MINISTÉRIO DA EDUCAÇÃO
SECRETARIA DE EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA
INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA FLUMINENSE
CAMPUS CAMPOS CENTRO
CURSO DE ENGENHARIA DE COMPUTAÇÃO

Algoritmo Genético com Chave Aleatória Viciada (Biased Random Key Genetic Algorithm - BRKGA)

Alunos: Andryck Santiago, Danilo Taveira, Frederico de Souza, Heverton Souza, Júlia Baptista e Matheus Rinaldi

Professor: Philippe Leal

Disciplina: Projeto e Análise de Algoritmo

SUMÁRIO

1 Introdução

1.1 O que é uma metaheurística e por que é usada em problemas NP-difíceis

Uma metaheurística é um procedimento computacional que orienta outras heurísticas na busca por soluções de problemas de otimização, permitindo explorar o espaço de soluções além de ótimos locais. Segundo a definição encontrada na literatura, as metaheurísticas são métodos de solução que coordenam procedimentos de busca locais com estratégias de mais alto nível, de modo a criar um processo capaz de escapar de mínimos locais e realizar uma busca robusta no espaço de soluções de um problema.

As metaheurísticas se tornaram essenciais para resolver problemas NP-difíceis, que são problemas computacionalmente intratáveis para os quais não se conhece um algoritmo de tempo polinomial exato. Estes problemas são caracterizados como sendo "pelo menos tão difíceis quanto os problemas mais difíceis em NP". Em termos práticos, isso significa que quando o tamanho do problema aumenta, a complexidade para encontrar uma solução ótima cresce exponencialmente, tornando inviável a aplicação de métodos exatos em tempo hábil.

Os problemas NP-difíceis aparecem em diversas aplicações do mundo real, desde roteamento de veículos até problemas de agrupamento e otimização de recursos. Para estes casos, as metaheurísticas oferecem uma alternativa viável para encontrar soluções de boa qualidade em tempo computacional razoável, mesmo que não garantam a otimalidade da solução.

1.2 Diferença entre métodos exatos e heurísticos/metaheurísticas

A principal distinção entre métodos exatos e métodos heurísticos/metaheurísticos reside na garantia de otimalidade e no tempo computacional necessário para encontrar soluções.

Métodos exatos, como o Simplex e algoritmos de programação linear inteira, são capazes de encontrar a solução ótima de um modelo matemático. Eles

garantem que a solução encontrada seja a melhor possível, mas têm como desvantagem o alto custo computacional, especialmente para problemas de grande escala. Para problemas NP-difíceis, o tempo de processamento destes métodos pode tornar-se impraticável ou "infinito" em relação à aplicação desejada.

Métodos heurísticos e metaheurísticos, por outro lado, podem encontrar a solução ótima de um problema, mas não oferecem garantia de que a solução seja realmente ótima. Estes métodos são utilizados quando o problema é tão complexo que o tempo de processamento dos algoritmos exatos se torna incompatível com as necessidades práticas. As metaheurísticas, em particular, possuem estruturas flexíveis com componentes genéricos que podem ser adaptados ao problema específico, permitindo escolhas estratégicas de soluções piores que as já encontradas na tentativa de superar a otimalidade local.

Dentre os exemplos de metaheurísticas utilizados para resolver problemas NP-Difíceis está o Algoritmo Genético com Chave Aleatória Viciada (Biased Random Key Genetic Algorithm - BRKGA).

2 Conceitos

2.1 Origem do algoritmo

O BRKGA é uma variação especializada, introduzida por Gonçalves e Resende (2011) como uma evolução do modelo de *Random-Key Genetic Algorithm* (RKGA) proposto anteriormente por Bean (1994). Ele se enquadra na família de algoritmos genéticos populacionais, utilizando os conceitos fundamentais da evolução biológica - como seleção, cruzamento e mutação - para resolver problemas de otimização combinatória. Sua principal característica distintiva é o uso de chaves aleatórias (random keys) para representar soluções, onde cada solução é codificada como um vetor de números reais no intervalo *[0,1)*. Essa representação numérica permitiu simplificar o processo de manipulação genética, tornando-o mais adequado para problemas complexos e facilitando a implementação de operadores de recombinação e mutação.

A proposta do BRKGA surgiu como uma resposta à necessidade de melhorar a convergência e eficiência do RKGA. A principal inovação foi a introdução de um viés probabilístico no operador de cruzamento (*crossover*), de forma a favorecer a propagação de genes oriundos de indivíduos de maior aptidão (*elite*). Essa modificação aumentou a taxa de herança de características de soluções de alta qualidade, acelerando o processo de busca sem comprometer a diversidade genética da população.

Desde a sua concepção, o BRKGA tem sido aplicado em diversos problemas de otimização, incluindo escalonamento, roteamento, alocação de recursos e problemas de programação inteira, sendo reconhecido pela simplicidade de implementação e desempenho competitivo em relação a métodos mais sofisticados.

2.2 Diferença para Algoritmos Genéticos Tradicionais

Os *algoritmos genéticos tradicionais* (AGs) utilizam representações discretas ou binárias para codificar as soluções (por exemplo, cadeias de bits). O BRKGA, por sua vez, trabalha com representações contínuas na forma de *random keys*, o que lhe confere algumas diferenças fundamentais:

- Representação da solução:
    - *AG tradicional*: cada indivíduo é representado como uma sequência de genes codificados de forma discreta (binária ou inteira).
    - *BRKGA*: cada indivíduo é um vetor de números reais uniformemente distribuídos no intervalo *[0,1)*, independentemente do problema. A interpretação desses valores é feita por um decodificador que converte as chaves em uma solução viável.
- Dependência do decodificador
    - No BRKGA, o núcleo do algoritmo é independente da natureza do problema; apenas o decodificador muda para cada aplicação. Já nos AGs tradicionais, a codificação e os operadores genéticos costumam estar diretamente relacionados ao formato da solução.
- Crossover com viés

    - Em AGs tradicionais, o cruzamento entre dois pais tende a ser equilibrado, com probabilidade semelhante para cada gene ser herdado de um dos pais.
    - No BRKGA, o crossover é parcialmente direcionado, favorecendo a herança de genes do indivíduo mais apto (pai *elite*) com uma probabilidade fixa *p>0.5*, enquanto o outro pai é escolhido da população não-elite.
- Controle da diversidade
    - O BRKGA mantém a diversidade naturalmente ao selecionar o segundo pai de fora do conjunto de elite, evitando convergência prematura.
    - Nos AGs tradicionais, a diversidade é muitas vezes mantida apenas por operadores de mutação ou por estratégias adicionais.

Essas diferenças tornam o BRKGA particularmente adequado para problemas onde a representação contínua e a interpretação via decodificador oferecem maior flexibilidade e eficiência.

2.3 Ideia Principal: Random Keys e Viés no Crossover

O conceito central do BRKGA está na utilização de random keys e no viés controlado no processo de cruzamento.

a) Random Keys (chaves aleatórias):

As soluções são representadas por vetores de números reais no intervalo [0,1). Esses vetores não correspondem diretamente à solução do problema, mas funcionam como uma codificação genérica que será posteriormente interpretada por um decodificador. Essa abordagem permite aplicar o mesmo algoritmo a diferentes problemas de otimização, alterando apenas o decodificador.

b) Viés no Crossover (cruzamento enviesado):

O processo de recombinação privilegia os indivíduos mais aptos (elite). Ao gerar um novo indivíduo, as características do pai elite têm maior probabilidade de

serem herdadas em relação ao outro pai, que é escolhido fora da elite. Esse viés acelera a convergência para boas soluções, ao mesmo tempo em que mantém a diversidade.

Esses dois elementos (representação por random keys e cruzamento enviesado) formam a base do BRKGA. A implementação detalhada desses mecanismos será apresentada no capítulo seguinte.

3 Componentes do BRKGA

3.1 Codificação da solução (random keys)

Para entender a codificação do problema é necessário conhecer os conceitos utilizados para modelar o problema em uma metaheurística. Cada solução candidata é representada por um indivíduo, e o conjunto desses indivíduos forma a população.

Um indivíduo possui um genoma, composto por variáveis que o descrevem. Cada variável corresponde a um gene (ou cromossomo), e o valor associado a esse gene é chamado de alelo.

No caso específico do BRKGA, cada alelo não é um valor discreto, mas sim um número real gerado de forma aleatória dentro do intervalo [0,1), conhecido como random key. Assim, cada indivíduo é representado por um vetor de números reais, que não corresponde diretamente à solução final do problema, mas sim a uma codificação abstrata.

A interpretação dessas random keys é feita por meio de um decodificador, que traduz o vetor em uma solução viável para o problema em questão. Essa abordagem traz flexibilidade, pois o núcleo do algoritmo continua o mesmo para qualquer aplicação — apenas o decodificador precisa ser adaptado para cada tipo de problema.

3.2 População inicial (geração aleatória, tamanho)

A população inicial é formada por um conjunto de indivíduos gerados com genes aleatórios compreendidos no intervalo real [0 e 1) e serão avaliados posteriormente.

No entanto deve-se tomar cuidado quanto à quantidade de indivíduos na população de um algoritmo genético, pois caso seja pequena pode gerar uma solução que seja um ótimo local, ou seja, ela atende muito bem as demandas do problemas dentro de um contexto mais limitado, se comparado à realidade, o que não será o resultado mais satisfatório e próximo do ideal para a situação real.

Por outro lado, também é necessário manter um certo limite na quantidade de indivíduos na população, para garantir que não seja necessária uma operação muito extensa para alcançar um resultado satisfatório. O ideal é sempre se aproximar dos ótimos globais, pois eles indicam as melhores soluções possíveis para a maioria dos casos do problema.

Então, depois de gerados, os indivíduos serão avaliados pela função de avaliação(ou *fitness function*), que atribui uma medida de qualidade a cada solução candidata, servindo como critério de seleção dentro da população inicial.

3.3 Função objetivo (avaliação da qualidade das soluções)

Cada indivíduo, representado por um vetor de random keys, não pode ser comparado diretamente. Para isso, ele passa por um decodificador, que traduz as chaves em uma solução viável para o problema estudado.

Em seguida, aplica-se a função objetivo(função de avaliação), que mede a qualidade dessa solução de acordo com os critérios do problema (por exemplo: minimizar tempo, custo, distância ou maximizar lucro, eficiência, etc.).

Os indivíduos com melhor desempenho na função de avaliação são considerados mais aptos, enquanto os de pior desempenho são menos relevantes. Esse processo é fundamental, porque é a base de todas as etapas posteriores à seleção e reprodução no BRKGA.

3.4 Classificação das soluções (ordenação por qualidade)

Uma vez calculados os valores da função de avaliação para cada indivíduo, a população é ordenada de acordo com a qualidade das soluções. Essa ordenação permite identificar claramente quais indivíduos possuem maior aptidão.

Esse processo de ranqueamento é essencial, pois dele depende a formação do grupo de elite (os melhores indivíduos), o grupo não-elite (indivíduos medianos ou piores) e até a proporção de mutantes que serão introduzidos. Sem essa classificação, o algoritmo não teria como aplicar a seleção enviesada característica do BRKGA.

3.5 Divisão em grupos (elite, não-elite, mutantes)

Após a ordenação, a população é dividida em três subconjuntos:

- Elite: composto pelos indivíduos de melhor desempenho. São eles que mais influenciam na geração da nova população, garantindo a preservação das boas características encontradas.
- Não-elite: formados pelos demais indivíduos da população. Eles são importantes para manter a diversidade, pois permitem que soluções alternativas também participem do processo de recombinação.
- Mutantes: indivíduos gerados completamente de forma aleatória a cada nova geração. A presença dos mutantes é um diferencial importante, pois evita que o algoritmo fique preso em ótimos locais, reinjetando variabilidade no processo evolutivo.

3.6 Crossover enviesado (como funciona a probabilidade p)

O crossover é o operador responsável por gerar novos indivíduos (filhos) a partir da combinação de dois pais. No BRKGA, esse cruzamento é enviesado:

- Um pai é escolhido do grupo elite.
- O outro pai é escolhido do grupo não-elite.
- Para cada gene do filho, existe uma probabilidade p (tipicamente 0.7) de herdar o valor do pai elite, e probabilidade $1 – p$ de herdar do pai não-elite.

Esse mecanismo garante que boas características (do grupo elite) tenham maior chance de serem transmitidas, acelerando a convergência do algoritmo, mas ainda mantendo diversidade por meio do segundo pai e dos mutantes.

É importante notar que no caso dos crossovers, ainda se faz necessário cruzar indivíduo-elite com indivíduo não-elite para garantir a diversidade da

população e evitar ótimos locais. Entretanto, o objetivo dessa diversificação é aumentar as possibilidades de soluções globais, mas visando manter a avaliação do indivíduo em um nível que o classifique como elite. Assim, é possível evitar ótimos locais e diversificar as soluções.

3.7 Mutantes (geração de soluções novas)

Os mutantes são codificados como vetores de *random keys* gerados de forma totalmente aleatória e têm dois papéis principais:

- Aumentar a diversidade genética na população.
- Explorar novas regiões do espaço de busca, possibilitando a descoberta de soluções que não surgiriam apenas pelo cruzamento de elites e não-elites.

Normalmente, a quantidade de mutantes é definida por um parâmetro Pm, representando uma proporção do tamanho da população total.

3.8 Critério de parada (gerações, tempo, estagnação)

O algoritmo é iterativo, executando ciclos de seleção, cruzamento enviesado, inserção de mutantes e avaliação. O critério de parada define quando interromper a execução. Os mais comuns são:

- Número máximo de gerações (G_{max}): interrompe após executar um número predefinido de ciclos.
- Tempo máximo de execução: útil em aplicações onde há restrição temporal rígida.
- Estagnação: para quando não há melhoria no valor da melhor solução encontrada por um número consecutivo de gerações (G_{stall}).
- Solução satisfatória: encerra quando é atingido um valor mínimo aceitável da função objetivo.

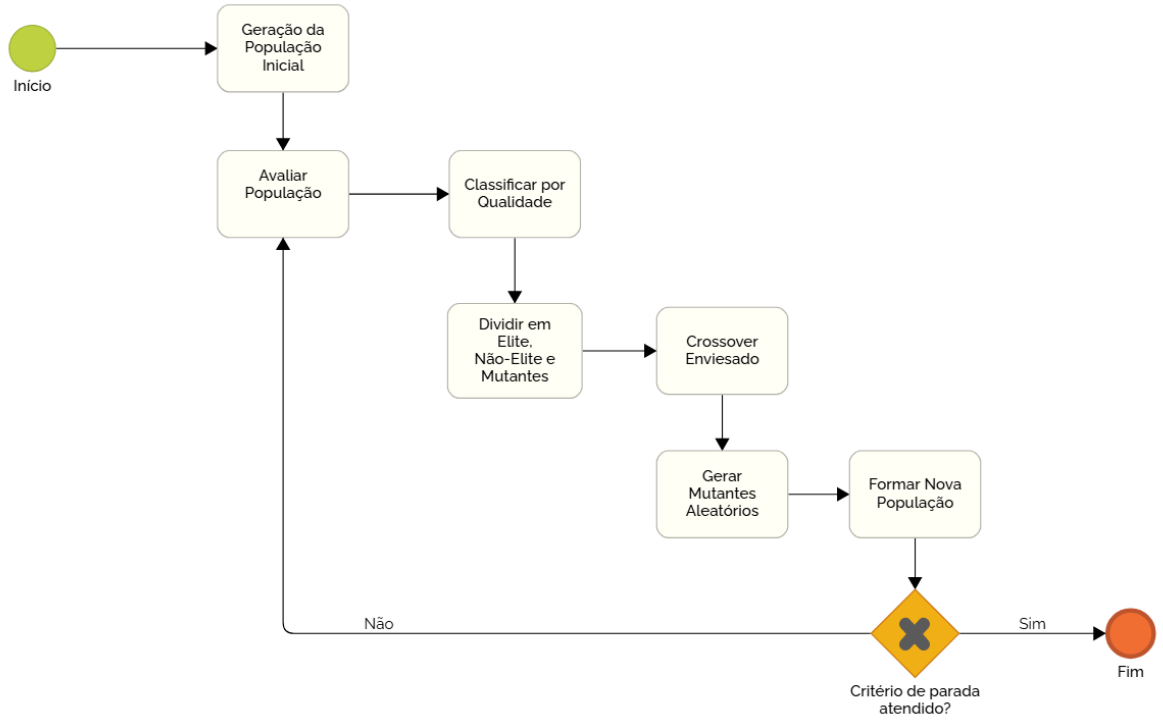
A escolha do critério depende do equilíbrio desejado entre tempo de execução e qualidade da solução

4. Passo a passo do algoritmo

O funcionamento básico do BRKGA pode ser descrito assim:

1. Inicialização
    - Gerar a população inicial de forma aleatória (cada indivíduo é um vetor de *random keys*).
    - Avaliar cada indivíduo com a função objetivo.
2. Classificação
    - Ordenar a população de acordo com o valor da função objetivo (melhor para pior).
3. Divisão em grupos
    - Elite: os melhores $P_e \times n$ indivíduos.
    - Não-Elite: o restante da população, excluindo mutantes.
    - Mutantes: novos indivíduos gerados aleatoriamente.
4. Crossover enviesado
    - Para cada novo indivíduo da próxima geração:
        - Escolher um pai da elite e outro não-elite.
        - Para cada gene (*random key*), herdar do pai da elite com probabilidade P_c e do não-elite com probabilidade $1 - P_c$
5. Inserção de mutantes
    - Adicionar $P_m \, x \, n$ novos indivíduos aleatórios.
6. Substituição
    - A nova população substitui a antiga.
7. Verificação do critério de parada
    - Se atendido, o algoritmo encerra; caso contrário, retorna ao passo 2.

Fluxograma 1: Fluxo de execução do BRKGA



Fonte: os autores

5 Vantagens e Limitações

5.1 Vantagens do método

O BRKGA apresenta diversas vantagens significativas que o tornam uma escolha atrativa para resolver problemas de otimização combinatória complexos sendo eles listados abaixo:

- Simplicidade de implementação e generalização: Uma das principais vantagens do BRKGA é sua natureza altamente desacoplada, sendo pouco dependente de partes específicas do problema. Isso significa que a maior parte do algoritmo pode ser reutilizada para diferentes problemas, necessitando apenas da implementação de um decodificador específico. Esta característica confere ao algoritmo uma "natureza genérica e

independente do problema específico", permitindo sua fácil adaptação a uma ampla variedade de problemas de otimização.

- Eficiência superior a outros algoritmos genéticos: Estudos empíricos demonstram que o BRKGA é mais eficiente na busca por soluções quando comparado ao RKGA tradicional. Os resultados indicaram que o BRKGA é mais eficiente na busca por soluções que o RKGA, considerando o conjunto de instâncias avaliado. A melhoria na qualidade da solução produzida foi, em média, superior a 20%".
- Estratégia de seleção enviesada: O BRKGA implementa uma estratégia de seleção que sempre escolhe um dos pais do conjunto elite para o cruzamento, enquanto o outro pode ser selecionado de qualquer parte da população. Esta abordagem contribui para a tendência do algoritmo de preservar e propagar as características das soluções mais promissoras ao longo do processo de evolução.
- Facilidade de evitar soluções inviáveis: Ao contrário dos algoritmos genéticos tradicionais, que necessitam de procedimentos especiais de reparo para lidar com permutações ou sequências, o BRKGA move todas as questões de viabilidade para o procedimento de avaliação objetiva e garante que toda a prole formada pelo cruzamento seja composta por soluções viáveis.
- Robustez e diversidade populacional: O algoritmo mantém diversidade através da introdução sistemática de mutantes (indivíduos gerados aleatoriamente) a cada geração, evitando convergência prematura e permitindo exploração contínua do espaço de soluções.

5.2 Limitações do método

Apesar de suas vantagens, o BRKGA também apresenta limitações importantes que devem ser consideradas quanto a sua aplicabilidade, sendo elas listadas abaixo:

- Dependência da qualidade do decodificador: A eficácia do BRKGA está intrinsecamente ligada à qualidade da implementação do decodificador, que é a única parte do algoritmo dependente do problema específico. Um decodificador mal projetado pode comprometer significativamente o

desempenho do algoritmo, tornando-o ineficiente mesmo com uma boa estrutura evolutiva.

- Maior tempo de execução: Embora o BRKGA produza soluções de melhor qualidade, estudos mostram que BRKGA também demonstrou exigir um tempo maior de execução. O tempo total foi, em média, mais de 16% superior quando comparado ao RKGA. Esta limitação pode ser crítica em aplicações com restrições rigorosas de tempo.
- Sensibilidade aos parâmetros: Como qualquer algoritmo genético, os parâmetros adotados constituem um ponto crítico dessa abordagem. A escolha inadequada de parâmetros como tamanho da população, tamanho da elite, número de mutantes e probabilidade de herança pode afetar significativamente o desempenho do algoritmo.
- Ausência de garantia de otimalidade: Como toda metaheurística, o BRKGA não oferece garantia de que a solução encontrada seja ótima global. Embora possa encontrar soluções de alta qualidade, sempre existe a possibilidade de que soluções melhores existam no espaço de busca.
- Necessidade de ajuste específico para cada problema: Embora o framework seja genérico, "estudos mais amplos são necessários para confirmar se a combinação de valores adotados é, de fato, a opção mais interessante para a resolução do problema abordado". Isso implica que cada aplicação pode exigir um processo de calibração específica dos parâmetros para obter o desempenho ideal.

6 O Problema da Mochila

Uma das aplicações do BRKGA está no problema da Mochila, um clássico caso de estudo da otimização combinatória devido à sua simplicidade conceitual, ampla aplicabilidade prática e elevada complexidade computacional.

6.1 Descrição do problema

Imagine que você vai fazer uma viagem e deseja colocar seus pertences em uma mochila. A mochila tem capacidade limitada, de modo que nem todos os

objetos podem ser levados. Logo, você precisa selecionar quais itens levar, considerando:

- O peso de cada item (quanto espaço ou carga ele ocupa), e
- O valor de cada item (quanto ele “vale” para você).

O objetivo é maximizar o valor total dos itens levados, respeitando a restrição de capacidade da mochila.

Existem diversas variações deste problema, como:

- Problema da Mochila Binária (Mochila 0-1): cada item pode ser colocado na mochila no máximo uma vez e não pode ser fracionado;
- Mochila com Capacidade Ilimitada: é permitido colocar várias unidades de cada item;
- Mochila Limitada: é permitido colocar um número limitado, porém maior que 1, de cada item;
- Entre outras variações.

Neste trabalho, será contemplado o Problema da Mochila Binária.

6.2 Formulação matemática

Considere uma mochila de capacidade $W > 0$ e lista de *n* elementos, cada qual com:

- Um peso ρ_i
- Um valor ω_i
- Uma variável binária x_i:
    - $1,\ se\ o\ item\ i\ for\ colocado\ na\ mochila$
    - $0,\ se\ não\ for\ colocado$

O problema pode ser formulado como:

$$max \sum_{i=1}^{n} \omega_i x_i,\ sujeito\ a \sum_{i=1}^{n} \rho_i x_i \leq W$$

Onde:

- A função objetivo busca maximizar o valor total dos itens escolhidos;
- A restrição garante que o peso total dos itens selecionados não exceda a capacidade da mochila;
- A variável binária define se o item será levado ou não.

6.3 Complexidade Computacional

Uma abordagem ingênua para resolver o problema seria avaliar todas as combinações possíveis de itens. Porém, como cada item pode estar ou não na mochila, o número de combinações é 2^n. Isso significa que o tempo de execução cresce exponencialmente com o número de itens, tornando-se impraticável para grandes valores de nnn.

Essa característica faz com que o Problema da Mochila Binária seja classificado como NP-difícil (NP-hard).

6.4 Aplicações do Problema da Mochila

Apesar de sua simplicidade, o Problema da Mochila aparece em diversos contextos reais, como:

- Gestão de investimentos: Selecionar ativos que maximizem o retorno esperado respeitando um orçamento máximo;
- Transporte e logística: Organizar a carga de caminhões, navios ou contêineres maximizando o valor transportado sem exceder o peso ou volume permitido;
- Seleção de projetos: Escolher projetos a serem financiados maximizando o benefício total dentro de um orçamento limitado;
- Planejamento de armazenamento: Alocar produtos em um espaço de estoque restrito maximizando o valor total dos itens armazenados.

7 BRKGA aplicado ao Problema da Mochila

Como apresentado anteriormente, no BRKGA, cada solução é representada por um vetor de chaves aleatórias (random keys), que são números reais no

intervalo [0,1). No BRKGA aplicado à mochila, cada chave está associada a um item da mochila. Essa representação não indica diretamente se o item será incluído ou não, mas define a prioridade de seleção: quanto menor o valor da chave, maior a prioridade do item **(representação da solução)**.

A partir dos conceitos apresentados nesse texto, será ilustrada aplicação do BRKGA em um exemplo do problema da mochila:

Considere uma instância com a capacidade da mochila $W = 10$ e os seguintes itens disponíveis:

Item	Peso p_i	Valor v_i
1	4	10
2	3	8
3	2	5
4	5	12
5	1	4

Parâmetros do BRKGA (para este exemplo):

Tamanho da População	6 cromossomos (vetores de chaves aleatórias)
Elite	2 melhores ($\approx 33\%$)
Mutantes por geração	1 ($\approx 17\%$)
Viés do Crossover	$p = 0,7$
Critério de parada	3 gerações

1. Inicialização: Será gerada a população inicial de forma aleatória e cada indivíduo será avaliado de acordo com a função objetivo do problema da mochila.
    a. Geração dos indivíduos e ordenação das chaves aleatórias.

Indivíduo	Cromossomo	Ordem por chave
A	[0.97, 0.50, 0.58, 0.71, 0.81]	2, 3, 4, 5, 1

B	[0.04, 0.83, 0.84, 0.59, 0.82]	1, 4, 5, 2, 3
C	[0.82, 0.74, 0.27, 0.09, 0.12]	4, 5, 3, 2, 1
D	[0.82, 0.70, 0.36, 0.41, 0.37]	3, 5, 4, 2, 1
E	[0.99, 0.46, 0.81, 0.40, 0.72]	4, 2, 5, 3, 1
F	[0.08, 0.86, 0.57, 0.23, 0.72]	1, 4, 3, 5, 2

b. Avaliar cada indivíduo com a função objetivo:

Seguindo a ordenação dos cromossomos realizados acima, será aplicada a função objetivo para cada indivíduo, encontrando o valor total (Fitness Value) de cada um. (2, 3, 4, 5, 1)

- Para o indivíduo A:
    - Inclui item 2 → peso acumulado = 3.
    - Inclui item 3 → peso acumulado = 5.
    - Inclui item 4 → peso acumulado = 10.
    - Item 5 não pode ser incluído (peso 1 → ultrapassa 10).
    - Item 1 não pode ser incluído (peso 4 → ultrapassa 10).
  - Solução final: {2, 3, 4}, valor total = 25.
- Repetindo o mesmo processo para todos os indivíduos obtemos:

Indivíduo	Solução	Valor total
A	{2,3,4}	25
B	{1,4,5}	26
C	{4,5,3}	21
D	{3,5,4}	21
E	{4,2,5}	24
F	{1,4,5}	26

2. Classificação e Divisão em Grupos: Será ordenada a população de acordo com o valor da função objetivo, dividindo os grupos em elite e não elite.

Indivíduo	Valor Total	Classificação
B	26	Elite

F	26	Elite
A	25	Não-Elite
E	24	Não-Elite
C	21	Não-Elite
D	21	Não-Elite

3. Crossover enviesado e inserção de mutantes: A vizinhança no BRKGA é gerada pelos operadores evolutivos do algoritmo. Dessa forma, cada nova geração corresponde a um conjunto de soluções vizinhas, que evoluem progressivamente em direção a combinações de maior valor para o problema da mochila. Neste caso, a nova vizinhança será formada pelos os indivíduos elite dessa geração, 1 mutante e 3 filhos por crossover enviesado.
    a. Exemplo de crossover enviesado:
        i. Pai elite: B = (0.04, 0.83, 0.84, 0.59, 0.82)
        ii. Pai não-elite: D = (0.82, 0.70, 0.36, 0.41, 0.37)
        iii. Filho gerado: (0.04, 0.83, 0.36, 0.59, 0.37)
    b. Exemplo mutante: uma nova chave aleatória é criada do 0
        i. Mutante: (0.91, 0.73, 0.66, 0.10, 0.81)
4. Substituição: Após essas etapas, a nova geração será semelhante à:

Indivíduo	Cromossomo	Ordem por chave
B (elite)	[0.04, 0.83, 0.84, 0.59, 0.82]	1, 4, 5, 2, 3
F (elite)	[0.08, 0.86, 0.57, 0.23, 0.72]	1, 4, 3, 5, 2
Filho 1 (BxD)	[0.04, 0.83, 0.36, 0.59, 0.37]	1, 3, 5, 4, 2
Filho 2 (BxA)	[0.04, 0.83, 0.84, 0.59, 0.82]	1, 4, 5, 2, 3
Filho 3 (FxE)	[0.08, 0.50, 0.58, 0.23, 0.72]	1, 4, 2, 3, 5
Mutante 1	[0.91, 0.73, 0.66, 0.10, 0.81]	4, 3, 2, 5, 1

5. Verificação do critério de parada: Após essas etapas, será verificado se o critério de parada foi atendido, caso sim, o programa será encerrado, e o resultado será o indivíduo de maior valor encontrado na última geração

calculada. Caso contrário, o código voltará ao passo 1.b (avaliando cada indivíduo com a função objetivo).

O resultado obtido ao longo das gerações tende a se concentrar em soluções com maior valor, até que o critério de parada seja atingido. No exemplo estudado, as melhores soluções atingiram valores de 26 unidades, o qual potencialmente aumentaria nas gerações seguintes. Isso indica que o BRKGA é uma abordagem robusta e eficaz, sendo capaz de produzir soluções de alta qualidade de forma eficiente para problemas complexos, como o Problema da Mochila. Essa análise reforça o potencial das metaheurísticas como alternativa viável em cenários onde métodos exatos se tornam inviáveis.

8 Referências

ALGORITMO genético de chaves aleatórias viciadas especializado para o problema de corte bidimensional não guilhotinado. Colloquium Exactarum, [S. l.], v. 14, n. 1, p. 136–145, 2023. ISSN 2178-8332. Disponível em: https://journal.unoeste.br/index.php/ce/article/view/4517. Acesso em: 3 ago. 2025.

BEAN, J. C. Genetic algorithms and random keys for sequencing and optimization. ORSA Journal on Computing, v. 6, n. 2, p. 154–160, 1994.

CARVALHO, Rubens. Problema da Mochila. 2015. 14 f. Relatório (Projeto Supervisionado I – disciplina MS777) – Instituto de Matemática, Estatística e Computação Científica, Universidade Estadual de Campinas, Campinas, 2015.

DEVELOPER ACADEMY PUC-Rio. Introdução ao Algoritmo Genético – Parte 1. [S. l.: s. n.], 2019. 1 vídeo (5 min 52 s). Publicado no canal Developer Academy PUC-Rio, 10 jan. 2019. Disponível em: https://www.youtube.com/watch?v=xtHMSJLnKsE&ab_channel=DeveloperAcademyPUC-Rioo. Acesso em: 4 ago. 2025

DEVELOPER ACADEMY PUC-Rio. Introdução ao Algoritmo Genético – Parte 2. [S. l.: s. n.], 2019. 1 vídeo (13 min 24 s). Publicado no canal Developer Academy PUC-Rio, 10 jan. 2019. Disponível em: https://www.youtube.com/watch?v=96mlWTAvjik&t=3s&ab_channel=DeveloperAcademyPUC-Rio. Acesso em: 4 ago. 2025.

GOLDBARG, Marco César; GOLDBARG, Elizabeth Wanner. Metaheurísticas: da teoria à prática. Rio de Janeiro: Elsevier, 2012.

GOLDBARG, Marco César; LUNA, Henrique Pacca. Otimização combinatória e programação linear: modelos e algoritmos. Rio de Janeiro: Elsevier, 2005.

GONÇALVES, J. F.; RESENDE, M. G. C. Biased random-key genetic algorithms for combinatorial optimization. Journal of Heuristics, v. 17, p. 487–525, 2011.

MAURÍCIO, Leandro. Estudo sobre as Metaheurísticas. São Paulo: Instituto de Matemática e Estatística, Universidade de São Paulo, 2009.

MOREIRA, Thiago da Silva. Meta-heurística BRKGA-QL aplicada em um problema de agrupamento com restrições semi-supervisionado. São Paulo: Universidade Federal de São Paulo, 2021.

PUTTI, Eduardo Paz; OLIVEIRA, Nathan Bettisteli; SCHMIDT, Carise Elisane. Algoritmos genéticos com chaves aleatórias. Revista Científica.

RESENDE, Mauricio G. C.; GONÇALVES, José Fernando; COSTA, Miguel Dias. A biased random-key genetic algorithm for the minimization of open stacks problem. 2013. Disponível em: https://arxiv.org/abs/1309.2558. Acesso em: 17 ago. 2025.

SILVA, Gustavo Peixoto. Metaheurísticas. Ouro Preto: Universidade Federal de Ouro Preto, Departamento de Computação, 2010.

SOBRAL, Luís. Teoria da decisão: introdução às metaheurísticas. Belo Horizonte: Universidade Federal de Minas Gerais, Programa de Pós-Graduação em Engenharia Elétrica, 2010.