

# ALGORITMO GENÉTICO ESPECIALIZADO APLICADO AO PROBLEMA GENERALIZADO DE ATRIBUIÇÃO

Leonardo Luiz Vetore, Rubén Augusto Romero Lázaro.

Engenharia Elétrica – Departamento de Engenharia Elétrica – Faculdade de Engenharia de Ilha Solteira – Campus de Ilha Solteira.

## 1. Introdução

---

O problema generalizado de atribuição (Generalized Assignment Problem, GAP) é um problema clássico no campo de pesquisa operacional. Esse tipo de problema apresenta características altamente combinatórias e aparece em muitos problemas de otimização reais.

O problema generalizado de atribuição consiste basicamente em encontrar a estratégia de custo mínimo na atribuição de  $n$  tarefas para  $m$  agentes. Cada agente realiza uma tarefa com um tempo especificado e um custo conhecido, consequentemente cada agente possui um tempo limite para a realização dessas tarefas.

Consideramos a variável de decisão  $x_{ij}$  relacionada com a possibilidade de que o agente  $i$  faça a tarefa  $j$ . Assim, se  $x_{ij} = 1$  o agente  $i$  realiza a tarefa  $j$  e  $x_{ij} = 0$ , o agente não realiza a tarefa. Nessas condições, a modelagem do problema **GAP**, assume o seguinte formato:

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

$$\sum_{j=1}^n a_{ij} x_{ij} \leq b_i \quad i = 1, 2, \dots, m \quad (2)$$

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

$$x_{ij} \in \{0, 1\} \quad i = 1, 2, \dots, m; j = 1, 2, \dots, n \quad (4)$$

Sendo que  $a_{ij}$  é o tempo utilizado pelo agente  $i$  para realizar a tarefa  $j$  a um custo  $c_{ij}$  e  $b_i$  é o tempo disponível pelo agente  $i$ . A restrição (2), é uma restrição por agente que indica que o mesmo não pode ultrapassar o tempo disponível. Já a (3) é uma restrição por tarefa, que indica que a somatória de participações dos agentes para realizar a tarefa  $j$  deve ser igual a 1, ou seja, uma tarefa completa. Concluindo, a (4) exige que uma tarefa deva ser realizada por apenas um agente.

Problemas do tipo GAP estão sempre presentes como exemplo temos os seguintes casos: problema de localização de plantas com carga fixa, de programação de projetos de redes, problemas de redes de comunicação, de designação de tarefas em redes de computadores, entre outros.

Para resolver o problema GAP, utilizaremos uma metaheurística baseada em algoritmos evolutivos, o algoritmo genético desenvolvido por Chu-Beasley. Realizamos algumas modificações nesse algoritmo, assim com a calibração adequada dos parâmetros genéticos foi possível encontrar soluções melhores que as encontradas pelo algoritmo original de Chu-Beasley, e também foram encontradas soluções ótimas para problemas complexos conhecidos na literatura especializada.

## 2. Algoritmo Genético de Chu-Beasley

---

Neste item analisaremos os principais componentes do algoritmo genético desenvolvido por Chu-Beasley para solucionar o problema GAP

## 2.1 Codificação do problema

A proposta de codificação nada mais é do que a forma de representar uma solução. A figura 01 a seguir mostra um exemplo da codificação usada para o problema com  $m$  agentes e  $n$  tarefas. Cada tarefa ocupa sempre a mesma posição do vetor de decodificação e o elemento do vetor indica o agente que realiza a tarefa. Assim na decodificação a seguir, a tarefa 1 é realizada pelo agente 3, a tarefa 2 pelo agente 5, e assim por diante, até a tarefa  $n$  realizada pelo agente 1.

1	2	3	4	5	6	.....					n-1	n
3	5	4	2	6	1	...	6	2	5	m	1	

Figura 01: Proposta de codificação

## 2.2 Função Objetivo e Infactibilidades

Para cada proposta de solução pode ser calculada facilmente sua função objetivo, assim como as infactibilidades. Assim sendo, a função objetivo original (fitness) e a infactibilidade (unfitness) de cada proposta de solução é calculada e armazenada separadamente e utilizada para finalidades diferentes.

A função objetivo é usada para implementar a seleção e na substituição de um elemento da população quando todos os elementos da população são factíveis. A infactibilidade é usada para substituir um elemento da população quando existem propostas de solução infactíveis na população inicial.

## 2.3 Substituição da população

Consiste em substituir os elementos da população corrente por descendentes. Em algumas aplicações é utilizado o conceito de elitismo, ou seja, são preservadas algumas das propostas de solução, no caso as de melhor qualidade, especialmente a incumbente.

A proposta a ser apresentada consiste em substituir, em cada passo, apenas um elemento da população. O descendente gerado e candidato a substituir um elemento da população é incorporado a população utilizando a seguinte método estratégico:

- Se o descendente é infactível então somente pode substituir um elemento da população que seja mais infactível.
- Se o descendente é factível então, deve-se substituir o elemento da população mais infactível e se todos os elementos da população são factíveis então, deve-se substituir o elemento da população de pior qualidade caso o descendente seja de melhor qualidade. Além disto, o descendente gerado deve ser diferente de todos os elementos da população.

## 2.4 Seleção

A seleção utilizada é baseada em torneio com  $k = 2$ , onde  $k$  é o numero de topologias da população que vão participar de cada jogo. A seleção por torneio é uma das formas mais eficientes para realizar a seleção desde que o valor de  $k$  seja adequadamente escolhido. Os valores geralmente utilizados de  $k$  variam entre 2 e 3, no nosso caso 2.

## 2.5 Recombinação

A recombinação usada é de um ponto, isto é, escolhe-se aleatoriamente um ponto de recombinação e, gera-se o descendente com uma parcela de cada uma das topologias geradoras separadas pelo ponto de recombinação. No algoritmo de Chu-Beasley, gera-se apenas um descendente. Assim aleatoriamente e com a mesma probabilidade é escolhido apenas um descendente, portanto,

eliminando o outro descendente. A figura 02 a seguir mostra um exemplo de recombinação de um ponto em que o primeiro descendente é descartado.


Ponto de Recombinação												
1	2	3	4		5	6	7	8	9	10	11	12
3	1	4	2		3	1	2	4	2	3	2	1
1	2	2	3	1	4	1	3	3	2	4	1	
Descendente após a recombinação												
1	2	2	3	3	1	2	4	2	3	2	1	

Figura 02: Recombinação de um ponto

## 2.6 Mutação

A mutação proposta consiste na troca dos agentes que realizam duas tarefas. Assim aleatoriamente são escolhidas duas tarefas e assim os agentes que realizam essas tarefas são trocados. A figura 03 a seguir mostra um exemplo típico de mutação em que foram selecionadas aleatoriamente as tarefas 3 e 6.



1	2		3	4	5		6	7	8	9	10	11	12
1	2	2	3	3	1	2	4	2	3	2	1		
Descendente após a mutação													
1	2	1	3	3	2	2	4	2	3	2	1		

Figura 03: Mutação de duas tarefas trocando os agentes

## 2.7 Melhorando a infactibilidade

Se o descendente gerado é infactível então, tenta-se melhorar a infactibilidade desse descendente. Assim é analisado o recurso de cada agente na sequência de **1** a **m**. Caso o recurso de cada agente tenha seu recurso violado então, tenta-se repassar uma tarefa desse agente para outro agente que tenha a possibilidade de assumir essa tarefa sem violar sua capacidade.

## 2.8 Melhorando a função objetivo

Nesta ultima etapa, pretende-se melhorar a qualidade da função objetivo do descendente gerado. Assim é analisada a possibilidade de que cada tarefa seja transferida para outro agente com capacidade de realizar a tarefa com menor custo.

## 3. Resultados Encontrados

---

Foram realizados testes com o algoritmo desenvolvido e usando dados de problemas muito conhecidos na literatura especializada. Na tabela 1 mostramos resultados para problemas dos tipos A, B, C e D. Esses problemas são de complexidade variada.

Tabela 1: Resultados encontrados

Problema	Melhor Solução Conhecida	CGA Solução de Chu-Beasley	CGA Modificado
A5x100	1698	Melhor	Melhor
A5x200	3235	Melhor	Melhor
A10x100	1360	Melhor	Melhor
A10x200	2623	Melhor	Melhor
A20x100	1158	Melhor	Melhor
A20x200	2339	Melhor	Melhor
B5x100	1843	Melhor	Melhor
B5x200	3553	3601	
B10x100	1407	1410	
B10x200	2831	Melhor	Melhor
B20x100	1166	Melhor	Melhor
B20x200	2340	2347	
C5x100	1931	1941	
C5x200	3458	3460	
C10x100	1403	1463	
C10x200	2814	2815	
C20x100	1244	Melhor	Melhor
C20x200	2397	Melhor	Melhor
D5x100	6373	6479	6371
D5x200	12796	12823	12810
D10x100	6379	6390	6400
D10x200	12601	12634	12563
D20x100	6269	6280	
D20x200	12452	12471	

#### 4. Conclusão

---

O algoritmo encontrou resultados de melhor qualidade que o algoritmo original de Chu-Beasley, e em alguns casos os resultados encontrados foram melhores do que os da própria literatura especializada.