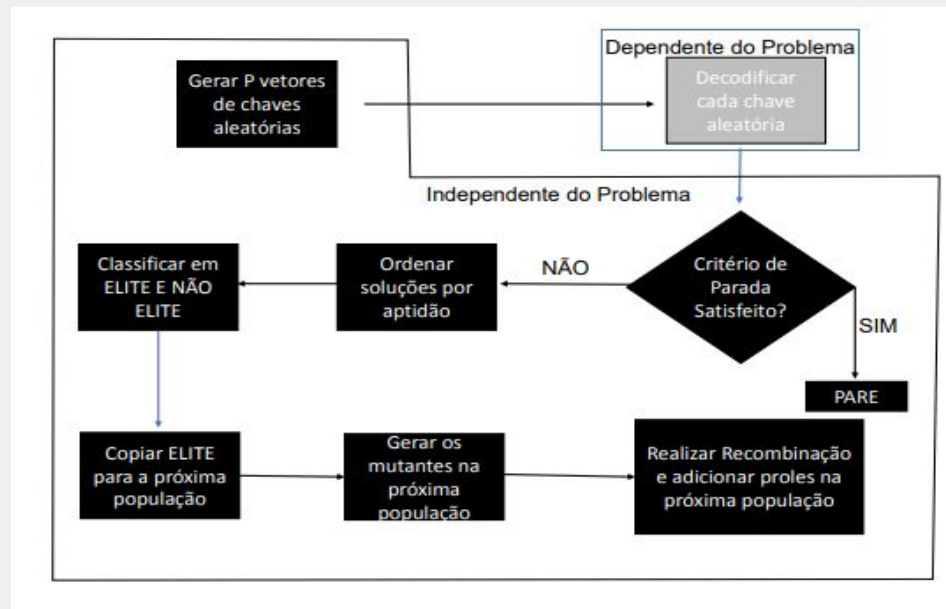


BRKGA

O BRKGA é uma metaheurística evolutiva utilizado para construção de heurísticas para resolver problemas de otimização combinatória.



BRKGA


- Gerar a população inicial
 - A população inicial é formada de p cromossomos, onde $p = n \cdot n$. Cada alelo é gerado de forma aleatória no intervalo real de $[0,1]$.
- Decodificador
 - O decodificador transforma os números aleatórios em soluções para o problema da atribuição. Se $\text{chave}[i,j] > 0.5 \Rightarrow x[i,j] = 1$; caso contrário $x[i,j] = 0$.
 - Depois, $x[i,j]$ passa pelas restrições e se atender a elas, a aptidão de cada cromossomo é calculada por sua função objetivo.

BRKGA

- Ordenar a solução
 - Cada solução é armazenada em uma lista que armazena as aptidões e o cromossomo que resultou naquela aptidão. E assim, ordena-se a lista de aptidões. E por fim, separa-se a população dessa lista, classificando em ELITE e NÃO ELITE.
- Cruzamento(parametrized uniform crossover)
 - Seleciona aleatoriamente um cromossomo-pai do subconjunto ELITE e um do subconjunto NAO ELITE. E o cruzamento é feito utilizando o vício probabilístico.
- Mutação
 - Esses mutantes são vetores de chaves aleatórias gerados da mesma forma que a população inicial (cada alelo do cromossomo mutante é gerado de forma aleatória no intervalo real de $[0,1]$)

BRKGA

Cruzamento



Cromossomo 1	0.46	0.91	0.33	0.75	0.51
Cromossomo 2	0.84	0.32	0.64	0.04	0.48
Número Aleatório	0.34	0.69	0.87	0.65	0.99
Vício de 0.7	<	<	>	<	>
Prole	0.46	0.91	0.64	0.75	0.48

BRKGA

- A próxima geração é formada:
 - A população elite é copiada para a próxima geração em sua íntegra.
 - Indivíduos resultantes do cruzamento entre membros da população elite e não elite.
 - Por fim, mutantes são gerados para compor o restante da próxima geração.

BRKGA

```
class BRKGA:
    def __init__(self, dados, tam_pop, max_gen, tx_cross, tx_mut, tx_elite):
        #Tamanho da população
        self.TAM_POP = int(tam_pop)
        #Critério de parada
        self.MAX_GEN = max_gen
        #Matriz de custos
        self.custos = np.array(dados.matriz)
        #Numero de genes em cada cromossomo
        self.TAM_CROM = int(dados.dim**2)
        #Taxa de cruzamento
        self.TX_CROSS = tx_cross
        #Taxa de mutação(Numero de mutantes criados)
        self.TX_MUT = tx_mut
        #Taxa da populacao que estara na populacao elite
        self.TX_ELITE = tx_elite
```