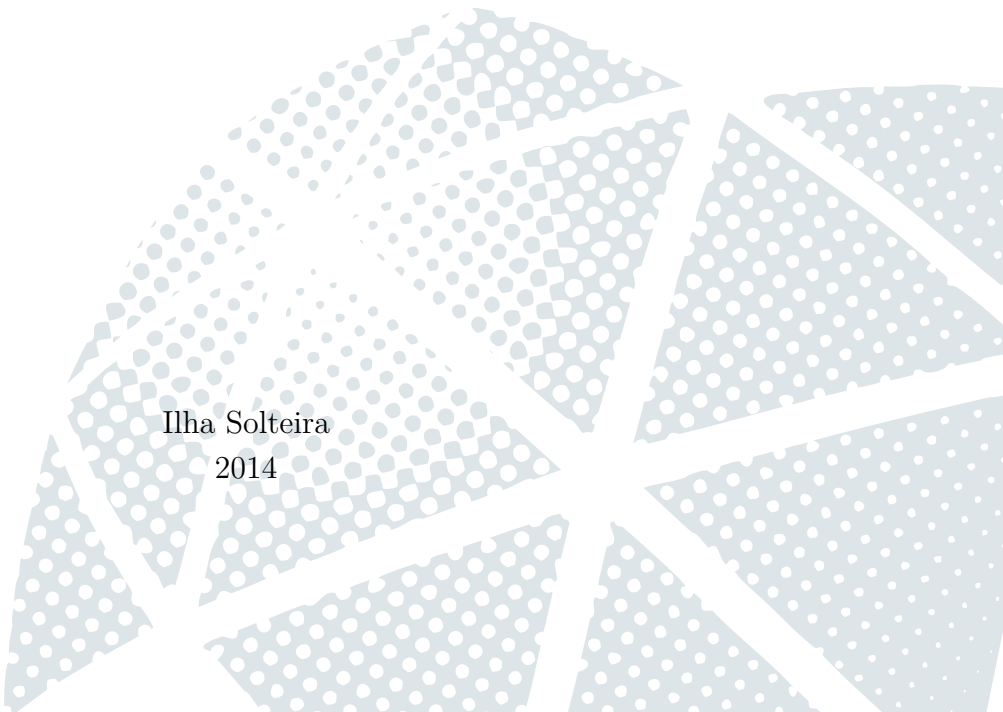
UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Campus de Ilha Solteira

Henrique Uzinski

Otimização de problemas multimodais usando meta-heurísticas evolutivas



Ilha Solteira
2014

UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Campus de Ilha Solteira

PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

Otimização de problemas multimodais usando meta-heurísticas evolutivas

Henrique Uzinski

Orientador: Dr. Rubén Augusto Romero Lázaro

Dissertação apresentada à Faculdade de Engenharia do Campus de Ilha Solteira - UNESP, como parte dos requisitos para obtenção do título de Mestre em Engenharia Elétrica.

Área de Conhecimento: Automação.

Ilha Solteira
2014

FICHA CATALOGRÁFICA
Desenvolvido pelo Serviço Técnico de Biblioteca e Documentação

U99o

Uzinski, Henrique.
Otimização de problemas multimodais usando meta-heurísticas evolutivas / Henrique Uzinski. -- Ilha Solteira: [s.n.], 2014
81 f. : il.

Dissertação (mestrado) - Universidade Estadual Paulista. Faculdade de Engenharia de Ilha Solteira. Área de conhecimento: Automação, 2014

Orientador: Rubén Augusto Romero Lázaro
Inclui bibliografia

1. Meta-heurísticas. 2. Algoritmo Genético Tradicional. 3. Algoritmo Genético de Chu-Beasley. 4. Algoritmo Genético de Chaves Aleatórias Viciadas.

unesp

UNIVERSIDADE ESTADUAL PAULISTA

CAMPUS DE ILHA SOLTEIRA

FACULDADE DE ENGENHARIA DE ILHA SOLTEIRA

CERTIFICADO DE APROVAÇÃO

TÍTULO: Otimização de problemas multimodais usando meta-heurísticas evolutivas

AUTOR: HENRIQUE UZINSKI

ORIENTADOR: Prof. Dr. RUBEN AUGUSTO ROMERO LAZARO

Aprovado como parte das exigências para obtenção do Título de Mestre em Engenharia Elétrica , Área: AUTOMAÇÃO, pela Comissão Examinadora:

Prof. Dr. RUBEN AUGUSTO ROMERO LAZARO
Departamento de Engenharia Elétrica / Faculdade de Engenharia de Ilha Solteira

Profa. Dra. MARINA LAVORATO DE OLIVEIRA
Departamento de Engenharia Elétrica / Faculdade de Engenharia de Ilha Solteira

Prof. Dr. MARCELO ESCOBAR DE OLIVEIRA
Instituto Federal de Educação, Ciência e Tecnologia de Goiás, Campus Itumbiara

Data da realização: 24 de outubro de 2014.

Dedico a Deus e à minha família.

AGRADECIMENTOS

Agradeço a Deus pelas bençãos, sabedoria e conhecimentos adquiridos durante esta jornada.

Ao professor Dr. Rubén Augusto Romero Lázaro pelo apoio, confiança e orientação deste trabalho. Aos professores Dra. Marina Lavorato de Oliveira e Dr. Marcelo Escobar de Oliveira, pelas correções e sugestões na banca de defesa. Agradecimentos aos professores Dr. Anna Diva Plasencia Lotufo e ao Dr. John Fredy Franco Baquero, pelas correções e sugestões na banca de qualificação. E a todos funcionários da UNESP, pela amizade e contribuição.

Aos meus pais Julio Uzinski e Maria Aparecida de Oliveira Uzinski, meus irmãos Julio Cezar Uzinski, Tarcisio Uzinski e Rafael de Oliveira Uzinski, pelo apoio e confiança.

Aos professores da UNEMAT, Diego Piasson, Daise Lago Pereira Souto, Marcia Cristina Dal Toé, Emivan Ferreira da Silva, Inédio Arcari e Minéia Cappellari Fagundes e demais professores pela confiança, ensinamentos e amizade.

Aos colegas do LAPSEE, Marlon Borges Correia de Oliveira, Leonardo Henrique Faria Macedo Possagnolo, Diogo Rupulo, Jeferson Back Vanderlinde, Fenando Vladimir Cerna Ñahuis e aos demais, pela amizade e contribuição. E também aos amigos em geral, pela amizade e apoio.

"O futuro pertence àqueles
que acreditam na beleza de seus sonhos."
(Elleanor Roosevelt)

RESUMO

Neste trabalho é proposta a resolução de problemas multimodais usando duas diferentes meta-heurísticas: Algoritmo Genético de Chu-Beasley modificado e o Algoritmo Genético de Chaves Aleatórias Viciadas (BRKGA), com foco principal nos resultados obtidos por esta última. É feita especificamente a implementação das meta-heurísticas e comparação dos resultados obtidos por estas diferentes técnicas. Uma característica muito importante do BRKGA é a estruturação que permite separar o algoritmo em duas parcelas claramente diferenciadas, uma parcela que depende exclusivamente das características do BRKGA e, portanto, independente do problema que se pretende resolver e outra parcela que depende exclusivamente das características especificas do problema que pretendemos resolver. Essa característica geral do BRKGA permite que ele seja facilmente aplicado a uma grande variedade de problemas, já que a primeira parcela pode ser integralmente aproveitada na resolução de um novo problema. Por outro lado, o Algoritmo Genético de Chu-Beasley (AGCB) é caracterizado pela substituição de um único indivíduo no ciclo geracional e pelo controle máximo de diversidade, mas isto não é suficiente para resolução de problemas complexos e multimodais, sendo assim, é apresentado o AGCB modificado, onde o critério de diversidade é estendido, a população inicial e o descendente gerado no ciclo geracional passa por uma melhoria local. Essas características tornam-o competitivo justificando a comparação com o BRKGA.

Palavras-chave: Meta-heurísticas. Algoritmo Genético de Chu-Beasley. Algoritmo Genético Tradicional. Algoritmo Genético de Chaves Aleatórias Viciadas (BRKGA).

ABSTRACT

In this work it is proposed the resolution of multimodal problems using two different metaheuristics: Chu-Beasley's Genetic Algorithm and Biased Random Key Genetic Algorithm (BRKGA), focusing mainly on the results obtained by the latter. Specifically the implementation and comparison of results obtained by these different techniques is made. There are several metaheuristics, each with its own specific characteristics which have advantages and disadvantages for the resolution of certain problems and in several ways in the implementation and results. A very important feature of the BRKGA is the structure that allows to separate the algorithm into two clearly different parts, one part that depends exclusively on the characteristics of BRKGA and therefore independent of the problem to be solved and another part that depends exclusively on the specific characteristics of the problem we intend to solve. This general feature of the BRKGA allows it to be readily applied to a variety of problems, because the first component part can be fully utilized to solve a new problem. On the other hand, Chu-Beasley's Genetic Algorithm (AGCB) is characterized by the replacement of a single individual in the generation cycle and by maximum control of diversity, but this is not enough to solve complex and multimodal problems, therefore it is presented the modified AGCB, where the diversity criterion is extended, the initial population and the descendant generated in the generational cycle passes through a local improvement. These features make it competitive, justifying the comparison with BRKGA.

Keywords: Metaheuristics. Chu-Beasley's Genetic Algorithm. Traditional Genetic Algorithm. Biased Random Key Genetic Algorithm (BRKGA).

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AG	Algoritmo Genético
AGCB	Algoritmo Genético de Chu-Beasley
BRKGA	Algoritmo Genético de chaves aleatórias viciadas
GAP	*Generalized Assignment Problem*
PUC	Cruzamento Uniforme Parametrizado
RKGA	Algoritmo Genético de chaves aleatórias

LISTA DE SÍMBOLOS

A_{pop}	População inicial
B_{temp}	Matriz temporária das soluções recombinadas
$f_{fitness}$	Vetor que armazena as funções objetivos / algoritmo BRKGA
$f(x)$	Função objetivo
$f^*(x)$	Função objetivo de melhor qualidade
$F(x)$	Função guia da região F de factibilidade
k	População corrente
$k+1$	Nova população
kn	Tamanho do vetor n_{aux}
l	Limite inferior de x
m	Recursos do problema GAP
n	Tamanho do indivíduo
n_{aux}	Vetor posição das soluções que não satisfazem o critério de diversidade
n_{pc}	Tamanho da população corrente
n_{pop}	Tamanho da população / tamanho da população ideal algoritmo AGCB
n_e	Tamanho da população de indivíduos de elite
n_{fit}	Vetor parâmetro de factibilidade
n_m	Tamanho da população mutante
n_{unf}	Vetor parâmetro de infactibilidade
u	Limite superior de x
ρ_e	Probabilidade de herdar variável de elite
ρ	Valor aleatório gerado entre [0,1] sujeito ao parâmetro ρ_e
ρ_m	Probabilidade de mutação
$\mathbb{R}$	Conjunto dos números reais

$\mathbb{Z}$	Conjunto dos números inteiros
ℓ	Limite inferior de x decodificado
F	Região factível
$fitness$	Vetor que armazena o valor das funções objetivos
S	Espaço de busca da $f(x)$
$unftness$	Vetor que armazena o valor da função de infactibilidade

SUMÁRIO

1 INTRODUÇÃO

Há diversas meta-heurísticas poderosas com características diversas podendo ser extremamente eficientes em alguns casos, mas que apresentam desvantagens em outros. São as características destas técnicas, que definem suas vantagens e desvantagens para problemas específicos.

Nesta dissertação a técnica em questão é o Algoritmo de Chaves Aleatórias Viciadas (BRKGA) apresentada por Gonçalves e Resende (2011), que será comparado com o Algoritmo Genético de Chu-Beasley(AGCB) modificado, que se trata de uma técnica com características fundamentais para resolução de problemas de otimização.

Em 1994, Bean introduziu o Algoritmo Genético de Chaves Aleatórias (RKGA) para resolver problemas de otimização combinatorial. Entre as características do RKGA, tem destaque uma muito importante que é a estruturação que permite separar o algoritmo em duas parcelas claramente diferenciadas (BEAN, 1994; GONÇALVES; RESENDE, 2011).

A palavra viciada (do inglês *Biased*) no BRKGA, incorporada pelos autores Gonçalves e Resende (2011) é para mostrar que a recombinação não é puramente aleatória, isto é, uma das soluções geradoras deve ser necessariamente de elite (viciada) em contraposição com a proposta original em que as duas soluções geradoras podem ser do conjunto da população.

O principal objetivo deste trabalho é resolver uma série de problemas utilizando duas meta-heurísticas, o AGCB e o BRKGA e compará-los para verificar as vantagens e desvantagens desta última meta-heurística (BEAN, 1994; GONÇALVES; RESENDE, 2011; SILVA et al., 2012). Esta ideia se justifica principalmente pela característica geral do BRKGA que permite que seja facilmente aplicado a uma grande variedade de problemas já que a primeira parcela pode ser integralmente aproveitada na resolução de um novo problema, entre outras justificativas apontadas após os objetivos secundários do trabalho.

Para atingir o principal objetivo deste texto, citado no parágrafo anterior, é necessário que alguns objetivos secundários sejam levados em conta, dentre eles pode-se elencar:

- Elaboração de um texto específico para abordagem de cada meta-heurística a ser utilizada de forma didática e ao mesmo tempo com a formalidade necessária;
- Apresentação de forma detalhada do Algoritmo Genético de Chu-Beasley Modificado, para compreensão do mesmo e principalmente suas características, para assim ficar claro seu potencial e motivo de escolha para comparação;
- Apresentação de forma detalhada do Algoritmo Genético de Chaves Aleatórias Vi-

ciadas, para compreensão do mesmo e principalmente de suas características, para assim ficar claro o objetivo central do trabalho;

- Implementação e apresentação dos resultados e comparações.

As meta-heurísticas propostas possuem características específicas que se apresentam como vantagens e desvantagens para a resolução de problemas multimodais. Neste trabalho são utilizadas e comparadas duas meta-heurísticas, o Algoritmo Genético de Chu-Beasley Modificado e o Algoritmo Genético de Chaves Aleatórias Viciadas com características que proporcionam a resolução de tais problemas, com uma atenção especial nos resultados obtidos por esta última. Cada uma delas tem características específicas e que serão apresentadas no decorrer do texto. Estas características influenciarão de diversas maneiras na implementação e resultados e por isto os mesmos serão comparados.

A ideia central está na análise comparativa do BRKGA com o AGCB modificado. Um detalhe muito importante do BRKGA é a estruturação que permite separar o algoritmo em duas parcelas claramente diferenciadas, uma parcela que depende exclusivamente das características do BRKGA e, portanto, independente do problema que se pretende resolver e outra parcela que depende exclusivamente das características especificas do problema que pretendemos resolver. Essa característica geral do BRKGA permite que seja facilmente aplicado a uma grande variedade de problemas já que a primeira parcela pode ser integralmente aproveitada na resolução de um novo problema. Em outras palavras, esta característica do BRKGA justifica e motiva a implementação dos testes e comparação dos resultados. Alguns trabalhos nestes sentidos são Coco, Noronha e Santos (2011), Mendes, Gonçalves e Resende (2009), Gonçalves e Resende (2011), Gonçalves, Resende e Mendes (2011), Silva e Resende (2013).

O Algoritmo Genético Tradicional é apresentado no Capítulo 2. Esta apresentação é feita de forma breve, sem preocupação com as origens epistemológicas desta meta-heurística, mas com bastante formalidade na abordagem do algoritmo propriamente dito. Desta forma, é feita uma breve introdução e o capítulo é subdividido de forma a abordar os passos do algoritmo. Suas subseções apresentam a geração de população inicial, a codificação, o cálculo da função objetivo, a seleção, a recombinação e mutação, além de outras discussões pertinentes.

No Capítulo 3 é apresentado o Algorítimo Genético de Chu-Beasley (AGCB) e sua versão modificada (AGCB modificado). O AGCB modificado será utilizado para resolver os problemas testes propostos no Capítulo 5 e cujos resultados serão comparados aos do Algoritmo Genético de Chaves Aleatórias Viciadas BRKGA. Na seção 3.1 deste capítulo é apresentado o AGCB e suas principais características e na seção 3.2 apresenta AGCB em sua versão modificada. A seção 3.1 apresenta a estrutura geral do AGCB, abordando a geração da população inicial, controle da diversificação, atualização das soluções no ciclo

geracional, enquanto que, a seção 3.2 aborda a busca local na população inicial, controle de diversidade estendido, a melhoria da solução descendente e o critério de aceitação na população corrente (teste de substituição).

O Capítulo 4 aborda de forma detalhada a teoria e a forma de implementação do BRKGA - *Biased Random Key Genetic Algorithm* ou algoritmo genético de chaves aleatórias viciadas. Para isto, o texto é subdividido em seções, sendo que as mesmas apresentam as características fundamentais do BRKGA, os detalhes de implementação computacional, os componentes do BRKGA, a implementação dos decodificadores, a escolha dos parâmetros do BRKGA, a montagem da população inicial e a comparação do BRKGA com o Algoritmo Genético Tradicional.

No Capitulo 5 são apresentados características dos problemas multimodais, os onze problemas testes utilizados, e por fim o comportamento em sua região de busca. Este capítulo esta dividido em duas seções, na seção 5.1 apresenta características especificas dos problemas multimodais e os problemas testes em subseções separadas e na seção 5.2 apresenta-se o comportamento destes problemas em sua região de busca.

O capítulo 6 traz as formas de implementações, os resultados das implementações e comparações entre os resultados das diferentes meta-heurísticas. Uma vez que são apresentadas nos capítulos anteriores as três meta-heurísticas utilizadas no trabalho, na seção 6.1 reforça-se detalhadamente a forma aplicada neste trabalho, na seção 6.2 são apresentadas as melhores soluções encontradas para o BRKGA e para o AGCB e na seção 6.3 a comparação entre meta-heurísticas de acordo com os resultados encontrados.

Após a abordagem dos resultados no trabalho no Capítulo 6, um capítulo para as considerações finais do trabalho é apresentado. Onde, são apresentados os resultados de acordo com os objetivos iniciais, ou seja, a conclusão do trabalho, além de discussões pertinentes, tais como, trabalhos futuros. Após a exposição das considerações finais são apresentadas as referências bibliográficas que norteiam e contribuem para a leitura deste trabalho.

2 ALGORITMO GENÉTICO - AG

É apresentado neste Capítulo o Algorítimo Genético (AG) e suas características fundamentais, as quais serão enfatizadas em comparação com as meta-heurísticas empregadas na resoluções dos problemas testes no decorrer do trabalho, para melhor compreensão da teoria das meta-heurísticas evolutivas.

2.1 Características Fundamentais do AG

Na década de 70, Holland desenvolveu um algoritmo que ficou conhecido por Algoritmo Genético (AG), inspirado na teoria da evolução e da seleção natural. A teoria da evolução das espécies proposta por Darwin em 1859 é baseada no processo de seleção natural que é determinada pela sobrevivência do indivíduo mais apto a determinado ambiente ou condições de sobrevivência, em outras palavras, pode-se dizer que o melhor indivíduo sobrevive e que seus descendentes serão melhores descendentes para sobreviver às condições impostas pelo ambiente. O algoritmo genético é baseado nesta ideia e também suas etapas são baseadas nas diferentes possibilidades da seleção natural (RENDÓN; ZULUAGA; ROMERO, 2006).

Algoritmos genéticos se tornaram muito populares, pois têm muitas vantagens, como resolver problemas de otimização combinatória complexos, problemas em que a função objetivo é estacionária ou não estacionária, linear ou não linear, contínua ou descontínua. Por outro lado este método também apresenta desvantagens, entre as quais se pode citar, o tamanho trivial da população, a escolha de parâmetros relativos à taxa de recombinação e mutação, e o cuidado com o critério de seleção da nova população (GENDREAU; POTVIN, 2010; YANG, 2010).

A evolução e a seleção natural são processos de otimização estocástica que acontece em tempo real, aplicando suas semelhanças encontradas na forma de resolver problemas de otimização matemática, isto inspirou criação do AG. Devido a complexidade podemos dizer que o AG é baseado somente em alguns componentes da evolução das espécies e seleção natural, assim de acordo com Romero e Lavorato (2012), em AG temos:

- Um cromossomo é correspondente a um indivíduo, o que é considerado como uma proposta de solução em otimização;
- Um gene é correspondente a uma elemento cromossômico (indivíduo), o que se pode considerar como grandezas codificadas de acordo com os problemas de otimização;
- A avaliação de aptidão genética é chamada fenótipo, o que pode-se considerar como função objetivo;

- *Crossing over* é o processo de troca genética entre indivíduos, onde dois cromossomos recombinam parcelas de genes, formando novos indivíduos;
- Mutação é um processo de alteração genética, onde alguns indivíduos da população recebem novos genes, o que na otimização seria a substituição de algumas grandezas por novas, o que geralmente gera um indivíduo de pior qualidade.

O Algoritmo Genético faz buscas em um conjunto de soluções candidatas, as quais possuem variáveis codificadas de acordo com o problema, avaliadas por uma função de adaptação (*fitness*), que simula as adversidades da região de busca das soluções (meio ambiente), classificando-as e procurando encontrar uma melhor solução. No entanto, esta procura pela melhor solução é multidirecional, pois se não cumprir um critério de parada (melhor adaptação), passa para os próximos passos, faz a seleção entre as soluções, implementa a recombinação e a mutação e avalia-se as soluções novamente e se mesmo assim não encontra a melhor solução, repete-se novamente os passos (YANG, 2010).

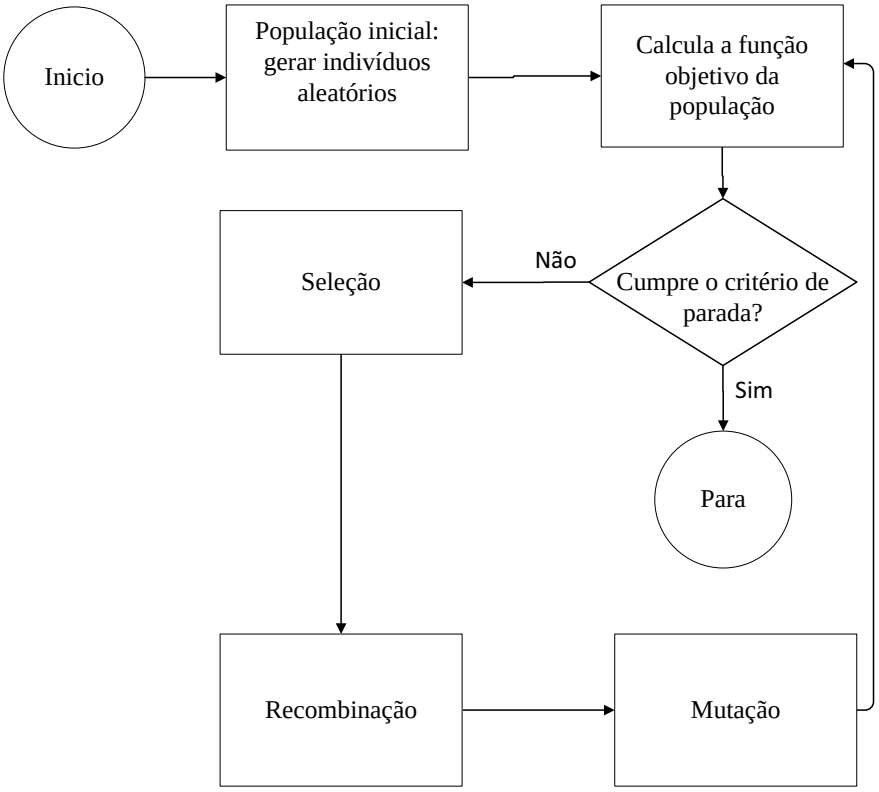
2.2 Algoritmo Genético Tradicional

A ideia fundamental do algoritmo genético consiste em otimizar uma função na forma de um vetor de números correspondentes aos cromossomos, onde as manipulações ou operações com as funções dispostas como vetores que formam a população são baseadas no algoritmo de seleção natural. Isto é feito de acordo com o procedimento a seguir (RENDÓN; ZULUAGA; ROMERO, 2006; YANG, 2010):

1. Etapa inicial: Escolher uma forma de codificação das soluções, como avaliar a melhor solução e o critério de parada. Escolher o tamanho da população, taxa de recombinação, taxa de mutação e tipo de seleção;
2. População inicial: gerar indivíduos com variáveis codificadas;
3. Função Objetivo: avaliar cada indivíduo da população e armazenar a incumbente (melhor solução encontrada durante o processo), se o critério de parada for satisfeito, pare. Em caso contrário, ir ao passo 4;
4. Seleção: selecionar aleatoriamente os indivíduos para participar de um processo evolutivo de acordo com a Função Objetivo;
5. Recombinação: fazer a recombinação entre dois indivíduos;
6. Mutação: realizar mutação e terminar de gerar a nova população da seguinte geração e voltar para o passo (3).

A estrutura do Algoritmo Genético é apresentada na Figura 1.

Figura 1 – Fluxograma geral de um algoritmo genético

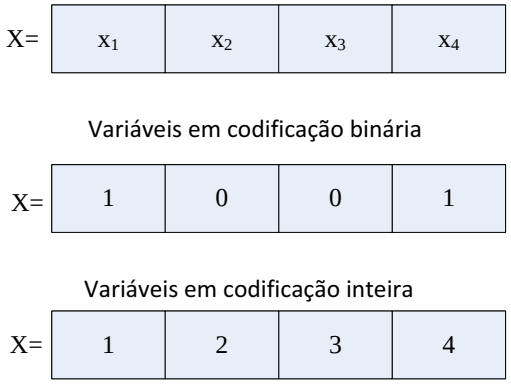


Fonte: Elaboração do próprio autor.

2.2.1 A Codificação

De acordo com Rendón, Zuluaga e Romero (2006), o processo de codificação depende da natureza das variáveis x_i de decisão do problema ou da representação de uma configuração $x = (x_1, x_2, \cdots, x_n)$. Por exemplo, as variáveis podem ser binárias, $x \in \{0, 1\}$ ou inteiras, $x \in \mathbb{Z}$ como apresentado na Figura 2.

Figura 2 – Codificação da variáveis no algoritmo genético



Fonte: Elaboração do próprio autor.

2.2.2 *Cálculo da Função Objetivo ou Função Fitness*

Para determinar a qualidade de uma solução necessita-se ter uma estratégia adequada, de acordo com o objetivo do problema, sendo esta uma função de adaptação *fitness* que em alguns casos pode ser a própria função objetivo. Com frequência usa-se uma função *fitness* para gerar seletividade das soluções determinando se a proposta de solução x é factível ou infactível. Desta forma, o AG mostra-se eficiente quando os valores da função objetivo apresentam-se bem diferentes. Assim, o algoritmo garante uma maior seletividade para quando sujeitas a um operador de seleção serem, separadas em funções objetivos de alta qualidade das de baixa qualidade (GENDREAU; POTVIN, 2010; YANG, 2010).

2.2.3 *Seleção*

A população atual é sujeita a um operador genético que permite selecionar as soluções que irão participar da geração da nova população. O operador genético termina quando completa o número de descendentes da nova população. Em consequência desta operação, algumas soluções podem gerar vários descendentes e outras nenhum, desaparecendo as informações das soluções consideradas de baixa qualidade (GENDREAU; POTVIN, 2010; YANG, 2010). A seguir, apresenta-se as formas de implementar seleção.

1. Seleção proporcional - é a forma mais simples de implementação, onde cada solução candidata gera um número de descendentes proporcional a função de adaptação.

Apresentada da seguinte forma:

$$Nd_i = \frac{z_i(x)}{z_m(x)} \tag{1}$$

onde

$$z_m(x) = \frac{1}{n}\sum_{i=1}^{n} z_i(x), \tag{2}$$

portanto:

$$Nd_i = n\frac{z_i}{\sum_{i=1}^{n} z_i(x)}. \tag{3}$$

Sendo Nd_i o número de descendentes, $z_i(x)$ o valor da função de adaptação, $z_m(x)$ a média da função objetivo.

Para resolver o problema de número de descendentes Nd_i não inteiro pode-se usar o esquema de seleção da roleta, do inglês *(roulette wheel selection scheme)*. Assim representa um dos componentes aleatórios quando é usado o AG no processo de otimização.

Nesta operação de seleção cada solução candidata é representada por uma proporção na roleta equivalente ao grau de aptidão de gerar descendentes, implementada pela seguinte relação:

$$2\pi(\frac{1}{n})Nd_i \Longleftrightarrow 360(\frac{1}{n})Nd_i. \tag{4}$$

2. Seleção por torneio - do inglês *(tournament selection)* é uma estratégia muito atrativa por ser significativamente diferente da seleção proporcional. Nessa estratégia as configurações são selecionadas realizando n jogos, onde n é também o número de indivíduos da população (RENDÓN; ZULUAGA; ROMERO, 2006).

   Cada jogo é realizado por um conjunto k de configurações escolhidas aleatoriamente, onde ganha a que tiver melhor função de adaptação. Esse conjunto k, em geral é um número pequeno de configurações, onde $k \in 2, 3, 4, 5$. O processo termina após completar-se n jogos.

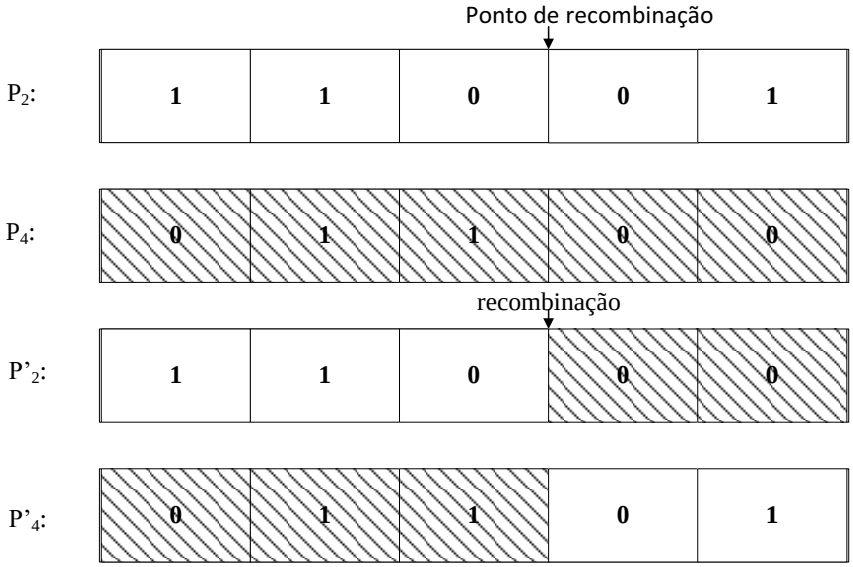
   Esse processo é atrativo por rápido e exigir baixo esforço computacional, ser o mesmo processo tanto em minimização quanto em maximização mudando somente o critério de melhor função objetivo.

2.2.4 Recombinação

As propostas de soluções eleitas pelo critério de seleção são submetidas à recombinação. Esse é um operador que consiste em trocar parcelas dos vetores, e gerar novos vetores aparentemente diferente do processo ocorrido na genética, esse processo também gera diversidade (YANG, 2010).

Neste processo trabalha-se com duas soluções originais denominadas pais, que após trocarem parcelas de informações, geram novas soluções denominadas filhas. Onde, as soluções pais fazem trocas das parcelas de informações que podem ocorrer em apenas um ponto ou múltiplos pontos. Como exemplificado a seguir na Figura 3:

Figura 3 – Recombinação entre P_2 e P_4.



Fonte: Elaboração do próprio autor.

Para este processo necessita-se duas soluções pais vencedoras do processo de seleção, como pode-se observar na Figura 3 gera duas soluções filhas, que seram imcorporadas na população corrente.

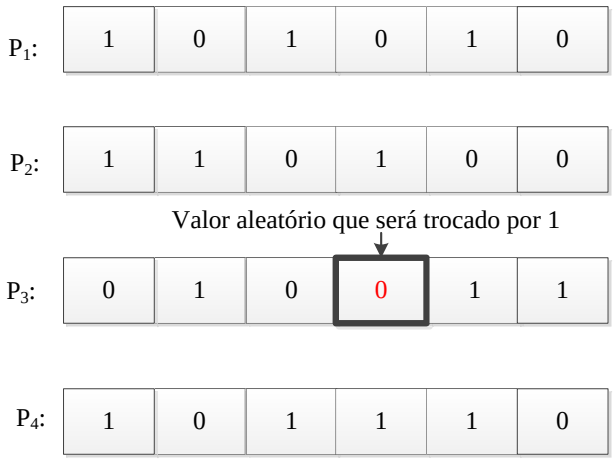
2.2.5 Mutação

Nos primeiros trabalhos apresentados na literatura especializada, a mutação é representada por uma pequena taxa ρ_m considerada como uma probabilidade que cada célula da solução pode mudar seu valor por um gerado aleatoriamente. Mas essa operação

demandava muito tempo de cálculo (RENDÓN; ZULUAGA; ROMERO, 2006; YANG, 2010).

Na prática têm-se técnicas mais rápidas, contando também com uma pequena taxa ρ_m, de probabilidade de mutação, considerando o tamanho da população e a quantidade de genes, elege-se a quantidade de genes que será mutado aleatoriamente ou por um operador de melhora de indivíduos. Exemplificando, em uma população com 4 indivíduos e 6 genes cada e uma taxa de mutação de 0,05, tem-se o produto de 1,2, deste modo 1 gene na população sofrerá mutação e será feita uma escolha aleatória de posições entre os números 1 e 24, como podemos ver na Figura 4.

Figura 4 – Representação da mutação.



Fonte: Elaboração do próprio autor.

No caso da Figura 4 a posição de mutação foi escolhida aleatoriamente entre os 24 valores e alterado para 1, já que as variáveis são binárias. Mas se fossem reais ou inteiras este valor seria gerado aleatoriamente respeitando seus limites.

3 ALGORITMO GENÉTICO DE CHU-BEASLEY - AGCB

Neste capítulo são apresentas detalhadamente as características fundamentais e a forma de implementação do Algoritmo Genético de Chu-Beasley (AGCB) e o AGCB modificado aplicado a problemas de otimização combinatória. Na apresentação dessa metaheurística será dada especial ênfase na análise comparativa das características do AGCB com o Algoritmo Genético Tradicional. Um detalhe muito importante do AGCB é a estruturação diferenciada que manipula a infactibilidade e a função objetivo separadamente, descartando a necessidade de penalização no calculo da função *fitness*. Outras características fundamentais do AGCB são o controle da qualidade e a manutenção da máxima diversidade.

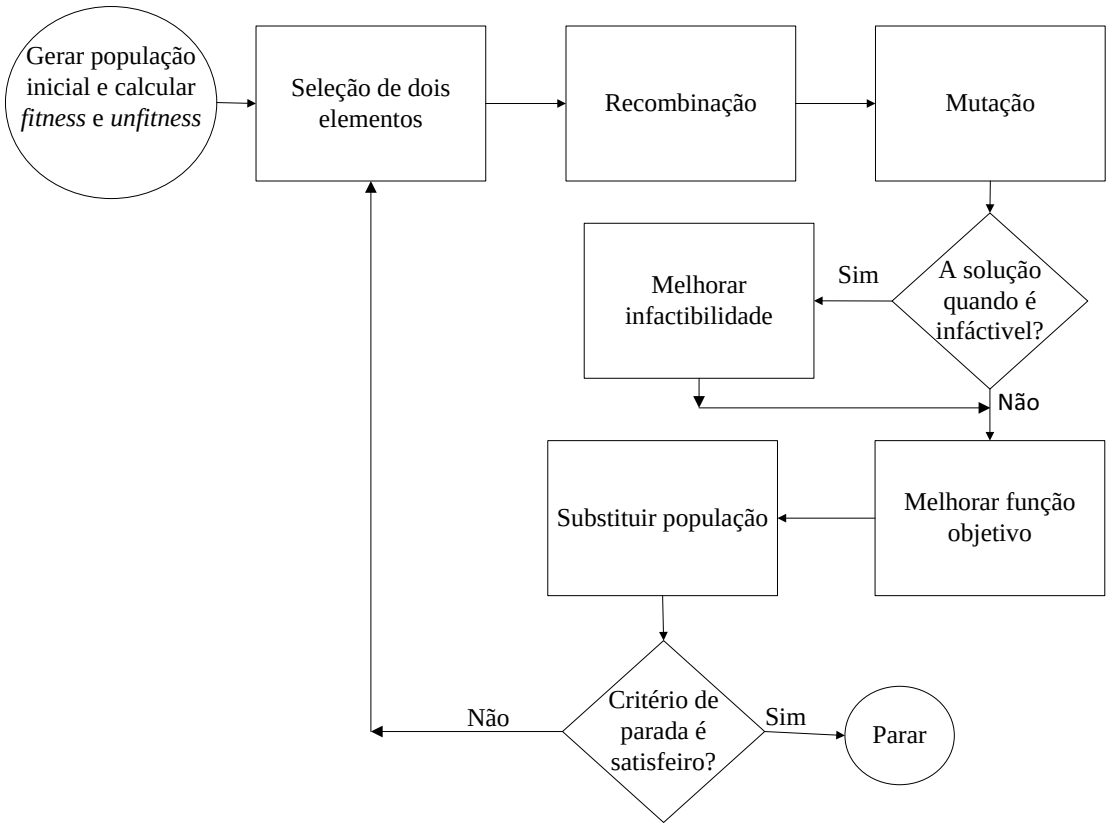
3.1 Características Fundamentais do AGCB

O Algoritmo Genético de Chu-Beasley (AGCB) é um algoritmo genético modificado, proposto por Chu e Beasley em 1997, primeiramente para resolver problemas de atribuição generalizado, do inglês *Generalized Assignment Problem* (GAP). Trata-se de um problema em que pretende-se otimizar a alocação de n tarefas para m agentes, sendo em geral $n \gg m$, onde cada agente possui recurso limitado. Este é um problema de minimização do tipo linear inteiro binário multirrestrito, onde para o tipo de codificação mais usado aparecem muitas propostas de soluções infactíveis como consequência da implementação dos operadores genéticos e na geração da população inicial (CHU; BEASLEY, 1997).

O AGCB também foi adaptado para outros tipos de problemas, como em sistemas elétricos apresentando bons resultados, como pode-se verificar em Macedo et al. (2014), Garces e Romero (2009) e Romero, Rider e Silva (2007). O AGCB é baseado na fundamentação básica do Algoritmo Genético com alterações significativas, que o torna um algoritmo eficiente para resolver problemas grandes e de alta complexidade.

O AGCB pode ser apresentado de forma simplificada como no fluxograma da Figura 5.

Figura 5 – Fluxograma geral de um algoritmo genético de Chu-Beasley



Fonte: Adaptado de Prado e Garces (2013).

Agora de forma mais específica será detalhado os passos do AGCB (ROMERO; RIDER; SILVA, 2007):

1. Especificação dos parâmetros de controles (tamanho da população, taxa de recombinação, taxa de mutação, etc).
2. Especificar características genéricas do algoritmo tais como tipo de codificação, montagem da população inicial, manipulação de infactibilidades, tipo de seleção, etc.
3. Encontrar uma população inicial de forma aleatória que se transforma na população corrente.Encontrar o *fitness* e *unfitness* da população corrente.
4. Implementar a seleção para escolher apenas duas soluções geradoras.
5. Implementar recombinação e preservar apenas um descendente.
6. Implementar mutação do descendente preservado.

7. Implementar uma fase de melhoria local.
8. Decidir se o descendente melhorado pode entrar na população, substituindo um elemento da população.
9. Se o critério de parada não for satisfeito voltar ao passo 4. Caso contrário terminar o processo.

No passo de geração de população inicial, o algoritmo genético de Chu-Beasley (AGCB), assim como o Algoritmo Genético Tradicional, sugere a geração de uma população de indivíduos gerados aleatoriamente. Esse processo caracteriza-se por gerar todos estes indivíduos infactíveis e distantes da factibilidade, observado em casos de problemas complexos de instâncias conhecidas do próprio problema GAP.

Uma proposta inovadora apresentada pelo AGCB é na manipulação das infactibilidades. Desta maneira, este algoritmo apresenta as infactibilidades e a função objetivo de forma separada, onde as funções objetivo são armazenadas em um vetor chamado *fitness* e as infactibilidades em um vetor chamado *unfitness*. O vetor *fitness* tem o propósito de seleção e o vetor *unfitness* juntamente com o *fitness* substituir um elemento da população. Desta forma, eliminando a necessidade de um parâmetro de penalização, quando as duas informações são juntadas em um único *fitness*.

Uma diferença significativa entre o AGCB e o Algoritmo Genético Tradicional é a forma de substituição dos elementos da população. No Algoritmo Genético Tradicional substitui-se todos, ou quase todos os elementos da população sem verificar a diversidade. Enquanto que no AGCB é substituído a cada passo, apenas um elemento da população, facilitando duas estratégias importantes no desempenho deste algoritmo:

- Permitir gerar descendentes melhorados, usando um processo de otimização local do descendente gerado;
- Permitir um controle absoluto da diversidade dos elementos da população corrente.

Essas duas propostas não podem ser implementadas com eficiência em um Algoritmo Genético Tradicional com substituição geracional. O AGCB sugere a implementação de uma melhoria local do descendente gerado, que pode ser uma melhoria de busca local muito simples ou sofisticada levando em conta as características específicas do problema. No caso da aplicação em problemas multirrestritos, este passo é dividido nas seguintes fases:

- Fase de melhoria da infactibilidade;
- Fase de melhoria da qualidade.

A fase de melhoria da infactibilidade do AGCB, aplicada ao problema GAP no descendente gerado, é realizada diminuindo a infactibilidade, através de um processo simples de repasse de tarefas de agentes com recursos violados para agentes que podem executá-las sem ter os recursos violados. Em seguida, na fase de melhoria da qualidade, realiza-se a tentativa de encontrar uma proposta de solução, através da troca de tarefas entre agentes que produzem valores de função objetivo de melhor qualidade e não violam a capacidade de recurso dos agentes envolvidos na troca de tarefas. Essas duas fases produzem descendentes melhorados, com grandes possibilidades de serem melhores que os da população corrente e podem ser incorporados no processo de substituição.

No processo de substituição o AGCB sugere-se substituir um elemento da população corrente pelo descendente gerado preservando a diversidade completa, ou seja, todos os elementos da população deve ser diferentes. Sendo assim, se o descendente for igual a um elemento da população ele é descartado. Caso contrário, segue-se os seguintes passos:

- Se o descendente gerado é infactível, verifica-se sua infactibilidade é menor que a infactibilidade do elemento com maior infactibilidade, caso seja verdadeiro, a substituição procede e em caso contrário o descendente gerado é descartado;
- Se o descendente gerado é factível, substitui-se o elemento da população com maior infactitibilidade. Mas, se todos os elementos da população forem factíveis, deve-se verificar se o descendente gerado é de melhor qualidade para possibilitar a troca.

De forma geral, para um descendente gerado ingressar na população corrente ele necessita ser diferente de todos os elementos da população e ser mais promissor que alguns elementos da população, desta maneira, cumprindo os critérios de infactibilidade e de qualidade das funções objetivo das propostas de soluções factíveis. Nestes casos, utiliza-se o vetor *unfitness*, para ordenar os elementos infactíveis da população e como "medida de infactibilidade", e o *fitness* é a função objetivo original, usada para ordenar as propostas de soluções factíveis, da mesma forma que no processo de seleção.

Os aspectos relevantes do AGCB são:

- Todos os elementos da população são diferentes;
- A lógica de substituição incrementa o número de elemento factíveis, porque as propostas de soluções infactíveis são as primeiras a serem descartadas, isto é, sempre que gera-se uma solução factível elimina-se uma proposta de solução infactível se ainda existir.

Decorrente das observações elencadas anteriormente, o processo encontra uma população corrente apenas com soluções factíveis e esse estágio pode ser atingido em um tempo que depende do tipo do problema e da melhoria local.

As melhores soluções sempre são preservadas por que em cada passo do processo de substituição elimina-se a solução de pior qualidade. A última observação significa que a estratégia funciona melhor que o elitismo dos algoritmos genéticos tradicionais. Entretanto, a grande vantagem do AGCB é o controle absoluto da diversidade. Dessa forma, em problemas altamente complexos e com grande dificuldade de encontrar soluções factíveis, pode ser interessante aumentar o tamanho da população permitindo armazenar soluções factíveis de composição genética diversificada.

3.2 Características Fundamentais do AGCB Modificado

O AGCB modificado é uma proposta direcionada para a aplicação na otimização de problemas complexos de sistemas de energia elétrica, nos quais já existem grande número de experimentos e diversidades de propostas relacionadas a algoritmos heurísticos, em especial com algoritmos construtivos, de acordo com Romero, Rider e Silva (2007).

Neste algoritmo proposto sugere-se modificar o AGCB em três tópicos que acredita-se ser muito importantes:

- A geração da população inicial;
- A fase de melhoria local do descendente gerado;
- E o incremento do controle de diversidade.

A geração de população inicial na meta-heurística proposta consiste em utilizar algoritmos eficientes para cada tipo de problema. Nesta fase pode-se criar uma população inicial de diversidade e qualidade usando algoritmos e estratégias adicionais simples. Dessa forma, na maioria das aplicações a população inicial será composta de propostas de soluções factíveis tornando o vetor *unfitness* pouco ativo ou irrelevante.

Na melhoria local também pode-se usar algoritmos heurísticos rápidos e eficientes, que na maioria dos casos, eliminam totalmente a infactibilidade do descendente gerado, e dessa forma pode melhorar a qualidade da função objetivo. Mas a melhoria local do AGCB para o problema GAP na maioria dos casos não elimina a infactibilidade e a melhoria da qualidade é rudimentar. Considera-se que nesse passo também podem ser usados algoritmos heurísticos eficientes existentes na bibliografia especializada para cada tipo de problema de energia elétrica.

O controle da diversidade no AGCB limita-se à verificação da diferença entre todos os elementos da população, o que pela experiência indica que não é suficiente em problemas complexos e multimodais. Com frequência, as soluções de uma população podem estar restritas a pequenas diferenças e como consequência a população corrente está

representando um número reduzido de regiões do espaço de busca. Uma forma simples de contornar esse problema é estender a diversidade, de forma que, o descendente gerado a cada iteração, pode entrar na população corrente se cumpre os seguintes requisitos:

- Se for de melhor qualidade que a solução armazenada de pior qualidade;
- Se for diferente de cada um dos elementos da população em um número mínimo de elementos do vetor codificação.

O descendente ainda pode ser incorporado na população se não satisfaz o critério de diversidade, desde que, seja de melhor qualidade que as soluções armazenadas na população com os quais não satisfaz o critério de diversidade. Desta forma, para preservar a diversidade deve-se retirar as soluções que não cumprem com o critério de diversidade, ocasionando uma variação no tamanho da população em relação ao tamanho desejado. O objetivo dessa estratégia é eliminar as soluções vizinhas de pior qualidade dos elementos da população corrente facilitando a busca em outras regiões do espaço de busca.

A meta-heurística modificada pode ser descrita, em síntese, pelos seguintes passos:

1. Especificar os parâmetros de controle (tamanho da população, taxa de recombinação, taxa de mutação, etc).
2. Especificar as características genéricas do algoritmo como tipo de codificação, montar a população inicial, manipulação de infactibilidades, escolha de seleção por torneio, etc.
3. Encontrar uma população inicial usando algoritmos heurísticos eficientes, robustos e rápidos. A proposta é priorizar o uso de algoritmos que geram apenas soluções factíveis. Montar o *fitness* e *unfitness* da população inicial.
4. Implementar a seleção por torneio para escolher apenas duas soluções geradoras.
5. Implementar recombinação e preservar apenas um descendente.
6. Implementar mutação do descendente preservado.
7. Implementar uma fase de melhoria local do descendente preservado usando algoritmos heurísticos eficientes.
8. Decidir se o descendente melhorado pode entrar na população substituindo um elemento da população após verificar o teste de substituição.
9. Se o critério de parada não for satisfeito, voltar ao passo 4. Caso contrário, terminar o processo.

3.3 Detalhes de Implementação Computacional

Nesta seção apresenta-se detalhes da implementação computacional, a geração da população inicial e busca local, cálculo da função objetivo, seleção, recombinação e a melhoria local na solução descendente para entrar na população corrente e também manutenção da qualidade das soluções sem perder a diversidade, de acordo com Macedo et al. (2014).

1. Primeiramente declara-se uma população inicial A_{pop} que possuirá as seguintes características:

   a) A população A_{pop} será gerada aleatoriamente;

   b) O tamanho n dos indivíduos, que será ajustada de acordo com o problema que será resolvido;

   c) n_{pop} será o tamanho da população inicial, com poucos indivíduos (Algoritmo Micro-genético), seu ajuste será controlado para melhor desempenho na busca por soluções de melhor qualidade;

   d) A população inicial A_{pop} é em seguida melhorada por um algoritmo de busca local. Este usa 10% do número de teste me na melhoria local.

2. Busca local na população inicial:

   a) Primeiro escolhe-se aleatoriamente uma variável que será mudada;

   b) Mudanças em variáveis continuas são discretizadas e elas devem ser pequenas: 15% de variação máxima possível (que mantém as variáveis dentro de seus limites);

   c) Variáveis inteiras podem mudar aumentando-se ou diminuindo-se uma unidade, dependendo do seu estado de sensibilidade anterior;

   d) Variáveis binárias mudam de 0 para 1 ou 1 para 0 se elas são escolhidas;

   e) Se um aumento (decréscimo) na variável escolhida melhora a *fitness*, então, na próxima vez que variável é escolhida, esta também aumentará (diminuirá).

3. O cálculo da função objetivo a ser utilizado na etapa de melhoria local é guiado pela função objetivo penalizada $F(x)$, sendo a função objetivo $f(x)$ somada a somatória do quadrado das funções de infactibilidades (restrições) $h(x)$ (funções de igualdade) e $g(x)$ (desigualdade) e ρ é o parâmetro de penalização, onde nh e ng é o número de infactibilidades, apresentada na seguinte forma:

$$F(x) = f(x) + \rho \left\{ \sum_{i=1}^{nh} [h_i(x)]^2 + \sum_{i=1}^{ng} \beta_i [g_i(x)]^2 \right\}, \tag{5}$$

onde $\beta_i = 0$ se $g_i(x) \leq 0$ e $\beta_i \neq 0$ se $g_i(x) > 0$.

4. O AGCB realiza seleção por torneio:

   a) São escolhidos aleatoriamente dois ou três indivíduos dependendo do tamanho da população, a partir de uma população corrente.

   b) O indivíduo com melhor *fitness* será selecionado.

   c) Os passos anteriores são repetidos mais uma vez, para escolher o segundo indivíduo para a recombinação.

5. Os indivíduos selecionados participam da etapa de recombinação:

   a) O número de pontos de recombinação é 1/5 do número de variáveis n arredondado;

   b) Estes pontos são selecionados aleatoriamente;

   c) Apenas o descendente com melhor *fitness* é mantido;

   d) Uma melhoria local é então realizada no melhor descendente.

6. Melhoria local:

   a) Semelhante à busca local desenvolvida na população inicial;

   b) É repetida até que nenhuma melhoria significativa no indivíduo seja alcançada;

   c) Mudanças em variáveis contínuas são discretizadas e elas devem ser pequenas: 20% de variação máxima possível (que matem as variáveis em seus limites).

   d) A variação máxima possível é dividida por dois sempre que nenhuma melhoria é obtida em algumas iterações.

3.4 Componentes do AGCB

Os componentes do AGCB são: uma população inicial, busca local na população inicial, seleção de dois indivíduos da população corrente por torneio, recombinação mantendo um único indivíduo descendente, melhoria local do descendente e inserção na população (teste de substituição).

No AGCB pretende resolver em particular os problemas atualizando uma solução por geração, mantendo o controle da qualidade e diversidade da população, e manipulando as infactibilidades em um vetor *unfitness* separadamente da função objetivo em um vetor *fitness*. Para viabilizar essas características é necessário que os indivíduos cumpram o teste de substituição, de acordo com Romero, Rider e Silva (2007).

3.4.1 Teste de Substituição

Para que uma solução descendente (recombinada) entre na população corrente, ela deve satisfazer os critérios do teste de substituição. Para executar o teste inicialmente percorre-se o vetor *fitness* e *unfitness* e armazenamos o valor dos seguintes parâmetros:

1. Onde o vetor n_{unf} assumirá os seguintes valores $n_{unf} = \{0, 1, n^*_{unf}\}$:

   a) Em que $n_{unf} = 0$ indica que não existem soluções infactíveis na população corrente,

   b) $n_{unf} = 1$ indica que existem soluções infactíveis, mas o descendente tem maior infactibilidade que todas elas e,

   c) $n_{unf} = n^*_{unf}$ indica a posição da solução com maior infactibilidade no vetor *unfitness* (maior que a infactibilidade do descendente).

2. Onde o vetor n_{fit} assumirá os seguintes valores $n_{fit} = \{1, n^*_{fit}\}$:

   a) Em que $n_{fit} = 1$ indica que todas as soluções da população são melhores que os descendentes;

   b) $n_{fit} = n^*_{fit}$ indica que a posição de pior qualidade na população é também de menor qualidade que o descendente, sendo assim candidato a ser retirado.

Deve-se usar também um vetor auxiliar n_{aux} em que armazena a posição das soluções que não satisfazem o critério de diversidade com o descendente, e da mesma forma o tamanho kn que define o número de soluções que não satisfazem o critério de diversidade. A preservação da diversidade pode, em alguns casos, mudar o tamanho da população. Deste modo, n_{pop} especifica o tamanho ideal da população (tamanho da população inicial) e n_{pc} o tamanho da população corrente.

Com os parâmetro especificados, o teste de diversidade, que determina o processo de substituição de um elemento da população por um descendente assume a seguinte forma:

1. Percorrer o vetor *fitness* e *unfitness*, encontrar o valor dos parâmetros de controle e verificar se o descendente satisfaz o critério de diversidade.

2. Se o descendente não satisfaz o critério de diversidade ($kn \neq 0$) faz-se o seguinte:

   a) Se o descendente gerado é factível ($n_{unf} = 0$) e a função objetivo desse descendente é melhor que todas as soluções que não satisfazem o critério de diversidade, então incorporar o descendente na população e eliminar todas as kn

soluções que não satisfazem o critério de diversidade . Fazer $n_{pc} = n_{pc} - kn + 1$ e ir ao passo 6.

b) Em caso contrário eliminar o descendente gerado e ir ao passo 6.

3. Se o descendente satisfaz o critério de diversidade fazer o seguinte:

a) Se $n_{pc} < n_{pop}$ então incorporar o descendente na população, fazer $n_{pc} = n_{pc} + 1$ e ir ao passo 6.

b) Em caso contrário ir ao passo 4.

4. Se o descendente for infactível fazer o seguinte:

a) Se não existem soluções infactíveis na população ($n_{unf} = 0$) eliminar o descendente gerado e ir ao passo 6. Em caso contrario ir ao passo 4(b).

b) Se existem soluções infactíveis ($n_{unf} \neq 0$) mas todas tem menor infactibilidade que o descendente gerado ir ao passo 6. Em caso contrário ir ao passo 4(c).

c) Em caso contrário, $n_{unf} = n^*_{unf}$ e existe uma função com maior infactibilidade que do descendente gerado na população retirando a solução que se encontra na posição n^*_{unf} ir ao passo 6.

5. Se o descendente gerado for factível fazer o seguinte:

a) Se existem soluções infactíveis então $n_{unf} = n^*_{unf}$ e nesse caso substituir a solução infactível localizada na posição n^*_{unf} pelo descendente gerado e ir ao passo 6. Em caso contrário ir ao passo 5(b).

b) Se todas as soluções factíveis são de melhor qualidade, $n_{fit} = 1$, então eliminar o descendente gerado e ir ao passo 6. Em caso contrário ir ao passo 5(c).

c) Se existe uma solução factível de pior qualidade localizada na posição n^*_{fit}, incorporar o descendente gerado na população trocando com a solução armazenada na posição n^*_{fit} da população ir ao passo 6.

6. Termina o processo de substituição atualizando alguns parâmetros e a solução incumbente se for o caso.

3.5 Escolha dos Parâmetros do AGCB

No AGCB é necessário ajustar alguns parâmetros para o melhor desempenho do mesmo e outros serão ajustados de acordo com algumas das características do problema (MACEDO et al., 2014). Esses parâmetros são:

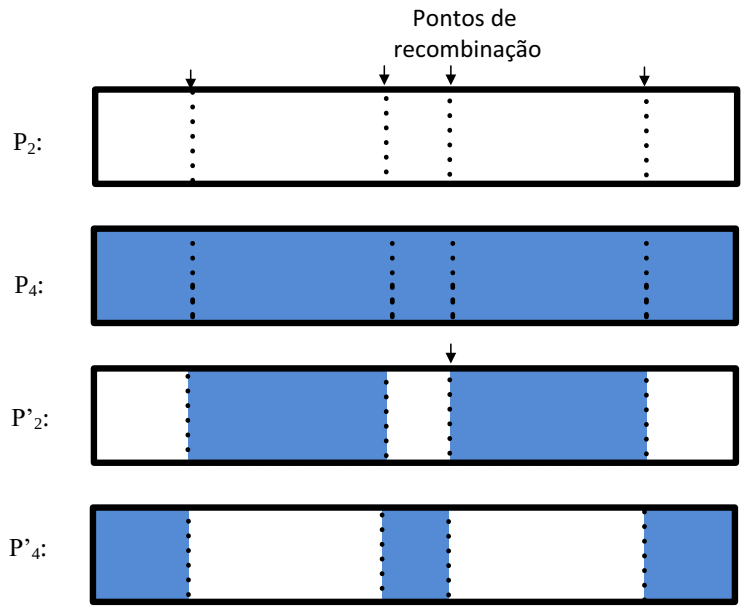
- O tamanho da solução n;
- O tamanho da população n_{pop};
- O número de iteração na busca local na população inicial
- E o número de vezes que o descendente é melhorado.

Enquanto, outros parâmetros tem valores fixos para a meta-heurística:

- O parâmetro de discretização de malha;
- O número de pontos de recombinação;
- E a discretização das variáveis do descendente.

A seleção utilizada no AGCB é a seleção por torneio, sendo executada duas vezes antes de cada processo de recombinação. As duas soluções vencedoras no processo de seleção são recombinadas, e o número de pontos de recombinação é 1/5 do tamanho do indivíduo n arredondado como pode-se ver na Figura 6:

Figura 6 – Operador de recombinação no AGCB



Fonte: Adaptado de Macedo et al. (2014).

Os pontos de recombinação são selecionados aleatoriamente gerando duas soluções, das quais apenas o descendente com melhor *fitness* é mantido, uma melhoria local é então realizada no melhor descendente.

A melhoria local aplicada ao descendente é semelhante ao método de busca executado na população inicial, pois as variáveis contínuas também são alteradas mantendo em seus limites superior e inferior, e será repetida até que obtenha uma melhoria.

3.6 Montagem da População Inicial

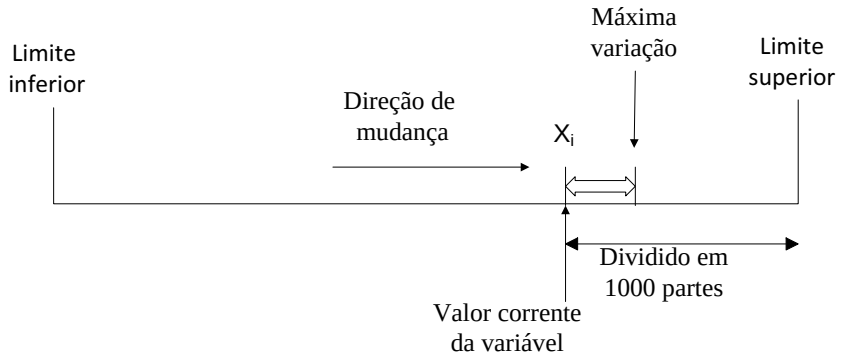
Na fase de geração de população inicial o algoritmo genético de Chu-Beasley (AGCB), sugere a geração de uma população de indivíduos gerados aleatoriamente, mas controlada por algoritmos e estratégias adicionais simples mantendo a diversidade e qualidade.

A população inicial é gerada com um pequeno número de indivíduos. Em seguida, a população inicial é melhorada por um algoritmo de busca local. Este algoritmo usa 10% do número de testes na melhoria local, de acordo com Macedo et al. (2014).

3.6.1 Busca Local na População Inicial

No procedimento de busca local na população inicial, é escolhida aleatoriamente uma variável que será modificada. Essas mudanças em variáveis contínuas são discretizadas e elas devem ser pequenas como mostra a Figura 7: 15% de variação máxima possível (que mantem as variáveis dentro de seus limites).

Figura 7 – Discretização de uma variável na melhoria local



Fonte: Adaptado de Macedo et al. (2014).

A busca por melhoria acontece da seguinte forma; se um aumento na variável escolhida melhora a função objetivo penalizada, então a próxima vez que a mesma variável for escolhida, esta será novamente aumentada. Mas, se um aumento nesta variável escolhida

deteriora a função objetivo penalizada, a próxima vez que a mesma variável for escolhida, esta será diminuída, e vice-versa.

4 ALGORITMO GENÉTICO DE CHAVES ALEATÓRIAS VICIADAS - BRKGA

Apresenta-se neste capítulo de forma detalhada a teoria e a forma de implementação do Algoritmo Genético de Chaves Aleatórias Viciadas, chamado de BRKGA, do inglês, *Biased Random Key Genetic Algorithm*, aplicado a problemas de otimização combinatória. Na apresentação dessa nova meta-heurística será dada especial ênfase na análise comparativa do BRKGA com as outras meta-heurísticas, especialmente com o algoritmo genético tradicional. Um detalhe muito importante do BRKGA é a estruturação diferenciada que permite separar o algoritmo em duas parcelas claramente diferenciadas, uma parcela que depende exclusivamente das características do BRKGA e, portanto, independente do problema que se pretende resolver e outra parcela que depende exclusivamente das características específicas do problema que pretende-se resolver. Essa característica geral do BRKGA, permite que seja facilmente aplicado a uma grande variedade de problemas, já que a primeira parcela pode ser integralmente aproveitada na resolução de um novo problema.

4.1 Características Fundamentais do BRKGA

O BRKGA é apenas uma modificação do chamado RKGA, do inglês *Random Key Genetic Algorithm*, inicialmente proposto por Bean (1994) onde a proposta original foi apresentada para resolver problemas de sequenciamento de tarefas. Para ilustrar a metodologia vamos supor que pretende-se resolver um problema de minimização de uma função objetivo $f(x)$ para um $x \in E^n$ e, portanto, uma proposta de solução é representado através de um vetor com n elementos.

No algoritmo original RKGA, uma proposta de solução é representada de forma codificada através de um vetor com n elementos. Os elementos desse vetor, que representam uma proposta de solução na forma codificada, assumem valores no intervalo $[0, 1]$. Portanto, essa proposta de solução apresentada de forma codificada deve ser decodificada usando um decodificador. Esse decodificador é um algoritmo determinístico que, a partir de uma proposta de solução apresentada de forma codificada, encontra uma solução factível do problema original, isto é, encontra os valores das variáveis do vetor x do problema original, assim como o valor da função objetivo que corresponde a essa proposta de solução. No caso usado por Bean (1994), o problema de sequenciamento de tarefas, uma proposta de solução codificada é um vetor de tamanho n com valores entre 0.0 e 1.0. Nesse contexto, o decodificador decide que a primeira tarefa selecionada é aquela representada pelo maior número armazenado no vetor de codificação e, assim sucessivamente. Deve-se observar que o decodificador tem uma tarefa fundamental no BRKGA porque permite identificar uma proposta de solução factível a partir de uma proposta de

solução codificada. A forma e estrutura do decodificador dependem do tipo de problema que pretende-se resolver.

O nome de algoritmo de chaves aleatórias (do inglês *Random Key*) é pela forma em que é gerada uma proposta de solução e pelo valor que assume cada componente de uma proposta de solução (valores no intervalo $[0, 1]$). A palavra viciada (do inglês *Biased*) no BRKGA, incorporada pelos autores de Gonçalves e Resende (2011), é para mostrar que a recombinação não é puramente aleatória, isto é, uma das soluções geradoras deve ser necessariamente de elite (viciada) em contraposição com a proposta original em que as duas soluções geradoras podem ser do conjunto da população.

O RKGA trabalha de forma geracional como um algoritmo genético tradicional, mas apresenta particularidades muito específicas. A primeira grande diferença é que o RKGA codifica uma proposta de solução em um vetor de tamanho n, onde cada elemento desse vetor é um número no intervalo $[0, 1]$, como já foi mencionado anteriormente. A população inicial de tamanho n_{pop} é montada de forma aleatória, isto é, para cada proposta de solução escolhemos seus n elementos gerando números aleatórios entre 0,0 e 1,0. O processo é repetido até completar os elementos da população.

Após montar a população inicial de forma aleatória, como já foi especificado, deve-se calcular a função objetivo ou *fitness* de cada proposta de solução. Essa tarefa é realizada na parte decodificada. Após calcular a função objetivo de todos os elementos da população, deve-se separar os elementos da população em dois grupos. Um grupo menor formado pelas n_e soluções de elite, isto é, formado por aqueles que têm valores de função objetivo de melhor qualidade e outro grupo formado pelas $n_{pop} - n_e$ soluções não-elite restantes.

Para gerar a nova população, o RKGA usa elitismo. Assim, todas as n_e soluções de elite da geração k passam diretamente a formar parte das soluções da geração $k+1$. Como toda estratégia elitista a ideia fundamental é preservar as melhores soluções e manter uma incumbente monotonamente decrescente. As outras propostas de solução que devem formar parte da geração $k+1$ são encontradas usando duas estratégias, isto é, introduzindo mutantes na população corrente e implementando o operador de recombinação.

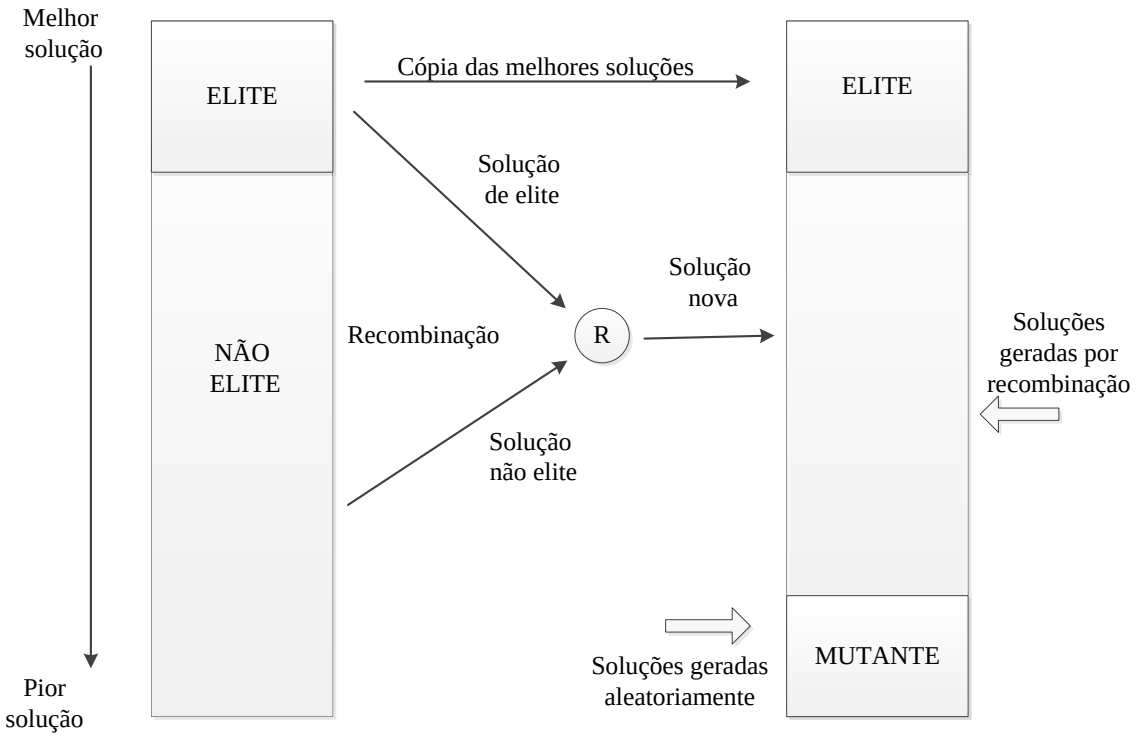
O RKGA não realiza a mutação como nos algoritmos genéticos tradicionais. A estratégia de mutação no RKGA consiste em gerar um conjunto reduzido de n_m propostas de solução usando a mesma estratégia usada para gerar as propostas de solução da população inicial. Assim, o algoritmo incorpora n_m propostas de solução na iteração $k+1$, substituindo as propostas de solução de pior qualidade. Essas propostas de solução são chamadas de mutantes e substituem a estratégia de mutação usada no algoritmo genético tradicional.

As outras propostas de solução para completar a nova população de n_{pop} elementos, na iteração $k+1$, são geradas usando o operador de recombinação. Assim, devem-se gerar

$n_{pop} - n_e - n_m$ propostas de solução usando o operador de recombinação. No algoritmo genético tradicional, o operador de seleção escolhe duas propostas de solução da população k e essas soluções são recombinadas para gerar duas novas propostas de solução que devem ser incorporadas na nova população na iteração $k+1$ (geralmente após implementar a mutação nesses dois novos descendentes). No RKGA não existe o operador de seleção convencional, existe apenas o operador de elitismo que já foi usado para incorporar as n_e soluções de elite na população da iteração $k+1$. Assim, para implementar a recombinação, escolhem-se aleatoriamente dois elementos (propostas de solução) da iteração k e esses elementos são submetidos a recombinação para gerar apenas um descendente que é incorporado na população da iteração $k+1$. Essa estratégia permite que soluções de qualidade diversificada sejam escolhidas aleatoriamente já que não é usado o valor da função objetivo para escolher as soluções geradoras de novas soluções. Nesta estratégia aparece uma nova proposta que é chamada de BRKGA apresentada em Gonçalves e Resende (2011).

Nessa proposta podem ser montadas duas estratégias novas para escolher as duas propostas de solução que devem ser submetidas a recombinação: (1) escolher uma proposta de solução geradora das n_e soluções de elite e a outra proposta de solução geradora dentre as outras $n_{pop} - n_e$ que não são soluções de elite ou, (2) escolher uma proposta de solução geradora das n_e soluções de elite e a outra proposta de solução geradora dentre todas as propostas de solução da população corrente. Deve-se observar que nessas novas propostas se encontra de forma implícita o operador de seleção. A primeira proposta sempre escolhe uma solução de elite e na segunda proposta sempre participa uma solução de elite e a outra proposta de solução pode ser de elite ou pode não ser de elite. Como as soluções de elite representam um número reduzido de elementos, então a participação das soluções de elite deve ser muito intensa na geração de novas soluções no BRKGA quando comparado com o RKGA. Obviamente, cada elemento da população pode ser chamado mais de uma vez para participar da recombinação. A Figura 8 mostra como se monta a nova população na iteração $k+1$ a partir da população na iteração k. O outro aspecto fundamental é a forma em que se implementa a recombinação.

Figura 8 – Montagem da população na iteração k + 1.

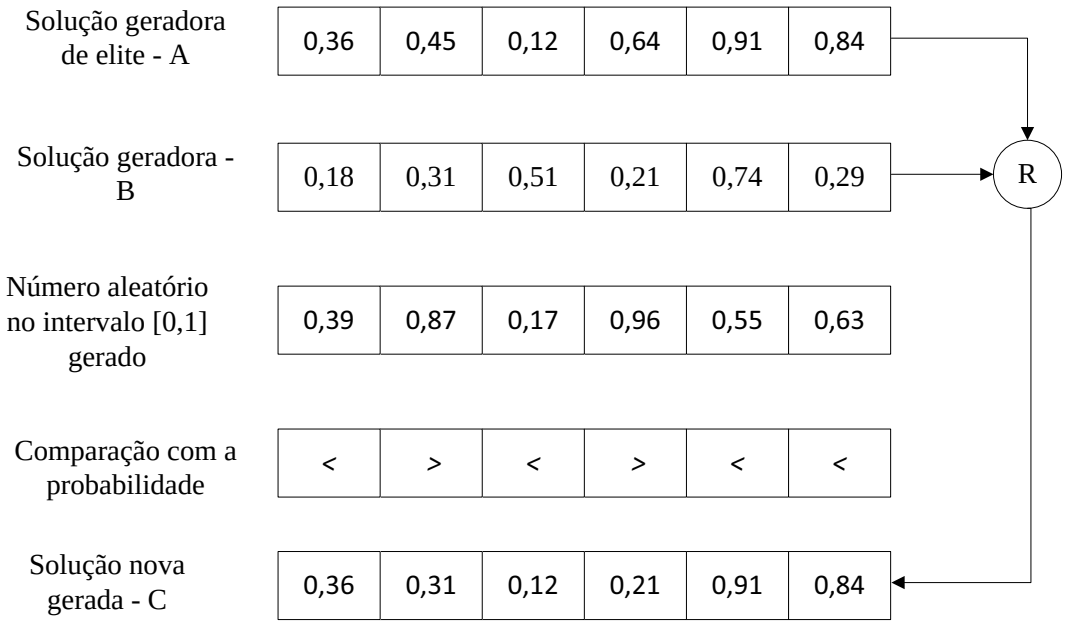


Fonte: Adaptado de Silva e Resende (2013).

Uma vez escolhidas as duas propostas de solução que devem participar da recombinação, então a recombinação proposta no BRKGA é a chamada recombinação PUC, do inglês *Parametrized Uniform Crossover*. A recombinação PUC é simplesmente a recombinação uniforme para gerar apenas um descendente. Para apresentar a recombinação PUC deve-se supor que foi selecionada a solução de elite A e a solução B para participar da recombinação. Adicionalmente, escolhe-se o parâmetro ρ_e que indica a probabilidade de que um elemento (alelo) da solução de elite A seja escolhido para ser incorporado no descendente que está sendo construído. Assim, para gerar os n elementos do descendente C deve-se decidir se copia o valor armazenado em A ou em B. Assim, em um processo que é repetido n vezes, gera-se um número aleatório $\rho \in [0, 1]$ e é comparado com ρ_e. Se $\rho \leq \rho_e$ então é copiado o valor armazenado na solução A e, em caso contrário, copia-se o valor armazenado em B. O valor usado para ρ_e é elevado e tipicamente é escolhido $\rho_e = 0,7$. Assim, em cada caso, existe uma probabilidade de 0,7 para que o valor armazenado em A seja copiado no descendente C que está sendo gerado. Obviamente essa estratégia permite que a maioria dos elementos copiados em C sejam elementos armazenados na solução de elite. Essa estratégia explica também os motivos pelos quais é gerado apenas um descen-

dente ao contrário do que é feito no algoritmo genético tradicional. A Figura 9 mostra a forma em que é gerado um descendente usando a recombinação PUC.

Figura 9 – Recombinação PUC para $\rho_e = 0{,}7$.

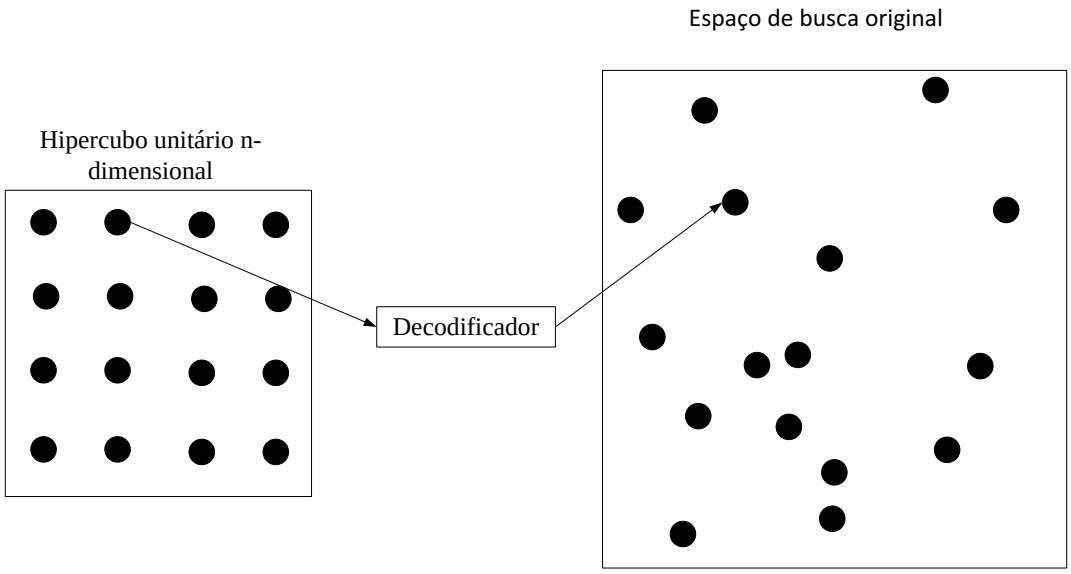


Fonte: Adaptado de Gonçalves, Resende e Mendes (2011).

Após montar a nova população da iteração $k+1$, deve-se calcular o valor da função objetivo das novas propostas de solução usando o decodificador e verificar o critério de parada. Assume-se que qualquer nova solução gerada é factível e essa solução pode ser identificada no espaço de busca original do problema de otimização usando o decodificador.

O BRKGA procura a solução do problema de otimização combinatória indiretamente procurando no espaço contínuo de um hipercubo unitário n-dimensional, usando o decodificador para mapear as soluções no hipercubo para soluções no espaço de busca original do problema de otimização combinatória onde é calculado o valor da função objetivo. Assim, a Figura 10 mostra a função do decodificador.

Figura 10 – Decodificador que procura a solução no espaço de busca original.



Fonte: Adaptado de Silva e Resende (2013).

A meta-heurística BRKGA na verdade apresenta uma estrutura geral para resolver problemas muito variados. Essa estrutura apresenta duas parcelas claramente diferenciadas como se mostra na Figura 11. Existe uma parcela que é independente do problema e, portanto, não precisa conhecer o problema que está sendo resolvido. Essa parcela limita a busca a um espaço definido por um hipercubo unitário n-dimensional. Assim, a única conexão com o problema que está sendo resolvido é através do decodificador que recebe uma proposta de solução codificada na forma de chaves aleatórias e, a partir dessa solução, encontra a solução no espaço de busca original, assim como o valor da função objetivo. Portanto, para definir as características específicas de um BRKGA é preciso apenas definir a forma de codificação na forma de chaves aleatórias e o decodificador.

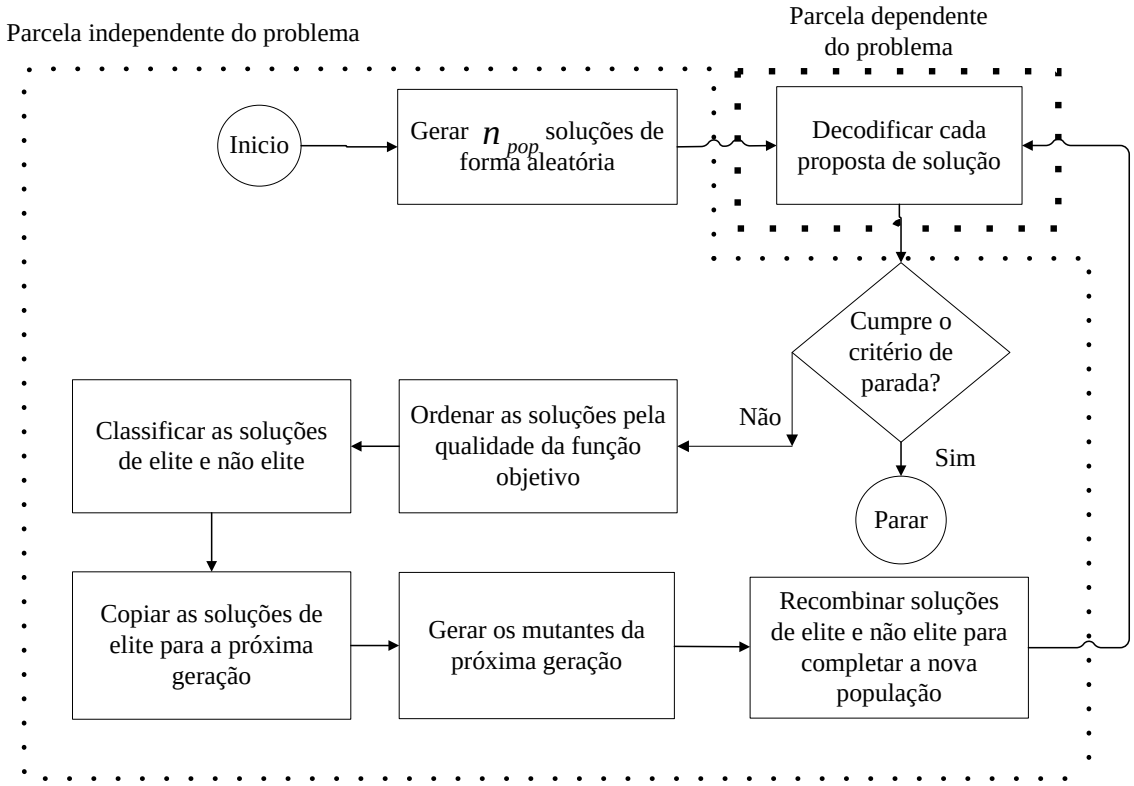
Para ilustrar a forma de codificação e o funcionamento do codificador no BRKGA foi escolhido o problema de cobertura mínima (*set covering problem*). Nesse problema existe uma matriz $A = [a_{ij}]$ binária (com elementos iguais a 0 ou 1) de dimensão $m \times n$ e queremos encontrar a cobertura mínima, isto é, o menor subconjunto de colunas $J^* \subseteq \{1, 2, \cdots, n\}$ de forma que para cada linha da matriz A, $i = \{1, 2, \cdots, m\}$ existe pelo menos um $j \in J^*$ tal que $a_{ij} = 1$. Uma proposta de codificação desse problema pode ser um vetor y de tamanho n e cada elemento desse vetor é um número aleatório no intervalo $[0, 1]$. O j-ésimo elemento do vetor de codificação y está relacionado com a j-ésima coluna da matriz A. A partir dessa proposta de codificação o decodificador encontra

uma proposta de solução J^*. Nesse contexto, o decodificador trabalha da seguinte forma:

- O índice da coluna j da matriz forma parte de J^* se o elemento j do vetor de codificação for maior ou igual a 0,5, isto é, se $y_j \geqslant 0,5$. Dessa forma, percorrendo os elementos do vetor de codificação y encontra-se os elementos de J^* que está formado por uma parcela dos índices das colunas na matriz A. Se o subconjunto encontrado J^* representa uma cobertura válida, então J^* representa uma proposta de solução factível com função objetivo igual a $|J^*|$. Em caso contrário, a proposta de solução não é factível e, portanto, devemos encontrar uma proposta de solução factível a partir de J^*.

- A partir de J^* pode-se encontrar uma solução factível usando um algoritmo heurístico construtivo para o problema de cobertura mínima. Dessa forma, enquanto exista uma linha da matriz A não coberta, escolher uma coluna j da matriz A que ainda não foi selecionada (não se encontra em J^*) e que cobre o maior número de linhas não cobertas da matriz A e, portanto, esse elemento j escolhido é incorporado em J^*. Em caso de empate, escolhe-se a coluna de menor índice. O processo é repetido, adicionando um índice de coluna da matriz em J^* a cada passo até encontrar uma cobertura válida, isto é, um J^* que representa uma solução factível. Finalmente, percorrer os índices das colunas da matriz A que se encontram em J^* e verificar se alguns índices das colunas podem ser retiradas de J^* por serem redundantes. Caso seja possível retirar esse elemento j de J^* e repetir esse processo até avaliar todos os elementos de J^*. Assim, a função objetivo dessa proposta de solução é $|J^*|$.

Figura 11 – Diagrama de fluxo do BRKGA.



Fonte: Adaptado de Silva et al. (2012).

Deve-se observar que o decodificador é um algoritmo determinístico, isto é, uma vez conhecido o vetor de codificação y, então o decodificador sempre encontra a mesma resposta. Adicionalmente, o decodificador sempre encontra uma resposta factível no espaço original do problema, isto é, um J^* que representa uma proposta de cobertura factível. Observemos que caso o J^* inicial obtido pelo decodificador represente uma cobertura factível, então encontra-se o valor da função objetivo e, em caso contrário, um algoritmo heurístico construtivo encontra uma cobertura factível.

É bom enfatizar novamente, que existe apenas uma pequena diferença entre BRKGA e RKGA. A diferença principal está na forma de escolher as duas propostas de solução que devem gerar um novo descendente por recombinação. Assim, no BRKGA uma das soluções geradoras sempre é do conjunto de soluções de elite e a outra solução geradora é escolhida do conjunto de soluções da população. Por outro lado, no RKGA as duas soluções geradoras são escolhidas do conjunto da população. Nesse contexto pode acontecer que nenhuma das soluções geradoras seja de elite no RKGA e, por outro lado, no BRKGA as duas soluções geradoras podem ser de elite. Adicionalmente, os dois algorit-

mos realizam a recombinação PUC que ajuda de forma especial com a estratégia usada pelo BRKGA. Embora a diferença entre ambos os algoritmos seja pequena, na prática existe uma grande diferença de desempenho entre ambos algoritmos, sendo que o BRKGA encontra soluções de qualidade muito melhores quando se considera o mesmo tempo de processamento (BEAN, 1994).

4.2 Detalhes de Implementação Computacional

Nesta seção é feita analise dos detalhes de implementação computacional, especialmente os tópicos relacionados com a estratégia do BRKGA de separar a parte independente do problema com a parte específica e dependente do problema, os tipos de decodificadores, a forma de montar a população inicial, a separação da população, as implementações paralelas do BRKGA e a pós-otimização usando *path relinking* entre as soluções de elite.

1. Primeiramente a população inicial A_{pop} apresenta as seguintes características:

   a) As soluções X são geradas aleatoriamente;

   b) As variáveis X_i possuem valores entre [0, 1];

   c) O tamanho destas soluções é n.

2. A cada elemento de A_{pop} pode ser decodificado usando a equação:

$$x_i = l_i + X_i * (u_i - l_i). \tag{6}$$

3. Após as soluções serem decodificadas passam por uma tentativa de melhorá-las usando a seguinte busca local:

   a) É repetida 100 vezes o número de variáveis;

   b) Mudanças em variáveis contínuas são discretizadas e elas devem ser pequenas: 20% de variação máxima possível (que mantém as variáveis em seus limites).

   c) A variação máxima possível é dividida por dois sempre que nenhuma melhoria é obtida em algumas iterações;

4. Após a busca local as soluções são recolocadas na população pelo seguinte processo:

$$X_i = (x_i - l_i)/(u_i - l_i). \tag{7}$$

5. Durante o processo de decodificação é calculado o vetor *fitness* pela função objetivo, as infactibilidades que serão armazenadas em um vetor *unfitness*, e depois serão agrupadas pela função objetivo penalizada $F(x)$ em um único vetor.

$$F(x) = f(x) + \rho \left\{ \sum_{i=1}^{nh} [h_i(x)]^2 + \sum_{i=1}^{ng} \beta_i [g_i(x)]^2 \right\},$$

onde $\beta_i = 0$ se $g_i(x) \leq 0$ e $\beta_i \neq 0$ se $g_i(x) > 0$.

$F(x)$ é a junção das função objetivo $f(x)$ somada ao parâmetro de penalização ρ multiplicado ao quadrado do somatório das infactibilidades $h(x)$ de igualdade e $g(x)$ de desigualdade, onde nh e ng é o número de restrições.

Nos problemas multimodais a função $F(x)$ será minimizada guiando a função objetivo $f(x)$ para a solução ótima.

4.3 Componentes do BRKGA

Uma particularidade muito especial do BRKGA é que pode ser estruturado em duas partes muito bem diferenciadas, uma parcela que assume uma forma completamente genérica e, portanto, aplicável a qualquer tipo de problema e outra parcela que depende do tipo de problema que se pretende resolver. Assim, a parcela independente do BRKGA pode ser montada uma única vez e depois pode ser reaproveitada para resolver outros problemas. Portanto, quando pretendemos resolver um novo tipo de problema devemos nos preocupar apenas com implementar o tipo de decodificador mais eficiente para o novo problema.

O módulo independente do problema apresenta alguns componentes fundamentais. Esses componentes estão relacionados com o número de elementos que apresenta uma proposta de solução (n), o número de propostas de solução que fazem parte da população corrente n_{pop}, o número de soluções consideradas de elite n_e, o número de propostas de solução novas que devem ser introduzidas na população corrente, isto é, as soluções mutantes n_m e da probabilidade de que um descendente incorpore a informação existente na solução geradora de elite ao implementar a recombinação ρ_e. A população (conjunto de propostas de solução) é armazenada na matriz A_{pop} de dimensão $n_{pop} \times n$ e cujos elementos são números reais do intervalo $[0, 1]$ em que a i-ésima solução é armazenada na linha i da matriz A_{pop}. Após montar os elementos da matriz A_{pop} e, como mencionado anteriormente, com seus elementos sendo números reais do intervalo $[0, 1]$, deve-se encontrar o valor da função objetivo ou *fitness* de cada proposta de solução. Essa tarefa é realizada usando o decodificador. Os valores de função objetivo de cada proposta de solução pode ser armazenado no vetor $f_{fitness}$.

Em cada passo geracional do BRKGA, devem-se implementar os seguintes passos:

1. Ordenar o vetor $f_{fitness}$, que armazena as funções objetivo das propostas de solução, em ordem crescente (problema de minimização) e reordenar as linhas de A_{pop} de acordo com os valores ordenados de $f_{fitness}$. Na verdade, os elementos de A_{pop} não precisam ser removidos, apenas devem ser levadas em conta suas posições verdadeiras de acordo com o ordenamento de $f_{fitness}$. Entretanto, para explicar os próximos passos vamos supor de que as linhas de A_{pop} foram removidas e ordenadas para refletir o ordenamento de acordo com a qualidade de $f_{fitness}$.

2. Gerar $n_{pop}-n_e-n_m$ soluções novas usando recombinação. Assim, gerar $n_{pop}-n_e-n_m$ pares de soluções geradoras em que o índice de uma solução geradora é um número aleatório inteiro que está no intervalo $[1; n_e]$ (é uma solução de elite) e o índice da outra solução geradora é um número aleatório inteiro que está no intervalo $[n_e+1; n_{pop}]$. O i-ésimo descendente gerado usando a recombinação é armazenado na matriz temporária B_{temp} de dimensão $(n_{pop} - n_e - n_m) \times n$. Cada nova solução encontrada usando a recombinação é encontrada gerando n números aleatórios $\{r_1, r_2, \cdots, r_n\}$ no intervalo $[0, 1]$. Assim, para cada $j = 1, 2, \cdots, n$, se $r_j \leq \rho_e$, então o j-ésimo elemento do descendente que está sendo gerado é copiado do j-ésimo elemento da solução geradora de elite. Em caso contrário, nessa posição copia-se o valor do j-ésimo elemento da outra solução geradora.

3. Gerar n_m soluções novas, chamadas de soluções mutantes, de dimensão n. Essas soluções mutantes são geradas usando a mesma estratégia usada para gerar a população inicial. A i-ésima solução mutante é armazenada na linha $n_{pop} - n_m + i$ da matriz A_{pop}.

4. Copiar os $(n_{pop}-n_e-n_m)$ elementos da matriz B_{temp} nas linhas $n_e+1, \cdots, n_{pop}-n_m$ da matriz A_{pop}.

5. Calcular a função objetivo ou *fitness* de cada uma das novas propostas de solução armazenadas nas linhas $n_e + 1, \cdots, n_{pop}$ da matriz A_{pop} e armazenar esses valores nas posições $n_e + 1, \cdots, n_{pop}$ do vetor $f_{fitness}$.

O processo mostrado anteriormente é repetido iterativamente e cada iteração é chamada de uma geração. O processo é repetido até satisfazer um critério de parada que pode ser de vários tipos tais como os seguintes: parar o processo após executar um número de iterações previamente especificado, parar o processo se a incumbente não melhora após executar um número fixado de iterações, parar o processo após atingir um limite de tempo de processamento ou após atingir uma solução de qualidade melhor que um valor previamente especificado.

4.4 Implementação dos Decodificadores

Os decodificadores realizam a função mais importante no BRKGA porque fazem a conexão entre as soluções geradas no hipercubo unitário n-dimensional com as soluções no espaço de busca original do problema de otimização combinatória e que permite encontrar o valor da função objetivo do problema que está sendo otimizado. Os decodificadores podem variar muito em complexidade na sua implementação. Na forma mais simples pode apenas envolver um mapeamento direto entre os valores do vetor de codificação y (no espaço do hipercubo unitário n-dimensional) e os valores das grandezas originais no espaço de busca original que permite encontrar o valor da função objetivo da proposta de solução. Por outro lado, pode assumir formas mais complexas em que a partir de uma proposta de solução y, deve-se encontrar uma solução no espaço de busca original usando um algoritmo heurístico construtivo que ainda pode usar busca local adicional ou as vezes pode ser necessário cálculos adicionais tipo caixa preta.

Supor que o espaço de busca original está constituído por todas as permutações de n números na forma $\Pi_n = \{1, 2, \cdots, n\}$ como acontece no problema de atribuição quadrática ou no problema do caixeiro viajante. Nesse contexto, Bean (1994) mostrou que simplesmente ordenando os elementos do vetor de chaves aleatórias y de uma proposta de solução produz todas as permutações possíveis em $\Pi_n = \{1, 2, \cdots, n\}$, isto é, o vetor de chaves aleatórias pode mapear o espaço completo de Π_n. Assim, no caso do problema de atribuição quadrática, se deseja-se selecionar ρ elementos de um conjunto de n elementos, então monta-se um vetor de chaves aleatórias de n elementos (com cada elemento assumindo um valor no intervalo $[0, 1]$), ordena-se os elementos do vetor de chaves aleatórias e seleciona-se os ρ elementos que correspondem aos ρ menores elementos no vetor de chaves aleatórias. No caso do problema do caixeiro viajante, uma proposta de solução no espaço de busca original é formada pelos índices ordenados dos elementos do vetor de chaves aleatórias ordenados de menor a maior.

Em algumas aplicações pode ser necessário usar um vetor composto de chaves aleatórias. Supor um problema em que os itens devem ser ordenados e adicionalmente cada elemento deve ser alocado em duas posições diferentes, normal e deitado. Neste caso é conveniente que o vetor de codificação de chaves aleatórias seja de $2n$ elementos onde os n primeiros elementos são usados para definir o ordenamento em que os itens são alocados e os outros n elementos para definir o estado normal ou deitado de cada item. Assim, se um elemento da segunda parcela de n elementos tiver um valor menor ou igual a 0,5 então o estado do item é normal e se for maior então o estado desse item é deitado.

4.5 Escolha dos Parâmetros do BRKGA

O BRKGA apresenta poucos parâmetros que precisam ser escolhidos e calibrados. Esses parâmetros são o tamanho do vetor de codificação de chaves aleatórias (n), o tamanho da população (n_{pop}), o número das soluções consideradas de elite (n_e), o número das soluções mutantes (n_m) e a probabilidade (ρ_e) de que um descendente incorpore a informação armazenada na solução geradora de elite. Como em toda meta-heurística, calibrar adequadamente esses parâmetros já representa um novo problema e também, como em toda meta-heurística, esses parâmetros são calibrados de forma empírica e usando o bom senso. Na Tabela 1 são apresentados valores recomendados desses parâmetros mostrados em Gonçalves e Resende (2011). Esses valores foram obtidos através de testes experimentais em vários tipos de problemas.

Tabela 1 – Valores recomendados dos parâmetros do BRKGA.

Parâmetro	Especificação	Valores recomendados
n_{pop}	Tamanho da população	$n_{pop} = a \times n$ onde $1 \leq a \in \mathbb{R}$ é uma constante que depende de n comprimento de uma proposta de solução codificada pelo vetor de chaves aleatórias
n_e	Tamanho das soluções de elite	$0,10n_{pop} \leq n_e \leq 0,25n_{pop}$
n_m	Tamanho das soluções mutantes	$0,10n_{pop} \leq n_m \leq 0,30n_{pop}$
ρ_e	Probabilidade de herdar a informação da solução geradora de elite	$0,5n_{pop} \leq \rho_e \leq 0,8n_{pop}$

Fonte: Adaptado de Silva et al. (2012).

4.6 Montagem da População Inicial

A forma mais conhecida de gerar uma população inicial formada por n_{pop} propostas de solução no BRKGA é através de uma forma aleatória. Assim, cada elemento da matriz A_{pop} é gerada de forma aleatória e com valores no intervalo $[0, 1]$. A geração aleatória da população inicial faz parte da estratégia fundamental dos algoritmos genéticos tradicionais. Entretanto, sabemos que existem algoritmos genéticos especializados em que pelo menos uma parcela da população é gerada usando algoritmos heurísticos. Portanto, a população inicial de um BRKGA também pode ser gerada usando estratégias heurísticas que levem em conta as características específicas do problema. Uma estratégia desse tipo

foi implementada em Buriol et al. (2005) para um problema de roteamento. Nesse caso apenas uma solução é gerada usando o algoritmo heurístico $InvCap$ e as outras soluções são geradas de forma aleatória. Entretanto, existe uma dificuldade para a implementação de estratégias de geração de soluções não aleatórias explicada adiante.

No BRKGA é muito fácil e totalmente determinístico usar o decodificador para encontrar uma solução no espaço de busca original a partir de uma proposta de solução codificada no espaço de busca de um hipercubo unitário n-dimensional. Entretanto, a operação inversa pode não ser trivial nem determinístico já que nesse caso a partir de uma solução no espaço de busca original conhecida pretende-se encontrar uma proposta de solução y no espaço de busca definido por um hipercubo unitário n-dimensional. No problema abordado em Buriol et al. (2005) essa operação inversa é possível e também é possível implementar essa estratégia nos problemas de permutação como os problemas de sequenciamento de tarefas e no problema do caixeiro viajante. Entretanto, em outros problemas pode ser muito complicado implementar essa estratégia.

4.7 Comparação do BRKGA com o Algoritmo Genético Tradicional

Em Gonçalves e Resende (2011) é mostrado em detalhes que o BRKGA é superior aos algoritmos genéticos tradicionais que usam a recombinação e a mutação de acordo com a teoria básica dos algoritmos genéticos. Essa verificação foi realizada comparando diferentes tipos de problemas. Adicionalmente, em Gonçalves e Resende (2011) também se mostra que o BRKGA é superior à maioria das outras meta-heurísticas em uma grande variedade de problemas de otimização combinatória. Assim, o BRKGA se apresenta como uma meta-heurística de grande potencial e deve ser implementado em outras aplicações, especialmente na solução de problemas relacionados com a otimização de sistemas elétricos de potência.

Adicionalmente, podem-se implementar estratégias híbridas como, por exemplo, incorporar uma fase de *path relinking* com as soluções de elite.

5 PROBLEMAS USADOS NOS TESTES

Neste Capítulo são apresentados os problemas para os quais são feitas as implementações, que serão apresentadas no Capítulo 6. Estes problemas escolhidos para realização de testes, por suas características e complexidades na resolução, podem ser encontrados em Silva et al. (2012), Koziel e Michalewicz (1999), Floudas e Pardalos (1990) e Zini (2009).

5.1 Problemas

Problemas multimodais são problemas com múltiplas soluções ótimas, onde algumas podem ser melhores soluções globais e outras melhores soluções locais. A multimodalidade na busca e otimização geralmente provoca dificuldade em um algoritmo de otimização, uma vez que existem muitas soluções atrativas que o algoritmo pode ser direcionado. Portanto, o algoritmo de otimização multimodal tem a tarefa de encontrar múltiplas soluções ótimas simultaneamente ou uma após a outra de forma sistemática Deb e Saha (2010).

Com a finalidade de resolver esses problemas multimodais, serão utilizados algoritmos evolucionários com algumas alterações para serem particularmente úteis em encontrar múltiplas soluções ótimas simultaneamente, simplesmente devido à sua abordagem da população e sua flexibilidade para modificar os operadores de pesquisa. Para fazer esses algoritmos adequados para resolver problemas de otimização multimodal, o principal desafio tem sido o de manter uma diversidade adequada entre os membros da população de modo que múltiplas soluções ótimas possam ser encontradas e mantidas de uma geração para outra.

Nesta seção são apresentados os problemas que foram resolvidos com as diferentes meta-heurísticas. Para maior facilidade de discutir os problemas, cada um será disposto em uma subseção que dará um nome ao problema e apresentará os melhores resultados encontrados na literatura.

5.1.1 Problema G1

$$
\begin{aligned}
\min G1(\vec{x}) \quad &= \quad 5x_1 + 5x_2 + 5x_3 + 5x_4 - 5\sum_{i=1}^{4} x_i^2 - \sum_{i=5}^{13} x_i \\
s.a. \quad & \\
& 2x_1 + 2x_2 + x_{10} + x_{11} \leq 10 \\
& -8x_1 + x_{10} \leq 0 \\
& -2x_4 - x_5 + x_{10} \leq 0 \\
& 2x_1 + 2x_3 + x_{10} + x_{12} \leq 10 \\
& -8x_2 + x_{11} \leq 0 \\
& -2x_6 - 2x_7 + x_{11} \leq 0 \\
& 2x_2 + 2x_3 + x_{11} + x_{12} \leq 10 \\
& -8x_3 + x_{12} \leq 0 \\
& -2x_8 - x_9 + x_{12} \leq 0 \\
& 0 \leq x_i \leq 1, \qquad i = 1, \cdots, 9 \\
& 0 \leq x_i \leq 100, \qquad i = 10, 11, 12 \\
& 0 \leq x_{13} \leq 1.
\end{aligned}
$$

Este é um problema de minimização com $n = 13$ variáveis e nove restrições lineares. O mínimo global encontrado é $(\vec{x}^*) = (1; 1; 1; 1; 1; 1; 1; 1; 1; 3; 3; 3; 1)$, e com função objetivo $G1(\vec{x}^*) = -15$, com apenas seis restrições ativas (1^a, 2^a, 3^a, 7^a, 8^a e 9^a).

5.1.2 Problema G2

$$
\begin{aligned}
\max G2(\vec{x}) \quad &= \quad \left| \frac{\sum_{i=1}^{n} \cos^4(x_i) - 2\prod_{i=1}^{n} \cos^2(x_i)}{\sqrt{\sum_{i=1}^{n} i x_i^2}} \right| \\
s.a. \quad & \\
& \prod_{i=1}^{n} x_i \geq 0,75 \\
& \sum_{i=1}^{n} x_i \leq 7,5n \\
& 0 \leq x_i \leq 10, \qquad 1 \leq i \leq n.
\end{aligned}
$$

Este problema apresenta uma função de maximização não-linear, com duas restrições. Para $n = 20$ variáveis, obteve-se para a melhor solução conhecida $(\vec{x}^*)$ da função

objetivo $G2(\vec{x}^*) = 0,803553$.

5.1.3 Problema G3

$$\begin{aligned} \max G3(\vec{x}) &= (\sqrt{n})^n \prod_{i=1}^{n} x_i \\ s.a. & \\ & \sum_{i=1}^{n} x_i^2 = 1 \\ & 0 \leq x_i \leq 1, \qquad 1 \leq i \leq n. \end{aligned}$$

Para este problema quando $n = 10$ foi encontrado a melhor solução $(\vec{x}^*)$, sendo a melhor função objetivo $G3(\vec{x}^*) = 1$.

5.1.4 Problema G4

$$\begin{aligned} \min G4(\vec{x}) &= 5,3578547x_3^2 + 0,8356891x_1x_3 + 37,293239x_1 - 40792,141 \\ s.a. & \\ & 0 \leq 85,334407 + 0,0056858x_2x_5 + 0,0006262x_1x_4 - 0,0022053x_3x_5 \leq 92 \\ & 90 \leq 80,51249 + 0,0071317x_2x_5 + 0,0029955x_1x_2 + 0,0021813x_3^2 \leq 110 \\ & 20 \leq 9,300961 + 0,0047026x_3x_5 + 0,0012547x_1x_3 + 0,0019085x_3x_4 \leq 25 \\ & 78 \leq x_1 \leq 102 \\ & 33 \leq x_2 \leq 45 \\ & 27 \leq x_i \leq 45, \qquad i = 3, 4, 5. \end{aligned}$$

Para este problema, possuindo $n = 5$ variáveis e 3 restrições, a solução ótima global é $(\vec{x}^*) = (78,0; 33,0; 29,995; 45,0; 36,776)$ e desta forma a melhor função objetivo é igual a $G4(\vec{x}^*) = -30665,5$.

5.1.5 Problema G5

$$
\begin{aligned}
\min G5(\vec{x}) \quad &= \quad 3x_1 + 0,000001x_1^3 + 2x_2 + \frac{0,000002}{3}x_2^3 \\
s.a. \quad & \\
& x_4 - x_3 + 0,55 \geq 0 \\
& x_3 - x_4 + 0,55 \geq 0 \\
& 1000\,\mathrm{sen}(-x_3 - 0,25) + 1000\,\mathrm{sen}(-x_4 - 0,25) + 894,8 - x_1 = 0 \\
& 1000\,\mathrm{sen}(x_3 - 0,25) + 1000\,\mathrm{sen}(x_3 - x_4 - 0,25) + 894,8 - x_2 = 0 \\
& 1000\,\mathrm{sen}(x_4 - 0,25) + 1000\,\mathrm{sen}(x_4 - x_3 - 0,25) + 1294,8 = 0 \\
& 0 \leq x_i \leq 1200, \qquad i = 1,2 \\
& -0,55 \leq x_i \leq 0,55, \qquad i = 3,4.
\end{aligned}
$$

Para este problema problema com $n = 4$ variáveis e possuindo 5 restrições, a melhor solução conhecida é $(\vec{x}^*) = (679,9453; 1026,067; 0,1188764; -0,3962336)$, desta forma a melhor função objetivo é $G5(\vec{x}^*) = 5126,4976$.

5.1.6 Problema G6

$$
\begin{aligned}
\min G6(\vec{x}) \quad &= \quad (x_1 - 10)^3 + (x_2 - 20)^3 \\
s.a. \quad & \\
& (x_1 - 5)^5 + (x_2 - 5)^2 - 100 \geq 0 \\
& -(x_1 - 6)^2 + (x_2 - 5)^2 + 82,81 \geq 0 \\
& 13 \leq x_1 \leq 100 \\
& 0 \leq x_2 \leq 100.
\end{aligned}
$$

Para este problema de minimização é apresentado com $n = 2$ e possuindo 4 restrições, a solução ótima global $(\vec{x}^*) = (14,095; 0,84296)$ e consequentemente a função objetivo $G6(\vec{x}^*) = -6961,81381$.

5.1.7 Problema G7

$$\begin{aligned}
\min G7(\vec{x}) \quad = \quad & x_1^2 + x_2^2 + x_1x_2 - 14x_1 - 16x_2 + (x_3 - 10)^2 + 4(x_4 - 5)^2 + (x_5 - 3)^2 + \\
& +2(x_6 - 1)^2 + 5x_7^2 + 7(x_8 - 11)^2 + 2(x_9 - 10)^2 + (x_{10} - 7)^2 + 45 \\
s.a. & \\
& 105 - 4x_1 - 5x_2 + 3x_7 - 9x_8 \geq 0 \\
& -10x_1 + 8x_2 + 17x_7 - 2x_8 \geq 0 \\
& 8x_1 - 2x_2 - 5x_9 + 2x_{10} + 12 \geq 0 \\
& 3x_1 - 6x_2 - 12(x_9 - 8)^2 + 7x_{10} \geq 0 \\
& -3(x_1 - 2)^2 - 4(x_2 - 3)^2 - 2x_3^2 + 7x_4 + 120 \geq 0 \\
& -x_1^2 - 2(x_2 - 2)^2 + 2x_1x_2 - 14x_5 + 6x_6 \geq 0 \\
& -5x_1^2 - 8x_2 - (x_3 - 6)^2 + 2x_4 + 40 \geq 0 \\
& -0.5(x_1 - 8)^2 - 2(x_2 - 4)^2 - 3x_5^2 + x_6 + 30 \geq 0 \\
& -10 \leq x_i \leq 10, \qquad i = 1, \cdots, 10.
\end{aligned}$$

Para este problema de minimização de uma função objetivo quadrática, com $n = 10$ variáveis e possuindo 3 restrições lineares e 5 não-lineares, o mínimo global encontrado é $(\vec{x}^*) = (2,1719; 2,3636; 8,7739; 5,0959; 0,99065; 1,4305; 1,3216; 9,8287; 8,2800; 8,3759)$ e assim a função objetivo é $G7(\vec{x}^*) = 24,3062091$, sendo estas as ($1^a$, 2^a, 3^a, 4^a, 5^a e 6^a) seis restrições ativas.

5.1.8 Problema G8

$$\begin{aligned}
\max G8(\vec{x}) \quad = \quad & \frac{\text{sen}^3(2\pi x_1)\,\text{sen}(2\pi x_2)}{x_1^3(x_1 + x_2)} \\
s.a. & \\
& x_1^2 - x_2 + 1 \leq 0 \\
& 1 - x_1 + (x_2 - 4)^2 \leq 0 \\
& 0 \leq x_i \leq 10, \qquad i = 1, 2.
\end{aligned}$$

Para este problema existem $n = 2$ variáveis e 2 restrições, para o qual foi encontrado a seguinte solução ótima $(\vec{x}^*) = (0,00015; 0,0225)$ e função objetivo não-linear $G8(\vec{x}^*) = 0,095825$.

5.1.9 Problema G9

$$\begin{aligned}
\min G9(\vec{x}) \quad &= \quad (x_1-10)^2 + 5(x_2-12)^2 + x_3^4 + 3(x_4-11)^2 + 10x_5^6 + 7x_6^2 + x_7^4 - \\
&\quad\; 4x_6x_7 - 10x_6 - 8x_7 \\
s.a. \quad & \\
&\quad\; 127 - 2x_1^2 - 3x_2^4 - x_3 - 4x_4^2 - 5x_5 \geq 0 \\
&\quad\; 196 - 23x_1 - x_2^2 - 6x_6^2 + 8x_7 \geq 0 \\
&\quad\; 282 - 7x_1 - 3x_2 - 10x_3^2 - x_4 + x_5 \geq 0 \\
&\quad\; -4x_1^2 - x_2^2 + 3x_1x_2 - 2x_3^2 - 5x_6 + 11x_7 \geq 0 \\
&\quad\; -10 \leq x_i \leq 10, \qquad i = 1, \cdots, 7.
\end{aligned}$$

Para este problema de minimização da função objetivo polinomial, com soluções limitadas por $n = 7$ variáveis e possuindo 4 restrições não-lineares, a solução mínima global encontrada é

$$(\vec{x}^*) = (2,330449; 1,951372; -0,4775414; 4,365726; -0,6244870; 1,038131; 1,594227)$$

e sua função objetivo é $G9(\vec{x}^*) = 680,6300573$.

5.1.10 Problema G10

$$\begin{aligned}
\min G10(\vec{x}) \quad &= \quad x_1 + x_2 + x_3 \\
s.a. \quad & \\
&\quad\; 1 - 0,0025(x_4 + x_6) \geq 0 \\
&\quad\; 1 - 0,01(x_8 + x_5) \geq 0 \\
&\quad\; x_2x_7 - 1250x_5 - x_2x_4 + 1250x_4 \geq 0 \\
&\quad\; 1 - 0,0025(x_5 + x_7 - x_4) \geq 0 \\
&\quad\; x_1x_6 - 833,33252x_4 - 100x_1 + 83333,333 \geq 0 \\
&\quad\; x_3x_8 - 1250000 - x_3x_5 + 2500x_5 \geq 0 \\
&\quad\; 100 \leq x_1 \leq 10000 \\
&\quad\; 1000 \leq x_i \leq 10000, \qquad i = 2, 3 \\
&\quad\; 10 \leq x_i \leq 1000, \qquad i = 4, \cdots, 8.
\end{aligned}$$

Para este problema de minimização da função objetivo linear, com $n = 8$ variáveis e possuindo 3 restrições lineares e 3 restrições não-lineares, a solução mínima conhecida é

$(\vec{x}^*) = (579,3167; 1359,943; 5110,071; 182,0174; 295,5985; 217,9799; 286,4162; 395,5979)$ e desta forma a função objetivo é $G10(\vec{x}^*) = 7049,330923$, com todas restrições ativas.

5.1.11 Problema G11

$$\begin{aligned} \min G11(\vec{x}) &= x_1^2 + (x_2 - 1)^2 \\ s.a. & \\ & x_2 - x_1^2 = 0 \\ & -1 \leq x_i \leq 1, \qquad i = 1, 2. \end{aligned}$$

Este problema de minimização da função objetivo quadrática, com $n = 2$ variáveis e possuindo uma restrição não-linear, tem a solução ótima global conhecida é $(\vec{x}^*) = (\pm 0,70711; 0,5)$ e com função objetivo $G11(\vec{x}^*) = 0,75000455$.

5.2 Características Gerais dos Problemas

Os problemas multimodais são apresentados com suas regiões de busca limitadas por restrições, onde a solução ótima pode localizar-se entre o limite das regiões factíveis e infactíveis ou em uma região factível separada (FLOUDAS; PARDALOS, 1990; KOZIEL; MICHALEWICZ, 1999).

Pretende-se neste trabalho otimizar problemas multimodais, na versão de minimização, encontrar uma solução ótima x^* pertencente a uma região de busca S contida em $\mathbb{R}^n$, tal que $f(x^*) \leqslant f(x)$, para todos x pertencentes a S, onde S é alguma região de $\mathbb{R}^n$ e a função objetivo f é $f : S \to \mathbb{R}$.

$$\begin{aligned} \min \qquad f(x), \qquad x &= (x_1, x_2, \cdots, x_n) && (8) \\ s.a. & \\ & g_i(x) \leq 0, \qquad i = 1, \cdots, q && (9) \\ & h_i(x) = 0, \qquad j = q+1, \cdots, m && (10) \\ & l_i \leq x_i \leq u_i, \qquad i = 1, \cdots, n && (11) \end{aligned}$$

A otimização de tais problemas consiste em encontrar uma solução x^* e em consequência $f(x^*)$, quando $x \in F(x)$. A função objetivo $f(x)$ é definida no seu espaço de busca S e a função objetivo penalizada $F(x)$ define a região factível. A região factível definida por $F(x) \subseteq S$ é definida por m restrições adicionais ($m \geq 0$), onde $g_j(x) \leq 0$, para $j = 1, 2, \cdots, q$, e $h_j(x) = 0$, para $j = q+1, \cdots, m$.

Na prática comum as equações $h_j(x) = 0$ são substituídas por um conjunto de desigualdades $h_j(x) \leq \delta$ para um pequeno $\delta > 0$. Desta forma, pode-se substituir todas as restrições por $g_j(x) \leq 0$. Assim, em qualquer ponto de $x \in F(x)$, quando as restrições g_j satisfazem $g_j(x) = 0$, é dito que a restrição é ativa para x.

Os onze problemas listados neste trabalho apresentam vários tipos de:

- Funções objetivo lineares ou não lineares. As funções objetivos não lineares podendo ser trigonométricas, polinomiais que podem ser quadráticas, cúbicas, etc;
- Funções de infactibilidades (restrições), que podem ser equações lineares ou não lineares e inequações lineares ou não lineares;
- Variáveis que em geral são reais com limites inferiores e superiores específicos.

A relação entre o tamanho do espaço de busca factível da função penalizada $F(x)$ e o tamanho de todo o espaço de busca não penalizado S para estes problemas varia de 0% para quase 100%. As topologias de espaços de busca viáveis também são bastante diferentes. Na Tabela 2, para cada problema teste, lista-se: o número n de variáveis, o tipo da função $f(x)$, o tamanho relativo da região factível no espaço de busca (ρ), o número de restrições de inequações lineares (LI), equações não-lineares (NE) e inequações não-lineares (NI) e as restrições ativas a no ponto ótimo.

Tabela 2 – Caracteríticas dos problemas testes

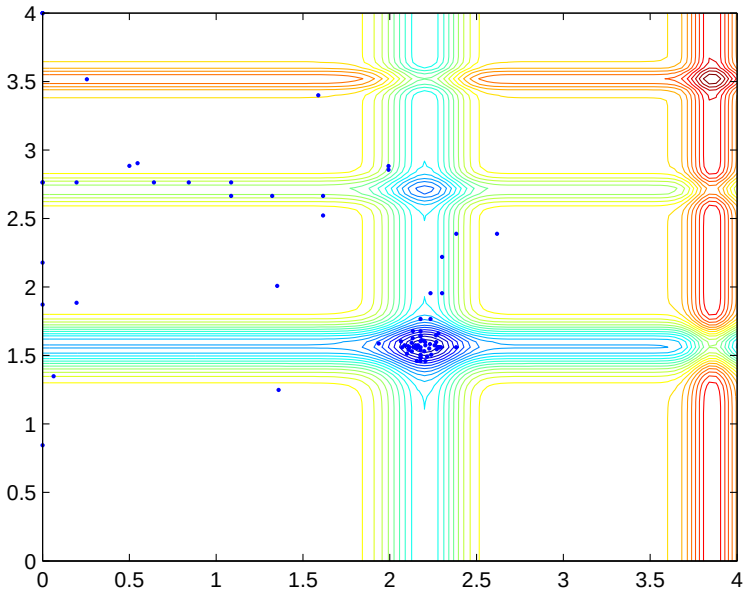
Problemas	n	$f(x)$	ρ	LI	NE	NI	a
$G1$	13	quadrática	0,0111%	9	0	0	6
$G2$	k	não-linear	99,8474%	0	0	2	1
$G3$	k	polinomial	0,0000%	0	1	0	1
$G4$	5	quadrática	52,1230%	0	0	6	2
$G5$	4	cúbica	0,0000%	2	3	0	3
$G6$	2	cúbica	0,0066%	0	0	2	2
$G7$	10	quadrática	0,0003%	3	0	5	6
$G8$	2	não-linear	0,8560%	0	0	2	0
$G9$	7	polinomial	0,5121%	0	0	4	2
$G10$	8	linear	0,0010%	3	0	3	6
$G11$	2	quadrática	0,0000%	0	1	0	1

Fonte: Adaptado de Koziel e Michalewicz (1999).

A relação $\rho = |F|/|S|$ foi determinada experimentalmente através da geração de 1.000.000 de pontos aleatórios da região S e verificada se eles pertencem a F (região de factibilidade). Para G2 e G3 é assumido $k = 50$.

Para fácil observação do comportamento da procura pelo ótimo global em um problema multimodal é apresentado um exemplo na Figura 12 da região de busca. Neste exemplo, verifica-se os pontos que iniciam da esquerda para a direita representando a função objetivo a procura da solução global. Adicionalmente, na Figura 13 pode-se ver as regiões com soluções de baixa e alta qualidades.

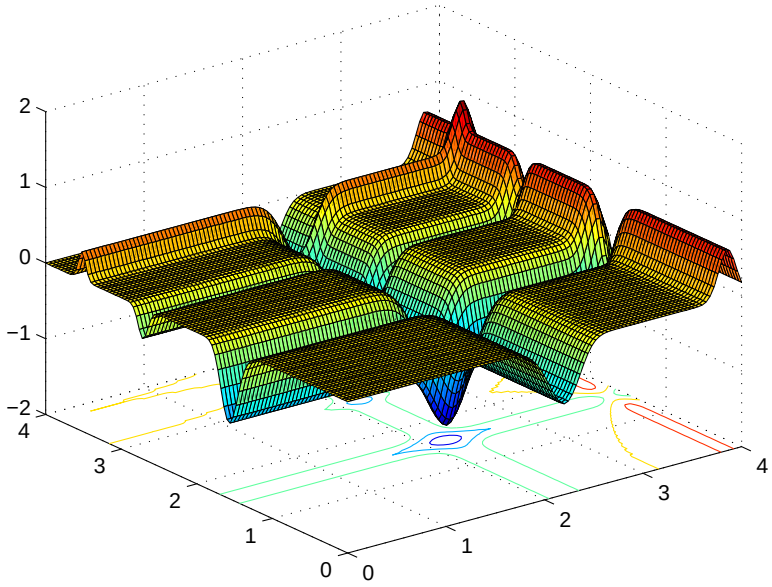
Figura 12 – Espaço de busca de um problema multimodal



Fonte: Elaboração do próprio autor.

Na Figura 13, pode-se observar que as saliências representam a região com melhores soluções e os picos são regiões com piores soluções, dessa forma a maior saliência representa a convergência para o ótimo global.

Figura 13 – Gráfico da função de um problema multimodal



Fonte: Elaboração do próprio autor.

Pode-se observar na Figura 2 os pontos $(f(x))$ que aparecem da esquerda para direita, eles passam por uma região de ótimo local, de onde por meio de uma estratégia que insira diversidade ou mantenha é possível escapar, depois são encaminhados para a região de ótimo global.

6 FORMAS DE IMPLEMENTAÇÕES, RESULTADOS E COMPARAÇÕES

Apresenta-se neste capítulo os resultados dos onze problemas elencados anteriormente, para realizar uma comparação de desempenho do BRKGA em relação ao AGCB modificado. Como se pode ver anteriormente nos capítulos específicos das meta-heurísticas, apesar de se tratarem de algoritmos evolutivos cada uma delas são muito diferentes em seu objetivo principal de convergência e prioridades na manutenção das soluções de qualidade. Também, pode-se observar no capítulo anterior, as características específicas dos problemas que serão resolvidos, assim como os melhores resultados encontrados na literatura. Estas características e dados serão enfatizados com mais detalhe neste capítulo em relação aos melhores resultados alcançados pelos algoritmos.

6.1 Implementações Computacionais

Esta seção será dividida em duas subseções para apresentar como foram ajustados alguns parâmetros para resolver os problemas propostos em cada meta-heurística em particular, apresentando também de forma exemplificada o mecanismo de busca e melhorias aplicadas no decorrer das iterações, assim como os critério de convergência. Deste modo, é apresentado primeiramente para o BRKGA e a seguir para o AGCB modificado.

6.1.1 Implementação Computacional do BRKGA

Como já foi apresentado anteriormente, o BRKGA trabalha de forma separada. Uma parte codificada onde são implementados os operadores da meta-heurística (geração da população inicial, divisão da população em elite e não elite, seleção, recombinação e mutação), e a outra parte decodificada onde serão implementados os operadores de resolução do problema. Desta forma, a meta-heurística trabalha fazendo buscas sem influenciar diretamente com a região de busca do problema que será resolvido (função objetivo, função *fitness*, ordenador da função objetivo e melhoria local).

1. Para resolver estes problemas precisa-se primeiramente declarar alguns parâmetros:
    - O tamanho n_{pop} da população codificada A_{pop} e decodificada B_{pop} varia de acordo com a dificuldade de encontrar boas soluções;
    - O tamanho n das soluções variaram de acordo com os problemas, tanto em A_{pop} quanto em B_{pop};
    - O tamanho da população elite $n_e = 20\%$ de n_{pop}, e o restante $n_{pop} - n_e = 80\%$ não-elite;

- O tamanho da população mutante gerada aleatoriamente é $n_m = 20\%$ de n_{pop} que substituirá as soluções de pior qualidade na população corrente (nova A_{pop});
- O tamanho da população recombinada é $n_{pop} - n_e - n_m$ armazenada em uma matriz B_{temp} correspondente a 60% da população corrente;
- A probabilidade de herdar um individuo de elite $\rho_e = 0,7$.

2. Agora é gerada uma população A_{pop} com n_{pop} indivíduos de tamanho n gerados aleatoriamente com valores X_i codificados entre [0, 1].

3. Em seguida esta população A_{pop} é decodificada e armazenada em uma população B_{pop} depois é calculada a função objetivo $f(x)$ representada no capítulo anterior ($G(\vec{x})$) dos indivíduos e armazenada em um vetor *fitness* de tamanho n_{pop} que será ordenado da melhor para pior solução de acordo com o vetor que armazena as soluções da função $F(x)$ que também promove a convergência para a melhor solução.

4. De acordo com as posições de *fitness* a população A_{pop} é dividida em indivíduos elite n_e e não-elite $n_{pop} - n_e$.

5. O processo de seleção e recombinação é realizado até completar $n_{pop} - n_e - n_m$ em B_{temp}:

   a) No processo de seleção é selecionado aleatoriamente um individuo de elite e ou não-elite,

   b) As duas soluções são recombinadas pelo processo de recombinação PUC gerando um único individuo descendente que cumpre os seguintes critérios:

      i. Para as duas soluções escolhidas é gerado aleatoriamente um valor ρ entre [0, 1], correspondente a posição de suas n variáveis;

      ii. O descendente é composto das variáveis de elite herdadas de acordo com os valores de ρ superiores ao parâmetro ρ_e;

      iii. Os valores inferiores de ρ a ρ_e são herdados do vetor não-elite.

6. Na mutação a população mutante n_m é gerada aleatoriamente da mesma forma que a população inicial A_{pop} e introduzida na população substituindo as piores soluções.

7. Depois é montada novamente a população A_{pop} corrente e decodificada novamente e calculada a função objetivo, desta vez a população B_{pop} passa pela melhoria local, que devolve a população codificada.

6.1.2 Implementação Computacional do AGCB Modificado

O AGCB é modificado para resolver problemas complexos e multimodais, como pode ser visto anteriormente as características fundamentais do AGCB não são suficientes para resolver estes problemas e assim esta meta-heurística é modificada acrescentando uma busca local na população inicial, cada descendente candidato a entrar na população corrente é sujeito a uma melhoria local e o critério de diversidade estendido.

1. Para resolver estes problemas é preciso primeiramente declarar alguns parâmetros:

   - O tamanho n_{pop} da população codificada A_{pop}, será pequeno e feito variações de acordo com a dificuldade de encontrar boas soluções;
   - O tamanho n das soluções variaram de acordo com os problemas em A_{pop};
   - Na população inicial é utilizado 10% do número de testes na tentativa de melhorá-la na busca local;

2. Busca local na população inicial:

   a) Primeiramente escolhe aleatoriamente uma variável que será mudada;

   b) Mudanças em variáveis contínuas são discretizadas e elas devem ser pequenas: 15% de variação máxima possível (que mantem as variáveis dentro de seus limites).

   c) Se um aumento (decresce) na variável escolhida melhora a *fitness*, então, a próxima variável é escolhida, esta também aumentará (diminuirá).

3. O cálculo da função objetivo para problemas multimodais é guiado pela função objetivo penalizada $F(x)$.

4. O AGCB realiza seleção por torneio:

   a) São escolhidos aleatoriamente dois ou três indivíduos dependendo do tamanho da população, a partir de uma população corrente.

   b) O individuo com melhor *fitness* será selecionado.

   c) Esta operação é repetida mais uma vez, completando a quantidade necessária para recombinação.

5. Os indivíduos selecionados são combinados por recombinação:

   a) O número de pontos de recombinação é 1/5 do número de variáveis n arredondado;

b) Estes pontos são selecionados aleatoriamente;

c) Apenas o descendente com melhor *fitness* é mantido;

d) Uma melhoria local é então realizada no melhor descendente.

6. Melhoria local:

a) Semelhante à busca local desenvolvida na população inicial;

b) Repete no descendente três vezes o número de variáveis;

c) Mudanças em variáveis continuas são discretizadas e elas devem ser pequenas: 20% de variação máxima possível (que matem as variáveis em seus limites);

d) Se a soma de todas as infactibilidades é menor que 0.1, então a alteração máxima é 10%;

e) A variação máxima possível é dividida por dois sempre que nenhuma melhoria é obtida em algumas iterações;

f) Então, repete-se dez vezes, se uma melhoria é obtida, zera o contador;

6.2 Resultados Obtidos com as Meta-heurísticas BRKGA e AGCB

Nesta seção são apresentados os melhores resultados das implementações das meta-heurísticas BRKGA e AGCB, para observação de suas variáveis e das melhores funções objetivos encontradas. Por motivo do comportamento característico da meta-heurística BRKGA geralmente a população tem $n_{pop} = 100$, número de gerações $nger = 20$ e $\rho = 10^7$, que precisou ser alterado em alguns problemas com lenta tendência e regão de buscas complexas.

Tabela 3 – Melhores soluções para problema G1

Problema G1			
Meta-heurísticas		BRKGA	AGCB
Valor objetivo		-14,9992020057	-14,9999980
Variáveis de decisão	x_1	1,0000000	1,0000000
	x_2	1,0000000	1,0000000
	x_3	1,0000000	1,0000000
	x_4	1,0000000	1,0000000
	x_5	1,0000000	1,0000000
	x_6	1,0000000	1,0000000
	x_7	1,0000000	1,0000000
	x_8	1,0000000	1,0000000
	x_9	1,0000000	1,0000000
	x_{10}	2,9992454	2,9999979
	x_{11}	3,0006964	3,0000021
	x_{12}	2,9992603	2,9999979
	x_{13}	1,0000000	1,0000000

Fonte: Elaboração do próprio autor.

Tabela 4 – Melhores soluções para problema G2

Problema G2			
Meta-heurísticas		BRKGA	AGCB
Valor objetivo		0,8035966557	0,8033466
Variáveis de decisão	x_1	3,1600554	3,1474388
	x_2	3,1253100	3,1063225
	x_3	3,0957159	3,0832861
	x_4	3,0622577	3,0555419
	x_5	3,0260252	3,0072646
	x_6	2,9882979	3,0068919
	x_7	2,9548856	2,9488048
	x_8	2,9160034	2,9345863
	x_9	0,4913406	0,5012562
	x_{10}	0,4870854	0,4890434
	x_{11}	0,4864616	0,4908883
	x_{12}	0,4784491	0,4809507
	x_{13}	0,4707659	0,4928559
	x_{14}	0,4652367	0,4644524
	x_{15}	0,4631835	0,4625501
	x_{16}	0,4584481	0,4500935
	x_{17}	0,4491773	0,4405982
	x_{18}	0,4473526	0,4426931
	x_{19}	0,4463946	0,4422807
	x_{20}	0,4425383	0,4375346

Fonte: Elaboração do próprio autor.

Tabela 5 – Melhores soluções para problema G3

Problema G3			
Meta-heurísticas		BRKGA	AGCB
Valor objetivo		0,9870310649	0,9998378
Variáveis de decisão	x_1	0,2961039	0,3175667
	x_2	0,3107695	0,3138722
	x_3	0,3187817	0,3171422
	x_4	0,3273485	0,3149634
	x_5	0,3252668	0,3179689
	x_6	0,3186953	0,3156202
	x_7	0,3107701	0,3148963
	x_8	0,3377556	0,3170512
	x_9	0,3054531	0,3161380
	x_{10}	0,3092737	0,3170330

Fonte: Elaboração do próprio autor.

Tabela 6 – Melhores soluções para problema G4

Problema G4			
Meta-heurísticas		BRKGA	AGCB
Valor objetivo		-30665,2216088665	-30665,4260468
Variáveis de decisão	x_1	78,0000000	78,0000000
	x_2	33,0000099	33,0012253
	x_3	29,9968495	29,9964750
	x_4	45,0000000	45,0000000
	x_5	36,7728193	36,7715300

Fonte: Elaboração do próprio autor.

Tabela 7 – Melhores soluções para problema G5

Problema G5			
Meta-heurísticas		BRKGA	AGCB
Valor objetivo		5126,7151316566	5126,1399293
Variáveis de decisão	x_1	657,7967672	675,4082783
	x_2	1049,2740327	1030,8063976
	x_3	0,1345374	0,1220503
	x_4	-0,3886629	-0,3947015

Fonte: Elaboração do próprio autor.

Tabela 8 – Melhores soluções para problema G6

Problema G6			
Meta-heurísticas		BRKGA	AGCB
Valor objetivo		-6960,9932903483	-6961,3399527
Variáveis de decisão	x_1	14,0953713	14,0952089
	x_2	0,8436892	0,8433817

Fonte: Elaboração do próprio autor.

Tabela 9 – Melhores soluções para problema G7

Problema G7			
Meta-heurísticas		BRKGA	AGCB
Valor objetivo		24,3553538663	24,6329770
Variáveis de decisão	x_1	2,1599904	2,1960333
	x_2	2,3817875	2,4260081
	x_3	8,7673775	8,5454124
	x_4	5,0486223	5,0000001
	x_5	0,9900670	1,0027170
	x_6	1,4217049	1,4280627
	x_7	1,3059959	1,3048549
	x_8	9,8186599	9,7778214
	x_9	8,2451395	8,2991404
	x_{10}	8,3565572	8,3897263

Fonte: Elaboração do próprio autor.

Tabela 10 – Melhores soluções para problema G8

Problema G8			
Meta-heurísticas		BRKGA	AGCB
Valor objetivo		-0,0958250280	-0,0958249
Variáveis de decisão	x_1	1,2280162	1,2280162
	x_2	4,2453413	4,2453413

Fonte: Elaboração do próprio autor.

Tabela 11 – Melhores soluções para problema G9

Problema G9			
Meta-heurísticas		BRKGA	AGCB
Valor objetivo		680,8270347131	680,6616450
Variáveis de decisão	x_1	2,3142968	2,3282382
	x_2	1,9724395	1,9434833
	x_3	-0,2886904	-0,4062787
	x_4	4,3056878	4,3840602
	x_5	-0,5983324	-0,6230667
	x_6	1,1098844	1,0711787
	x_7	1,5764442	1,5989103

Fonte: Elaboração do próprio autor.

Tabela 12 – Melhores soluções para problema G10

Problema G10			
Meta-heurísticas		BRKGA	AGCB
Valor objetivo		7049,8772993940	7049,3293170
Variáveis de decisão	x_1	560,8614630	581,5094992
	x_2	1339,1563746	1403,5158946
	x_3	5149,8594618	5064,3039232
	x_4	180,4633318	182,2013902
	x_5	294,0526695	297,4009303
	x_6	219,5715366	217,7986099
	x_7	286,4942542	284,8004600
	x_8	394,0384775	397,4142158

Fonte: Elaboração do próprio autor.

Tabela 13 – Melhores soluções para problema G11

Problema G11			
Meta-heurísticas		BRKGA	AGCB
Valor objetivo		0,7505699874	0,7500048
Variáveis de decisão	x_1	-0,7236793	0,7052600
	x_2	0,5237037	0,4973937

Fonte: Elaboração do próprio autor.

Para a maioria dos problemas a soma das infactibilidades é 0 e alguns muito próximo de 0. Os resultados foram coletados de acordo com a função *fitness* e o grau de factibilidade.

6.3 Comparações entre as Meta-heurísticas

Neste trabalho são apresentadas duas meta-heurísticas evolutivas de alto desempenho para comparação na resolução de problemas multimodais. Estas meta-heurísticas tendem a melhorar indivíduos de uma população, que aparecem nestes problemas bem distantes da factibilidade em sua população inicial. No entanto, essas meta-heurísticas tem características bem diferentes tanto na geração da população inicial quanto na busca por uma solução de boa qualidade, assim precisa-se avaliar a eficiência levando em consideração as diferentes prioridades tomadas nestes processos na resolução dos problemas.

Anteriormente foi visto que a meta-heurística BRKGA gera uma população grande de indivíduos reais codificados entre [0, 1], que tomam as características do espaço de busca após passar pelo decodificador, isso garante que os indivíduos da população inicial tenham um alto grau de diversidade. Por outro lado, o AGCB modificado gera uma pequena população inicial aleatória entre seus limites, mas procede com uma busca local nestes indivíduos tentando melhorá-los.

O BRKGA procede de maneira semelhante ao Algoritmo Genético Tradicional em suas iterações, mas mantendo soluções elites e substituindo uma pequena parcela dos piores indivíduos na mutação, enquanto no AGCB modificado são selecionadas duas soluções que passam o processo de recombinação, e a melhor solução é mantida para depois decidir se substitui a solução de pior qualidade na população corrente.

A melhoria local utilizada no BRKGA é semelhante a utilizada no AGCB, mas nesta última meta-heurística a melhoria local é utilizada uma vez na população inicial e outra no descendente gerado com pouca diferença de implementação. Estas melhorias possuem um parâmetro de discretização de malha que procede mudando de acordo com o sucesso e insucesso de busca.

Uma diferença importante entre estas meta-heurísticas é que o BRKGA utiliza mutação para fugir de ótimos locais, mas esse processo pode gerar soluções infactíveis a cada geração e para corrigir e facilitar a convergência, é aplicado uma melhoria local nas soluções decodificadas. Em contrapartida, o AGCB não possui mutação na forma implementada, sendo que o que impede que ele fique preso em um ótimo local é a forma que procede nas iterações melhorando as soluções e a manutenção máxima de diversidade.

Para comparação de desempenho das meta-heurísticas é apresentado na Tabela 14, os melhores resultados encontrado pelas funções objetivo em comparação com os melhores da literatura. Lembrado que para a comparação dos resultados, precisa-se avaliar a função objetivo de acordo com a minimização ou maximização e também seu grau de infactibilidade.

Tabela 14 – Comparação dos Resultados entre as meta-heurísticas BRKGA e AGCB

Problemas	n	$f(x)$ para BRKGA	$f(x)$ para AGCB	Valor ótimo
$G1$	13	-14,9992020057	-14,9999980	-15
$G2$	20	0,8035966557	0,8033466	0,80361910412559
$G3$	10	0,9870310649	0,9998378	1
$G4$	5	-30665,2216088665	-30665,4260468	-30665,5
$G5$	4	5126,7151316566	5126,139929	5126,4976
$G6$	2	-6960,9932903483	-6961,3399527	-6961,81387558015
$G7$	10	24,3553538663	24,6329770	24,3062091
$G8$	2	-0,0958250280	-0,0958249	-0,0958250414180359
$G9$	7	680,8270347131	680,6616450	680,6300573
$G10$	8	7049,8772993940	7049,3293170	7049,330923
$G11$	2	0,7505699874	0,7500048	0,75000455

Fonte: Elaboração do próprio autor.

Na coleta dos resultados foram levados em conta a soma das infactibilidades. Se as infactibilidades estiverem zeradas o resultado em relação a função de objetivo penalizada deve torna-se igual a função objetivo, na forma implementada para problemas multimodais. Se as infactibilidades estiverem muito altas é alterado o parâmetro de penalidade. Dessa forma, os parâmetros ρ, n_{pop} e n_{ger} foram alterados, verificando a diminuição das restrições e a velocidade da convergência,para um melhor desempenho das meta-heurísticas.

A velocidade da resolução dos problemas pelas meta-herísticas variou de acordo com o número n de variáveis e o número n_{fo} de cálculo da função objetivo (quantidade de vezes que precisaram avaliar suas soluções). Na Tabela 15, é apresentado o número de cálculo da n_{fo} para cada um dos problemas em relação a meta-heurística usada.

Tabela 15 – Número de avaliações das soluções nas meta-heurísticas BRKGA e AGCB

Problemas	n_{fo} para BRKGA	n_{fo} para AGCB
$G1$	2.563.152	191.988
$G2$	34.572.605	500.320
$G3$	1.959.099	2.181.087
$G4$	2.462.691	1.995.719
$G5$	15.792.739	581.598
$G6$	16.500.615	619.488
$G7$	14.823.029	2.473.508
$G8$	3.424.160	88.118
$G9$	2.273.661	4.785.954
$G10$	59.075.638	4.374.818
$G11$	450.509	134.128

Fonte: Elaboração do próprio autor.

Na tabela pode-se perceber que a meta-heuristica AGCB obteve vantagem na resolução da maioria dos problemas, e sua característica influenciou em uma maior perfeição nos resultados, comparando com a tabela anterior. O que significa que a forma implementada se adequou melhor a boa parte dos problemas que o BRKGA.

As duas meta-heurísticas apresentaram características semelhantes na melhoria dos diferentes problemas, precisando alterar parâmetros de penalização, tamanho da população nos mesmos problemas. O problema G2 é um problema que possui uma função objetivo trigonométrica, com uma região de busca bastante complexa, o que dificultou bastante a calibração dos parâmetros. Já o problema G10, apesar de possuir uma função objetivo linear, mas suas restrições dificultava a convergência, por apresentar valores muito altos quando infactíveis e pequenos quando factíveis. No problema G2 foi aumentado n_{ger} e o parâmetro de penalização ρ, e no problema G10 foi aumentado n_{ger}, parâmetro de penalização ρ e n_{pop}.

Apesar das duas meta-herísticas trabalharem com um número padronizado no tamanho da população, onde n_{pop} geralmente assumiu tamanho 100 para o BRKGA e 10 para o AGCB. Este motivo se justifica, pela maior velocidade na convergência do AGCB é por melhorar uma população menor, mas se igualada a população do BRKGA é diminuída drasticamente, isto é, sua vantagem esta em trabalhar com uma pequena população ao contrário do BRKGA. Esses parâmetros precisaram ser alterados, nos problemas G5, G6, G7, G8 e G10.

Ambas meta-heurísticas apresentaram boas soluções e muito próximas das ótimas ou mesmo iguais as melhores soluções encontradas na literatura. Em alguns casos os re-

sultados foram ligeiramente diferentes, em questão de casas decimais. A tolerância de infactibilidades é praticamente zero, para alguns problemas os que apresentam infactibilidades em sua convergência.

7 CONSIDERAÇÕES FINAIS

O objetivo deste trabalho foi apresentar os resultados de alguns problemas testes multimodais utilizando duas meta-heurísticas diferentes, o AGCB e o BRKGA e comparálos para verificar as vantagens e desvantagens em relação a esta última. A característica geral do BRKGA que permite a fácil aplicação a uma grande variedade de problemas, é pelo fato de que a primeira parcela pode ser integralmente aproveitada na resolução de um novo problema é uma justificativa para esta comparação.

Foram apresentadas as duas meta-heurísticas em questão, suas principais características e os problemas a serem resolvidos para fins de comparação de resultados. Onde todas as meta-heurísticas fazem buscas populacionais, as principais características do AGCB modificado é manter a diversidade e a qualidade aproximando gradativamente da solução ótima por meio do suas iterações. O BRKGA é uma proposta de melhoria no desempenho da resolução de problemas contínuos comparado com outras meta-heurísticas, já que trabalha com duas parcelas: uma dependente e outra independente do problema, o que facilita sua aplicação.

Os problemas propostos para comparação possuem regiões complexas de convergência para a solução ótima global e as meta-heurísticas em questão apresentam características com diferenças significativas desde a geração da população inicial até sua convergência para a solução ótima global, tais diferenças são: melhoria local na população inicial do AGCB e população de chaves aleatórias BRKGA, o BRKGA faz as suas operações em um hipercubo n-dimensional e o AGCB implementação seus operadores adicionando apenas uma solução na população corrente, etc. Desta forma não foram necessárias muitas iterações para encontrar a solução próxima da ótima ou ótima, mas um cuidado com o parâmetro de convergência observando a diminuição das infactibilidades.

Na meta-heurística BRKGA o controle da diversidade não é uma prioridade, mas significativo em relação a geração da população inicial codificada e a população mutante inserida durante o ciclo geracional, necessitando uma melhoria local nas soluções da nova população decodificada antes de recolocar as soluções da nova população codificada. Agora a meta-heurística AGCB possui um controle máximo de diversidade, mas nos problemas resolvidos sua principal vantagem foi na convergência n_{pop}, n_{ger} e ρ adicionando apenas uma solução na solução corrente.

Nas duas meta-heurísticas pode-se coletar soluções de ótima qualidade observando a convergência destas, levando em conta a função *fitness*, e o cuidado com o aumento das infactibilidades. Na meta-heurística AGCB foi possível coletar resultados mais próximos do ótimo em alguns problemas levando em conta as casas decimais.

Uma vantagem do AGCB foi trabalhar melhorando duas soluções em seu ciclo

geracional e substituindo a pior na população corrente, esse processo em geral se caracteriza por uma rápida convergência e ótimas soluções quando a população é pequena. No entanto, o BRKGA depende de uma população consideravelmente grande, já que ele trabalha considerando indivíduos elites, inserindo novos indivíduos para o processo de mutação.

As duas meta-heurísticas testadas apresentaram bom desempenho na resolução dos problemas multimodais, pois apresentaram soluções boas e muito próximas das ótimas ou mesmo iguais as ótimas.

REFERÊNCIAS

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

BURIOL, L.; RESENDE, M.; RIBEIRO, C.; THORUP, M. A hybrid genetic algorithm for the weight setting problem in ospf-is-is routing. *Networks*, Washington, v. 46, n. 1, p. 36–56, 2005.

CHU, P. C.; BEASLEY, J. E. A genetic algorithm for the generalised assignment problem. *Computers & Operations Research*, Oxford, v. 24, n. 1, p. 17–23, 1997.

COCO, A. A.; NORONHA, T. F.; SANTOS, A. C. Algoritmo baseado em brkga para o problema de Árvore geradora mínima robusta. In: SIMPÓSIO BRASILEIRO DE PESQUISA OPERACIONAL, 45., 2011, Natal. *Anais ...* Natal: SBPO, 2011. p. 1–12.

DEB, K.; SAHA, A. Finding multiple solutions for multimodal optimization problems using a multi-objective evolutionary approach. In: ANNUAL CONFERENCE ON GENETIC AND EVOLUTIONARY COMPUTATION, 12., 2010, New York. *Proceedings ...* New York: ACM, 2010. p. 447–454.

FLOUDAS, C. A.; PARDALOS, P. M. *A collection of test problems for constrained global optimization algorithms*. Heidelberg: Lecture Notes in Computer Science, 1990.

GARCES, L.; ROMERO, R. Specialized genetic algorithm for transmission network expansion planning considering reliability. In: INTRNATIONAL CONFERENCE ON INTELLIGENT SYSTEM APPLICATIONS TO POWER SYSTEMS, 15., 2009, Curitiba. *Proceedings ...* Curitiba: IEEE, 2009. p. 1–6.

GENDREAU, M.; POTVIN, J.-Y. *Handbook of metaheuristics*. 2. ed. New York: Springer, 2010.

GONÇALVES, J.; RESENDE, M. Biased random-key genetic algorithms for combinatorial optimization. *Journal of Heuristics*, New York, v. 17, n. 5, p. 487–525, 2011.

GONÇALVES, J.; RESENDE, M.; MENDES, J. A biased random-key genetic algorithm with forward-backward improvement for the resource constrained project scheduling problem. *Journal of Heuristics*, New York, v. 17, n. 5, p. 467–486, 2011.

KOZIEL, S.; MICHALEWICZ, Z. Evolutionary algorithms, homomorphous mappings, and constrained parameter optimization. *Evolutionary Computation*, Cambridge, v. 7, n. 1, p. 19–44, 1999.

MACEDO, L. H.; RIDER, M. J.; FRANCO, J. F.; ROMERO, R.; CARREÑO, E. M. A. *A modified Chu-Beasley's genetic algorithm to solve the optimal power flow problem*. Whashington: *IEEE*, 2014.

MENDES, J.; GONÇALVES, J.; RESENDE, M. A random key based genetic algorithm for the resource constrained project scheduling problem. *Computers & Operations Research*, Kidlington, v. 36, n. 1, p. 92 – 109, 2009.

PRADO, I.; GARCES, L. Chu-beasley genetic algorithm applied to the allocation of distributed generation. In: CONFERENCE ON INNOVATIVE SMART GRID TECHNOLOGIES LATIN AMERICA - ISGTLA, 2013, São Paulo. *Proceedings ...* São Paulo: IEEE, 2013. p. 1–7.

RENDÓN, R. A. G.; ZULUAGA, A. E.; ROMERO, R. *Técnicas de optmización combinatorial*. 2. ed. Pereira: Universidad Tecnológica de Pereira, 2006.

ROMERO, R.; LAVORATO, M. *Metaheurísticas em sistemas elétricos de potência*: introdução ao estudo e aplicações. [S. l.]: SBSE, 2012.

ROMERO, R.; RIDER, M.; SILVA, I. J. A metaheuristic to solve the transmission expansion planning. *Power Systems, IEEE Transactions on*, Piscataway, v. 22, n. 4, p. 2289–2291, 2007.

SILVA, R. M. A.; RESENDE, M. G. C.; PARDALOS, P. M.; GONÇALVES, J. F. *Biased random-key genetic algorithm for bound-constrained global optimization*. [S. l.]: Proceedings of Gow, p. 133–136, 2012.

SILVA, R. M. de A.; RESENDE, M. G. C. *Algoritmo genético de chaves aleatórias viciadas para problemas de otimização global com restrições de caixa sujeitas a restrições não-lineares*. Natal: SBPO, 2013.

YANG, X.-S. *Engineering optimization*: an introduction with metaheuristic applications. New Jersey: John Wiley & Sons, 2010.

ZINI, É. de O. C. *Algoritmo genético especializado na resolução de problemas com variáveis contínuas e altamente restritos*. Dissertação (Mestrado em Engenharia Elétrica) — Faculdade de Engenharia de Ilha Solteira, Universidade Estadual Paulista, Ilha Solteira, 2009.