo	0 0 1	0 1 0	0 0 1	0 1 1	1 0 1
Cromossomo	0 1 1	0 1 1	0 0 1	1 0 1	0 1 0
Cromossomo	0 0 1	1 1 1	0 0 1	1 0 1	0 1 1
Cromossomo	0 0 1	0 1 1	0 0 1	0 1 1	1 1 0
Cromossomo	0 1 0	1 0 1	0 0 1	0 0 1	0 1 0
Cromossomo	0 0 1	0 1 0	0 1 0	0 0 1	0 1 1
Cromossomo	1 0 1	0 1 0	0 0 1	0 0 1	0 1 0
Cromossomo	0 0 1	0 1 1	0 0 1	0 0 1	0 1 0
Cromossomo	0 0 1	0 1 1	0 0 1	0 0 1	1 0 1
Cromossomo	1 0 1	0 1 1	0 0 1	0 0 1	0 1 0
Cromossomo	0 0 1	1 0 1	0 0 1	0 1 1	1 1 0

Fonte: Elaborada pela autora (2022)

Posteriomente de visualizar todas as soluções, temos uma ordenção com base no fitness de cada solução, o qual seria o menor custo das realização das

tarefas. Então após ordenado seleciona a primeira solução para ser a melhor solução do Algoritmo Genético, sendo ela :

	Tarefa 1	Tarefa 2	Tarefa 3	Tarefa 4	Tarefa 5
Cromossomo	0 1 0	0 0 1	0 1 0	1 0 0	0 0 1

Concluindo que o Agente 1 fez a quarta tarefa, o Agente 2 fez a primeira e a terceira tarefa e já o Agente 3 fez a segunda e a quinta tarefa. Aqui visualmente vemos que todas as tarefas foram feitas o que satisfaz uma restrição.

Em suma para certificar que essa solução é uma resposta do sub problema após a decomposição Dantzig-Wolfe:

$\vec{x} \in X = \{[x_{i,j}] \in [0,1] / \ x_{1,1} + x_{2,1} + x_{3,1} = 1, \ \ x_{1,2} + x_{2,2} + x_{3,2} = 1, \ x_{1,3} + x_{2,3} + x_{3,3} = 1, \ x_{1,4} + x_{2,4} + x_{3,4} = 1$ e $x_{1,5} + x_{2,5} + x_{3,5} = 1 \}$

Substituindo, temos:

$\vec{x} \in X = \{[x_{i,j}] \in [0,1] / \ 0 + 1 + 0 = 1, \ 0 + 0 + 1 = 1, \ 0 + 1 + 0 = 1, \ 1 + 0 + 0 = 1$ e $0 + 0 + 1 = 1 \}$

Resolvendo o problema master multiplicamos o vetor da solução do subproblema.

$$Min \ 36x_{1,1} + 12x_{2,1} + 34x_{3,1} + 46x_{1,2} + 33x_{3,2} + 23x_{2,2} + 34x_{1,3} + 26x_{2,3} + 36x_{3,3} + 25x_{1,4} + 10x_{2,4} + 34x_{3,4} + 29x_{1,5} + 32x_{2,5} + 21x_{3,5}$$

Substituindo pelo vetor solução do subproblema:

$$Min \ 36(0) + 12(1) + 34(0) + 46(0) + 33(0) + 23(1) + 34(0) + 26(1) + 36(0) + 25(1) + 10(0) + 34(0) + 29(0) + 32(0) + 21(1)$$

Então temos o seguinte custo minimizado:

$$Min \ 12 + 23 + 26 + 25 + 21 = 107$$

Para validar essa solução precisa ser verificada nas condições abaixo.

$$8(0) + 10(0) + 7(0) + 23(1) + 20(0) \leq 35$$

$$22(1) + 9(0) + 13(1) + 12(0) + 22(0) \leq 35$$

$$6(0) + 5(1) + 24(0) + 24(0) + 24(1) \leq 35$$

Resolvendo

$$23 \leq 35$$

$$35 \leq 35$$

$$29 \leq 35$$

Assim concluimos que a melhor solução escolhida pelo Algoritmo Genético satisfaz todas as restrições do Problema Generalizado de Atribuição.

5. Conclusões

Neste trabalho foi apresentado o Problema Generalizado de Atribuição, suas aplicações e sua modelagem. Foi visto que esse problema é muito estudado e tem diversas propostas de resolução. Uma das propostas é decompor e usar o algoritmo genético, que é uma meta heurística simples e eficiente encontrar boas soluções para o problema decomposto. Além de ser flexível e se adaptar habilmente ao problema.

No capítulo 2 foi apresentada uma breve revisão do problema generalizado de atribuição a partir de 2017 e o método de decomposição do problema. O capítulo 3 abordou o algoritmo genético, seus princípios e exemplos de cruzamentos e mutação.

A implementação do algoritmo foi realizada sem a utilização de pacotes disponíveis em diferentes softwares como forma de estudar o comportamento do algoritmo genético para o problema decomposto do problema generalizado de atribuição.

Do estudo realizado neste trabalho pode-se concluir que o algoritmo genético é um método não complexo e que pode ser muito explorado se combinado a outros métodos e heurísticas. Além disso, como o problema generalizado de atribuição tem uma diversificação muito grande de aplicações, recomenda-se que este tema seja explorado em pesquisas futuras.

6. Referências

ASLLANI, A; LARI, A. **Using genecti algorithm for dynamic and multiple criteria web-site optimizations**. Rearchgate,2007. Disponível em<https://www.researchgate.net/publication/222435760_Using_genetic_algorithm_for_dynamic_and_multiple_criteria_web-site_optimizations> Acesso em 22 de setembro de 2021.

BOKHARI, S. H. **Assignment Problems in Parallel and Distributed Computing,** Springer, 1987.

BRUN, A. L. **Algoritmos genéticos**. EPAC - Encontro Paranaense de Computação, 2007.

CATTRYSSE, D.G. e VAN WASSENHOVE, L.N. **A survey of algorithms for the generalized assignment problem**. *European Journal of Operational Research*, v. 60, p. 260-272, 1992.

DARWIN, C. R. **On the origin of species by means of natural selection: By means of natual selection**. [S.l.]: John Murray, 1859.

FISCHETTI, M.; LEPSCHY, C.; MINERVA, G.; ROMANIN-JACUR, G.; TOTO, E. **Frequency assignment in mobile radio systems using branch-and-cut techniques. European Journal of Operational Research**, v. 123, p. 241-255, 2000.

FISHER, M. L.; J”ORNSTEEN, K. O.; MADSEN, O. B. G. **Vehicle routing with time windows: Two optimization algorithms, operations research**. Revista Transportes da ANPET Associação Nacional de Pesquisa e Ensino em Transportes, 1979.

GEN, M e CHENG, R **Genetic algorithms and engineering design.** Editora John Wiley &S Sons, 1997.

GOLDBARG, M. C. e LUNA, H. P. **Otimização combinatória e programação linear.** Edtora Campus, 2005.

HOLLAND, J. H. **Adaptation in Natural and Artificial Systems: An introductory analysis with applications to biology, control and artificial intelligence**. [S.l.]: MIT Press Cambridge, MA, USA, 1975.

HU, Y E LIU, Q, **A Network Flow Algorithm for Solving Generalized Assigment Problem**, Mathematical Problems in Engineering, vol 2021, Artigo, 2021.

KUHN, H.W. **The Hungarian method for the assignment problem. Naval Research Logistic Quaterly**, v. 2, p. 83-97, 1955.

LOPES, H.S. RODRIGUES, L. C. A. e STEINER, M. T. A.. **Meta-Heurísticas em pesquisa operacional**. Curitiba, PR. Editora Omnipax, 2013.

MARTELLO, S., TOTH, P. **Algoritmos heurísticos para o problema da mochila múltipla**. *Computing* 27**,** 93-112 (1981).

NOWAKOVSJI, J.; SCHWARZLER, W.; TRIESCH, E. **Using the generalized assignment problem in scheduling the ROSAT space telescope. European Journal of Operational Research**, v. 112, n. 3, p. 531-541, 1999.

PHUDHAD, P; SETHANAN, K E JAMRUS, T, **Uma otimização de enxame de partículas híbridas para o problema de atribuição generalizada com janelas de tempo,** MATEC Web Conf.,192 (2018).

RULAND, K.S. **A model for aeromedical routing and scheduling. International Transactions in Operational Research**, v. 6, n. 1, p. 57-73, 1999.

WRIGHT, M.B. **Speeding up the Hungarian algorithm. Computers and Operations Research**, v. 17, n. 1, p. 95-96, 1990.

YAGIURA, M. ***Generalized assignment problem instances***. Disponível em: <https://www-or.amp.i.kyoto-u.ac.jp/members/yagiura/gap/>. Acesso em: 22 de setembro de 2021.

TANG F. e CHAKRABORTY N, "**Algorithm for Multi-Robot Chance-Constrained Generalized Assignment Problem with Stochastic Resource Consumption**", *2020 IEEE / RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020.

7. APÊNDICE – Exemplo prático usando o Algoritmo Genético

Para facilitar o entendimento e afim de visualizar de forma mais clara do Algoritmo Genético, será apresentado o algoritmo genético para um problema clássico de transporte.

O Problema de transporte tem como objetivo estabelecer o maior lucro em transportar produtos de várias origens para vários destinos. O Problema de transporte também é um Problema Linear e tem aplicação direta com a logística.

O problema apresentado será a logística de transporte de produtos do estoque para uma loja física. Então o problema é achar uma solução que leve os produtos de maior valor e sem gastar espaço no caminhão, pois o limite máximo do caminhão é de 3 m^3.

Temos a seguinte lista de produtos no estoque, lembrando que o objetivo é levar o maior valor de produtos e transportar a maior quantidade de produto.

"Geladeira Dako", espaço = 0.751, valor = 999.99

"Iphone 6", espaço = 0.0000899, valor = 2911.12

"TV 55", espaço = 0.400, valor = 4346.99

"Tv 50", espaço = 0.290, valor = 3999.90

"TV 42", espaço = 0.200, valor = 2999.00

"Notebook Dell", espaço = 0.00350, valor = 2499.90

"Ventilador Panasonic", espaço = 0.496, valor = 199.90

"Microondas Eletrolux", espaço = 0.0424, valor = 308.66

"Microondas LG", espaço = 0.0544, valor = 429.90

"Microondas Panasonic", espaço = 0.0319, valor = 299.29

"Geladeira Brastemp", espaço = 0.635, valor = 849.00

"Geladeira Consul", espaço = 0.870, valor = 1199.89

"Notebook Lenovo", espaço = 0.498, valor = 1999.90

"Notebook Asus", espaço = 0.527, valor = 3999.00

Após a iniciar a algoritmo genético, temos as seguintes soluções:

G: 0 Cromossomo: 1 1 0 1 1 1 1 0 0 0 0 0 1 1

G: 1 Cromossomo: 1 0 0 0 1 0 0 1 1 1 0 1 1 1

G: 2 Cromossomo: 0 1 0 0 1 0 1 1 1 1 0 1 1 1

G: 3 Cromossomo: 1 0 0 0 1 0 0 1 1 1 0 1 1 1

G: 4 Cromossomo: 1 0 0 0 1 1 1 1 1 1 0 0 0 1

G: 5 Cromossomo: 1 0 0 0 1 0 0 1 1 1 0 1 1 1

G: 6 Cromossomo: 1 0 0 0 1 0 0 1 1 0 0 1 1 1

G: 7 Cromossomo: 1 0 0 0 1 0 1 1 1 1 0 1 1 0

G: 8 Cromossomo: 1 0 0 0 1 0 0 1 1 1 0 1 1 1

G: 9 Cromossomo: 0 0 0 0 1 0 1 1 1 1 0 1 1 1

G: 10 Cromossomo: 1 0 0 0 1 1 0 1 1 1 0 1 1 1

G: 11 Cromossomo: 0 0 0 0 1 1 1 1 1 1 0 1 1 1

G: 12 Cromossomo: 1 0 0 0 1 0 0 1 1 1 0 1 1 1

G: 13 Cromossomo: 1 0 0 0 1 0 1 1 1 1 0 1 0 1

G: 14 Cromossomo: 1 0 0 0 1 0 1 1 1 1 0 0 1 1

G: 15 Cromossomo: 0 0 0 0 1 1 1 1 1 1 0 1 1 1

G: 16 Cromossomo: 1 0 0 0 1 0 0 1 1 0 0 1 1 1

G: 17 Cromossomo: 1 0 0 0 1 0 1 1 1 0 0 0 1 1

G: 18 Cromossomo: 1 1 0 0 1 0 0 1 1 0 0 1 1 1

G: 19 Cromossomo: 1 0 0 0 1 0 0 1 1 1 0 1 1 1

G: 20 Cromossomo: 1 0 0 0 1 0 0 1 1 0 0 1 1 1

G: 21 Cromossomo: 1 0 0 0 1 0 1 1 1 0 0 0 1 1

G: 22 Cromossomo: 0 0 0 0 1 0 0 0 1 0 1 1 1 1

G: 23 Cromossomo: 1 0 0 1 1 1 1 0 1 0 1 1 1 1

G: 24 Cromossomo: 0 1 0 0 1 1 1 0 1 0 0 1 1 1

G: 25 Cromossomo: 0 1 0 1 0 1 1 0 1 0 1 1 0 1

G: 26 Cromossomo: 1 0 0 0 1 1 1 0 1 0 1 0 0 1

G: 27 Cromossomo: 0 0 0 0 0 1 1 0 0 0 1 1 0 1

G: 28 Cromossomo: 1 0 0 0 1 1 1 0 1 0 0 1 0 1

G: 29 Cromossomo: 1 0 0 0 0 1 1 0 1 0 0 1 0 1

G: 30 Cromossomo: 0 1 0 0 0 1 1 0 0 0 1 1 0 1

G: 31 Cromossomo: 1 0 0 0 0 1 1 0 0 0 0 1 0 1

G: 32 Cromossomo: 0 0 0 0 0 1 0 0 0 1 1 1 0 1

G: 33 Cromossomo: 0 0 0 0 0 1 1 1 0 0 1 1 0 1

G: 34 Cromossomo: 0 1 0 0 0 1 1 1 0 0 1 0 1 1

G: 35 Cromossomo: 1 1 0 0 0 1 0 0 0 1 1 1 0 1

G: 36 Cromossomo: 1 1 0 0 0 1 0 0 1 1 1 1 0 1

G: 37 Cromossomo: 1 1 0 0 0 1 0 0 0 1 1 1 0 1

G: 38 Cromossomo: 1 1 0 0 0 1 1 0 0 1 1 0 1 1

G: 39 Cromossomo: 1 1 0 0 0 1 1 0 0 1 1 0 1 1

G: 40 Cromossomo: 1 1 0 0 0 0 1 0 1 1 1 0 0 1

G: 41 Cromossomo: 1 0 0 0 0 1 1 0 0 1 0 1 0 1

G: 42 Cromossomo: 1 0 0 0 0 0 1 1 0 1 1 0 1 1

G: 43 Cromossomo: 0 0 0 1 0 0 1 0 0 1 1 1 0 1

G: 44 Cromossomo: 1 0 0 1 0 0 1 1 0 1 1 1 1 1

G: 45 Cromossomo: 1 0 0 1 0 0 1 1 0 1 1 0 0 1

G: 46 Cromossomo: 1 0 0 0 0 0 1 1 0 0 0 1 0 1

G: 47 Cromossomo: 1 0 0 0 1 1 0 1 0 0 1 0 1 1

G: 48 Cromossomo: 1 0 1 0 0 0 0 1 0 0 1 0 1 1

G: 49 Cromossomo: 1 0 0 0 0 0 0 1 0 0 0 1 1 1

G: 50 Cromossomo: 0 0 0 0 0 1 0 1 0 0 1 1 1 1

G: 51 Cromossomo: 1 0 0 0 0 0 0 1 1 0 1 0 1 1

G: 52 Cromossomo: 1 0 1 0 0 0 0 1 0 0 1 0 1 1

G: 53 Cromossomo: 1 0 0 0 0 0 0 1 0 0 1 0 1 1

G: 54 Cromossomo: 0 0 0 0 1 1 0 1 0 0 1 1 1 1

G: 55 Cromossomo: 1 1 0 1 1 0 0 1 0 0 1 0 1 1

G: 56 Cromossomo: 1 0 0 1 1 0 0 0 0 0 1 0 1 1

G: 57 Cromossomo: 0 0 1 1 1 0 0 1 0 0 1 0 1 1

G: 58 Cromossomo: 1 0 0 1 0 0 0 1 0 0 0 1 1 1

G: 59 Cromossomo: 0 0 0 1 1 0 0 1 0 0 0 1 1 1

G: 60 Cromossomo: 1 0 1 1 1 0 0 1 0 0 0 0 1 1

G: 61 Cromossomo: 0 0 0 1 1 0 0 1 0 0 0 1 1 1

G: 62 Cromossomo: 1 1 0 1 1 0 0 1 0 0 0 1 1 0

G: 63 Cromossomo: 1 0 0 1 0 0 0 1 0 0 0 1 1 1

G: 64 Cromossomo: 0 0 0 1 1 0 0 1 0 0 0 1 1 1

G: 65 Cromossomo: 0 0 0 1 0 0 1 1 0 0 0 1 1 1

G: 66 Cromossomo: 1 0 1 1 0 1 1 1 0 0 0 1 1 1

G: 67 Cromossomo: 1 0 1 1 0 1 0 1 0 0 0 0 1 1

G: 68 Cromossomo: 1 0 0 1 0 0 1 1 0 0 0 1 1 1

G: 69 Cromossomo: 1 0 1 1 0 0 1 1 0 0 0 1 1 0

G: 70 Cromossomo: 1 0 1 1 0 0 1 1 0 0 0 1 1 0

G: 71 Cromossomo: 1 0 1 1 0 0 1 1 0 1 0 1 0 1

G: 72 Cromossomo: 1 0 1 1 0 0 0 1 0 1 0 1 0 1

G: 73 Cromossomo: 1 1 1 1 0 0 0 1 0 1 0 1 0 1

G: 74 Cromossomo: 0 1 1 1 1 0 1 1 0 1 0 1 0 1

G: 75 Cromossomo: 1 1 1 1 0 0 0 0 0 1 0 1 0 1

G: 76 Cromossomo: 1 0 1 1 0 0 0 1 1 1 0 1 0 1

G: 77 Cromossomo: 1 0 1 1 0 0 0 1 1 1 0 1 0 1

G: 78 Cromossomo: 1 0 1 1 0 1 0 1 1 1 0 1 0 1

G: 79 Cromossomo: 0 0 0 1 0 1 1 0 1 1 0 1 0 1

G: 80 Cromossomo: 1 0 0 1 0 1 1 0 1 1 0 0 1 0

G: 81 Cromossomo: 1 0 0 1 0 1 1 1 1 1 0 0 1 1

G: 82 Cromossomo: 1 0 0 1 0 1 1 1 0 1 0 1 1 0

G: 83 Cromossomo: 1 1 0 1 0 1 1 1 1 1 0 0 1 0

G: 84 Cromossomo: 1 1 0 1 0 1 1 1 1 1 0 0 1 0

G: 85 Cromossomo: 1 0 0 1 0 1 1 1 1 1 0 0 1 1

G: 86 Cromossomo: 1 1 0 1 0 1 1 0 1 1 0 0 1 0

G: 87 Cromossomo: 1 0 0 1 0 1 0 1 1 0 0 1 1 0

G: 88 Cromossomo: 0 0 0 1 0 1 1 1 1 1 0 1 1 0

G: 89 Cromossomo: 1 1 0 1 0 1 1 0 1 0 0 1 1 0

G: 90 Cromossomo: 1 0 1 1 0 1 0 1 1 1 0 1 1 0

G: 91 Cromossomo: 0 0 1 1 0 1 1 1 1 0 1 0 1 0

G: 92 Cromossomo: 0 0 1 1 0 1 1 1 1 0 1 0 1 1

G: 93 Cromossomo: 1 0 0 1 0 1 0 1 1 1 1 0 1 1

G: 94 Cromossomo: 1 1 0 1 0 1 1 1 0 0 0 0 1 1

G: 95 Cromossomo: 0 0 0 1 0 1 1 1 0 0 1 0 1 1

G: 96 Cromossomo: 1 0 0 0 0 1 1 1 1 0 1 1 1 1

G: 97 Cromossomo: 1 0 0 1 0 1 1 0 1 0 0 0 1 1

G: 98 Cromossomo: 1 1 0 0 0 1 1 1 1 0 1 1 1 1

G: 99 Cromossomo: 1 1 0 0 0 0 1 1 1 0 1 1 1 1

G: 100 Cromossomo: 1 1 0 0 0 0 0 1 1 0 1 0 1 1

Após a ordenação da função de avaliação, o fitness, escolher o primeiro que é o melhor fitness da população. O qual é a melhor solução do algoritmo genético.

Melhor Solução - G: Cromossomo: 0 1 1 1 1 0 1 1 0 1 0 1 0 1

Logo, esses produtos seriam transportados no caminhão para a loja física:

Nomes: Iphone 6 R$: 2911.12

Nomes: TV 55 R$: 4346.99

Nomes: Tv 50 R$: 3999.9

Nomes: TV 42 R$: 2999

Nomes: Ventilador Panasonic R$: 199.9

Nomes: Microondas Eletrolux R$: 308.66

Nomes: Microondas Panasonic R$: 299.29

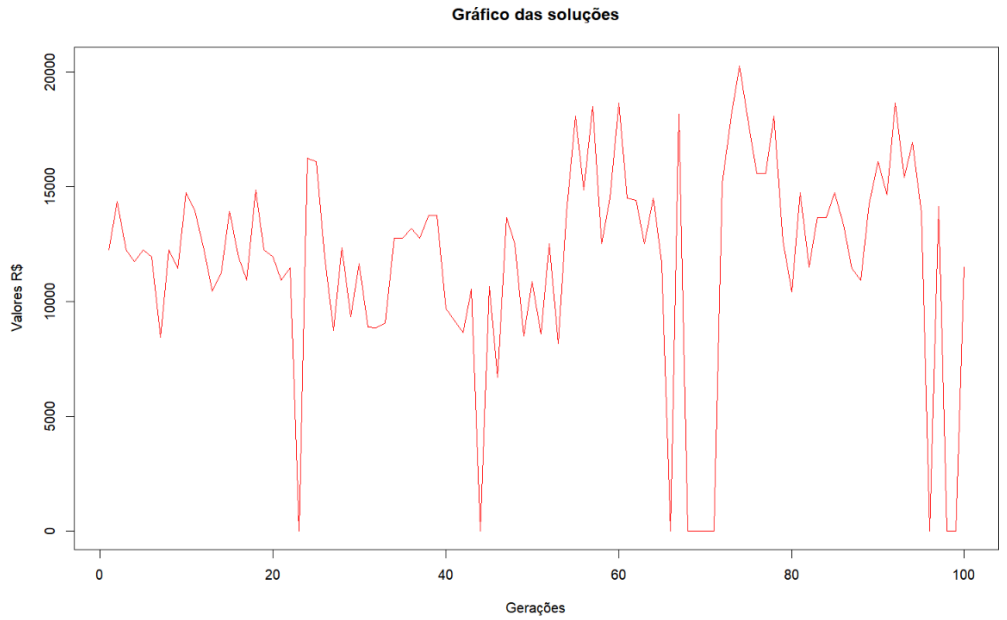
Nomes: Geladeira Consul R$: 1199.89

Nomes: Notebook Asus R$: 3999

Usando 2,8573899 m³ de espaço, ou seja, não está ultrapassando o limite de 3 m³ que foi colocado a princípio no problema.

Na figura abaixo podemos ver o gráfico da geração das soluções.

Figura 16-Gráfico de soluções



Fonte: Elaborada pela autora 2022