Pós-Graduação em Ciência da Computação

MARIANA ALVES MOURA

ALGORITMO GENÉTICO DE CHAVES ALEATÓRIAS SEGUNDO DISTRIBUIÇÃO DE LEVY PARA OTIMIZAÇÃO GLOBAL

Universidade Federal de Pernambuco
posgraduacao@cin.ufpe.br
http://cin.ufpe.br/~posgraduacao

Recife
2018

Mariana Alves Moura

ALGORITMO GENÉTICO DE CHAVES ALEATÓRIAS SEGUNDO DISTRIBUIÇÃO DE LEVY PARA OTIMIZAÇÃO GLOBAL

Trabalho apresentado ao Programa de Pós-graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Mestre em Ciência da Computação.

Orientador: Ricardo Martins de Abreu Silva

Recife
2018

Catalogação na fonte
Bibliotecária Monick Raquel Silvestre da S. Portes, CRB4-1217

M929a Moura, Mariana Alves
Algoritmo genético de chaves aleatórias segundo distribuição de Levy para otimização global / Mariana Alves Moura. – 2018.
101 f.: il., fig.

Orientador: Ricardo Martins de Abreu Silva.
Dissertação (Mestrado) – Universidade Federal de Pernambuco. CIn, Ciência da Computação, Recife, 2018.
Inclui referências e apêndices.

1. Ciência da computação. 2. Otimização global. I. Silva, Ricardo Martins de Abreu (orientador). II. Título.

004 CDD (23. ed.) UFPE- MEI 2018-101

Mariana Alves Moura

Algoritmo genético de chaves aleatórias segundo distribuição de Levy para otimização global

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Pernambuco, como requisito parcial para a obtenção do título de Mestre em Ciência da Computação

Aprovado em: 21/02/2018

BANCA EXAMINADORA

_____

Prof. Dr. George Darmiton da Cunha Cavalcanti
Centro de Informática / UFPE

_____

Prof. Dr. Rodrigo Gabriel Ferreira Soares
Departamento de Estatística e Informática/UFRPE

_____

Prof. Dr. Ricardo Martins de Abreu Silva
Centro de Informática / UFPE
(Orientador)

Dedico este trabalho aos meus pais Jozemal e Marlene, ao meu namorado Rafhael e ao meu orientador Ricardo, que me deram todo o apoio necessário para que eu conseguisse chegar até aqui.

AGRADECIMENTOS

Agradeço primeiramente a Deus por ter chegado até aqui.

Agradeço aos meus pais Marlene e Jozemal e à toda a minha família por todo suporte recebido.

Agradeço aos meus amigos por todo apoio e incentivo e por me ajudarem a acreditar que tudo iria dar certo no final.

Dedico também um agradecimento especial ao meu namorado Rafhael e à minha sogra Solange, que me acolheram em sua sempre que precisei, me apoiando e ajudando no que fosse possível.

Agradeço ao meu orientador Ricardo Martins por todo aprendizado que ele me proporcionou, não apenas com esta dissertação mas com todos os projetos que tive a oportunidade de participar durante os dois anos do mestrado.

Agradeço ao Centro de Informática da UFPE que disponibilizou a estrutura, seus professores e pesquisadores para que este e outros trabalhos pudessem ser concluídos com êxito.

Agradeço aos professores George Darmiton e Rodrigo Soares que aceitaram compor a banca deste trabalho.

Agradeço ao CNPq e à Motorola pelo suporte financeiro, permitindo que eu pudesse me dedicar ao meu mestrado sem maiores preocupações.

Por fim, agradeço à todos que, de alguma maneira, contribuíram para a minha formação.

"Não sabendo que era impossível, foi lá e fez."
(Mark Twain)

RESUMO

A otimização global tem como objetivo encontrar um mínimo ou máximo de uma função em um domínio discreto ou contínuo. Esta técnica possui aplicabilidade em diversas áreas do conhecimento, tais como física, biologia, engenharia, administração, economia, entre outras, que apresentam problemas passíveis de serem representados por meio de modelagens matemáticas. Uma classe de algoritmos utilizada para resolver este tipo de problema são os algoritmos genéticos, que se baseiam nos processos de evolução darwinistas para selecionar as melhores soluções dentro de uma população de soluções candidatas. O Algoritmo Genético de Chaves Aleatórias Viciadas, do inglês *Biased Random-Key Genetic Algorithm (BRKGA)*, proposto por Gonçalves e Resende (2011), é uma variação de algoritmos genéticos que consiste em representar as soluções de um problema como vetores de chaves reais definidas por valores gerados aleatoriamente no intervalo contínuo [0,1) e utiliza um decodificador determinístico para mapear estes vetores em soluções viáveis para o problema. Neste trabalho, foi investigado o impacto da substituição da distribuição uniforme, utilizada na geração de chaves aleatórias do BRKGA tradicional, pela utilização da distribuição de Levy (PAUL, 1937). Esta variação foi inserida no BRKGA tradicional e em uma hibridização do algoritmo com a inserção de um procedimento de busca local, proposto por Silva et al. (2013a). Posteriormente, as versões propostas foram comparadas com as versões da literatura e também com a metaheurística C-GRASP, que já possui resultados satisfatórios para este tipo de problema. Os experimentos foram realizados utilizando-se para tal um conjunto de funções de benchmark de otimização global unimodais e multimodais, utilizadas como funções-base para o *IEEE CEC Competition* e presentes também em diversos trabalhos da área de pesquisa. Os algoritmos foram comparados em termos de desempenho e qualidade das soluções e a variação proposta conseguiu alcançar resultados bastante competitivos em relação às demais técnicas.

Palavras-chave: Otimização Global. Computação Evolucionária. Algoritmos Genéticos. BRKGA. Distribuição de Levy. Metaheurísticas.

ABSTRACT

Global optimization aims to find a minimum or maximum of a function in a discrete or continuous domain. This technique has applicability in several areas of knowledge, such as physics, biology, engineering, administration, economics, among others, that present problems that can be represented through mathematical models. One class of algorithms used to solve this type of problem is genetic algorithms, which rely on Darwinian evolutionary processes to select the best solutions within a population of candidate solutions. Biased Random-Key Genetic Algorithm (BRKGA), proposed by Gonçalves e Resende (2011), is a variation of genetic algorithms that consists of representing the solutions of a problem as real-key vectors defined by randomly generated values in the continuous interval [0 , 1) and uses a deterministic decoder to map these vectors into feasible solutions to the problem. In this work, was investigated the impact of the substitution of the uniform distribution, used in the generation of traditional BRKGA random keys, by the use of the Levy distribution (PAUL, 1937). This variation was inserted in the traditional BRKGA and in a hybridization of the algorithm with the insertion of a local search procedure, proposed by Silva et al. (2013a). Subsequently, the proposed versions were compared with the literature versions and also with the C-GRASP metaheuristic, which already presents satisfactory results for this type of problem. The experiments were performed using a set of unimodal and multimodal global optimization benchmark functions, used as base functions for the IEEE CEC Competition and also present in several works of the research area. The algorithms were compared in terms of performance and quality of the solutions and the proposed variation managed to achieve very competitive results in relation to the other techniques.

Key words: Global Optimization. Evolutionary Computation. Genetic Algorithms. BRKGA. Levy Distribution. Metaheuristics.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ALGORITMOS

LISTA DE ABREVIATURAS E SIGLAS

ABC	Artificial Bee Colony
ACO	Ant Colony Optimization
AGS	Algoritmo Genético Simples
API	Application Programming Interface
BF	Bacterial Foraging
BRKGA	Biased Random-Key Genetic Algorithm
C-GRASP	Continuous Greedy Randomized Adaptive Search Procedure
CEC	Congress on Evolutionary Computation
CRAN	Comprehensive R Archive Network
CS	Cuckoo Search
DE	Differential Evolution
EDA	Estimation of Distribution Algorithm
GRASP	Greedy Randomized Adaptive Search Procedure
PSO	Particle Swarm Optimizer
RKGA	Random-Key Genetic Algorithm
SA	Simulated Annealing
TTT-Plot	Time to Target Plot

SUMÁRIO

1 INTRODUÇÃO

Muitos problemas reais podem ser modelados matematicamente como problemas de otimização global, despertando sempre o interesse de pesquisadores em desenvolver algoritmos eficientes para resolvê-los. Diversas técnicas são propostas na literatura para tentar encontrar boas soluções para este tipo de problema. Muitas delas buscam encontrar sempre a solução ótima, como é o caso da programação dinâmica (BELLMAN, 2013) e do *branch and bound* (LAWLER; WOOD, 1966), porém são limitadas a problemas de tamanho reduzido e/ou menos complexos. Outras técnicas são baseadas em métodos aproximativos, cujo objetivo é encontrar bons resultados, que se aproximem do ótimo, mas não garantem a otimalidade. Uma outra alternativa que tem sido amplamente utilizada na área de otimização global são as heurísticas e metaheurísticas, que tentam encontrar soluções de qualidade em um tempo computacional bom ou aceitável, guiadas por um conjunto de regras e métodos que governam a quantificação da proximidade das soluções em relação a um objetivo. Dentre estas últimas podemos citar alguns exemplos, tais como, como *Simulated Annealing* (KIRKPATRICK; GELATT; VECCHI, 1983), *Greedy Randomized Adaptive Search Procedure (GRASP)* (FEO; RESENDE, 1995) e Algoritmos Genéticos (GOLDBERG, 2006), entre outras.

Neste trabalho será utilizada a metaheurística Algoritmo Genético de Chaves Aleatórias Viciadas, do inglês *Biased Random-Key Genetic Algorithm (BRKGA)*, para o qual serão propostas melhorias a partir do uso da distribuição estatística de Levy.

1.1 OBJETIVOS E MOTIVAÇÃO

O principal objetivo deste trabalho consiste em analisar o impacto na qualidade e desempenho do BRKGA para problemas de otimização global com restrições de caixa via a utilização da distribuição de Levy na geração das suas chaves aleatórias em substituição à distribuição uniforme usada em sua versão canônica.

A distribuição de Levy foi escolhida devido à sua aplicabilidade em outros trabalhos, como por exemplo em Lee e Yao (2004), Yang e Deb (2009), Yang e Deb (2010), Gandomi, Yang e Alavi (2013), Yang e Deb (2014), entre outros, que propõem métodos de busca inspirados na natureza e apresentaram resultados promissores para problemas de otimização global. A hipótese levantada é de que a contribuição para os bons resultados destes métodos está principalmente no uso da distribuição de Levy e não apenas na metáfora bioinspirada subjacente a essas referências.

Outro motivo da escolha desta distribuição é a sua característica de se transformar em diversas outras distribuições de probabilidade de acordo com a configuração dos seus parâmetros, tais como a distribuição Normal e a distribuição de Cauchy, tornando-a mais

adaptável a problemas com características diferentes.

1.2 OTIMIZAÇÃO GLOBAL

O objetivo da otimização global é encontrar os valores do domínio de uma função $f : S \subset \mathbb{R}^n \to \mathbb{R}$, denominada **Função Objetivo**, que a maximiza ou minimiza, onde S representa o domínio ou espaço de soluções viáveis, com n variáveis de decisão (JAMIL; YANG, 2013).

Neste trabalho, os problemas serão representados como problemas de minimização, sem perda de generalidade, sujeitos a restrições de caixa, ou seja, as restrições são definidas como limites superiores e inferiores para cada variável de decisão:

$$\text{minimizar } f(\mathbf{x}) \tag{1.1}$$

sujeito a

$$l_i \leq x_i \leq u_i, \;\; i = 1, \ldots, n, \tag{1.2}$$

onde $\mathbf{x}$ representa uma solução para o problema com n variáveis de decisão e os vetores $\mathbf{l}$ e $\mathbf{u}$ contêm os limites superiores e inferiores destas variáveis. Isto significa que o domínio S é um hiper-retângulo $S = \{\mathbf{x} = (x_1, ..., x_n) \in \mathbb{R}^n\} : \mathbf{l} \leq \mathbf{x} \leq \mathbf{u}$, com $\mathbf{l}, \mathbf{u} \in \mathbb{R}^n$, tal que $\mathbf{l} \leq \mathbf{u}$.

Como instâncias de teste para fins de validação dos métodos propostos, serão utilizadas funções de *benchmark* de otimização global contínuas unimodais e multimodais presentes em diversos trabalhos da literatura da área e utilizadas como funções-base para as principais competições de otimização global, tais como a *IEEE CEC Competition*, que trata-se de uma competição entre algoritmos evolucionários realizada no Congress on Evolutionary Computation (CEC). Neste caso, o objetivo é tentar encontrar um $\mathbf{x_{min}} \subset S$, tal que:

$$f(\mathbf{x}_{min}) \leq f(\mathbf{x}^*) \tag{1.3}$$

onde a melhor solução conhecida na literatura é representada pelo vetor $\mathbf{x}^*$ e $f(\mathbf{x}^*)$ é o resultado da função objetivo aplicada a $\mathbf{x}^*$.

1.3 CONTRIBUIÇÕES

Este trabalho tem como principais contribuições:

1. O aperfeiçoamento do BRKGA por meio do uso da distribuição de Levy para a geração das chaves aleatórias;

2. A realização de um estudo comparativo em termos de qualidade e desempenho entre as abordagens do BRKGA propostas neste trabalho, utilizando distribuição de

Levy, e as abordagens tradicionais da literatura, cuja geração das chaves aleatórias é realizada da maneira tradicional (distribuição uniforme);

3. A validação da relevância da variação proposta, a partir da comparação com outra heurística da literatura, o C-GRASP, que apresenta bons resultados para problemas de otimização global com restrições de caixa.

1.4 ESTRUTURA DA DISSERTAÇÃO

Os próximos capítulos estão organizados da seguinte forma:

- No capítulo 2 será apresentada uma contextualização a respeito dos algoritmos genéticos e das suas variações de chaves aleatórias que embasaram esta pesquisa, o método de ajuste de parâmetros utilizado neste trabalho também será detalhado e será apresentado o estado da arte das metaheurísticas aplicadas a problemas de otimização global, assim como as pesquisas mais recentes na área.
- No capítulo 3 será detalhado o desenvolvimento do trabalho e a solução proposta nesta dissertação.
- No capítulo 4 serão apresentados os resultados obtidos pela proposta desenvolvida em comparação com as versões tradicionais já disponíveis na literatura, além do comparativo com a metaheurística C-GRASP.
- No capítulo 5 serão expostas as conclusões obtidas com a pesquisa e os trabalhos futuros.

2 REVISÃO DA LITERATURA

2.1 ALGORITMOS GENÉTICOS

Algoritmos Genéticos (GOLDBERG, 2006) constituem uma classe particular de Algoritmos Evolucionários que usam técnicas inspiradas pela biologia evolutiva darwinista (CHARLES, 1859), tais como hereditariedade, mutação, seleção natural e cruzamento. É uma metaheurística baseada no fundamento de que apenas os indivíduos mais adaptados são capazes de sobreviver a diversas gerações no decorrer do tempo devido à seleção natural. Neste processo, os indivíduos possuem características diferentes, determinadas a partir dos seus genes. Algumas destas características são favoráveis ao ambiente em que eles vivem enquanto outras podem não ser. Estas características são transmitidas aos seus descendentes por meio da hereditariedade, fazendo com que os indivíduos que herdam as melhores características consigam sobreviver por mais gerações.

AGs foram introduzidos por Holland (1975) e consistem em transferir um conjunto de indivíduos (ou soluções) mais aptos para uma nova população mediante um processo análogo ao de seleção natural, realizando-se alterações nestes indivíduos por meio de operadores de cruzamento e mutação em seus cromossomos. A ideia é que ao longo das iterações, os piores indivíduos sejam descartados e na população restem apenas os melhores. Um cromossomo pode ser representado por um vetor, onde cada componente do vetor pode ser compreendido como uma característica do indivíduo ou seu gene. Cada gene pode assumir um instância de uma determinada característica, que na biologia chamamos de alelo. Nos algoritmos genéticos é feita uma analogia entre os termos utilizados na biologia evolutiva com os termos da otimização. Uma ilustração desta analogia pode ser observada na Figura 1 para um melhor entendimento.

Como é demonstrado na Figura 1, uma solução do problema é denominada de indivíduo. Para cada indivíduo está associado um cromossomo, que é um vetor de n entradas representando uma solução. Cada entrada do vetor é chamada de gene e o valor contido no gene recebe o nome de alelo. A posição que cada gene ocupa no cromossomo é chamada de *locus*, logo a primeira entrada do vetor será denominada de gene no primeiro locus. Um conjunto de cromossomos recebe o nome de população.

O operador de seleção escolhe quais cromossomos participarão do processo de cruzamento, que gerará novos cromossomos a partir da combinação de suas características. Geralmente, os cromossomos mais aptos possuem maior probabilidade de serem selecionados, transmitindo suas características para as próximas gerações. Em outras palavras, de acordo com a aplicação da função de avaliação, é atribuído um valor de adaptação (*fitness*) para cada solução e as soluções que apresentarem melhor adaptação possuirão mais chances de transmitirem suas características para novas soluções no cruzamento.

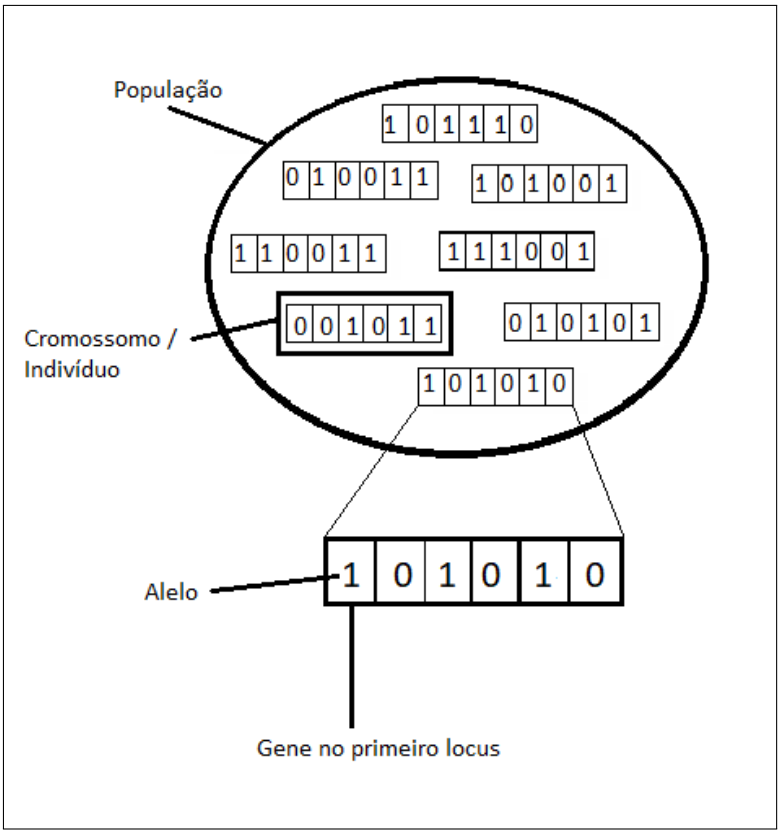


Figura 1 – Terminologia utilizada nos Algoritmos Genéticos.

O operador de cruzamento troca partes de duas soluções, imitando a reprodução biológica de dois organismos, em que os descendentes irão herdar características dos dois pais. Um exemplo de cruzamento, chamado de cruzamento de um ponto, está representado na Figura 2. Nesta técnica é determinado um ponto, que chamamos de ponto de corte, o qual dividirá as soluções em duas partes. Serão gerados dois descendentes e cada um receberá uma parte do primeiro pai e outra parte do segundo.

O operador de mutação pode alterar os valores de alguns alelos escolhidos aleatoriamente, gerando diversidade na população e evitando que o algoritmo convirja rapidamente para um ótimo local. Considerando como exemplo um Algoritmo Genético Simples (AGS), em que os cromossomos são representados por vetores binários, um tipo de operador de mutação consiste em selecionar aleatoriamente uma posição do cromossomo, e em seguida, alterar o valor desta posição. No caso de um vetor binário, se o alelo tiver valor 0, será trocado por 1 e vice-versa. Na Figura 3 podemos verificar este procedimento graficamente. Neste exemplo, a segunda posição do vetor foi selecionada e teve o valor do seu alelo alterado de 1 para 0.

Por fim, para mensurar a adaptação, é necessário definir uma função de aptidão que atribui um determinado valor (*fitness*) para cada cromossomo, a partir das informações contidas em seus genes.

Um pseudocódigo do funcionamento de um algoritmo genético tradicional pode ser

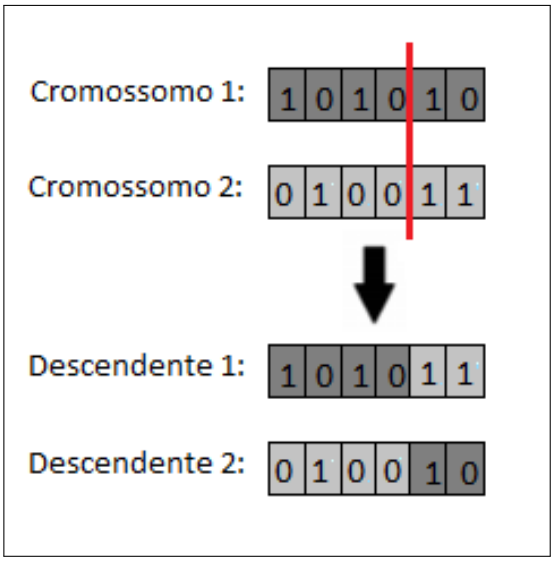


Figura 2 – Exemplo de cruzamento de um ponto.

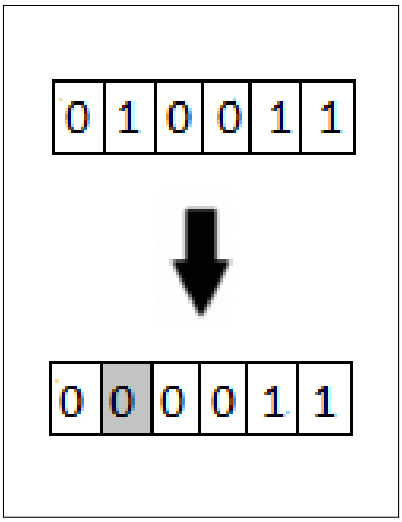


Figura 3 – Exemplo de operação de mutação.

visualizado no Algoritmo 1.

Na linha 2, a população inicial P é inicializada. Em seguida, nas linhas 3 a 9, ocorre o processo evolucionário: na linha 4 é calculada a aptidão de cada indivíduo da população, na linha 5 é aplicado o operador de seleção de pais para participarem do cruzamento na linha 6, na linha 7 o algoritmo realiza a mutação e na linha 8 a população é atualizada para a próxima geração. Este procedimento é repetido até que um determinado critério de parada seja atingido. Este critério pode ser determinado por um número máximo de gerações (iterações), número limite de iterações sem melhoria, entre outros. Por fim, na linha 10 o algoritmo retorna a melhor solução encontrada.

Um problema que pode ocorrer nos algoritmos genéticos após os operadores de variação (cruzamento e mutação) é a geração de soluções inviáveis para um determinado problema. No caso do cruzamento de um ponto demonstrado na figura 4, por exemplo, se estivésse-

Algoritmo 1 Pseudocódigo genérico do Algoritmo Genético

```
1: método Algoritmo Genético
2:     Inicialize a população inicial P;
3:     enquanto Critério de parada não satisfeito faça
4:         Calcule a aptidão de cada indivíduo em P;
5:         Selecione os pais para o cruzamento;
6:         Realize o cruzamento;
7:         Realize a mutação;
8:         Atualize a população;
9:     fim enquanto
10:    retorna Melhor indivíduo da população
11: fim método
```

mos tratando de um problema de sequenciamento, onde uma solução é representada pela permutação de alguns valores sem repetições, os dois descendentes gerados seriam soluções inviáveis pois em ambos os casos ocorreram repetições. Para contornar este problema, muitos autores desenvolveram soluções fortemente dependentes dos problemas que pretendiam resolver, como podemos observar em Goldberg, Lingle et al. (1985), Grefenstette et al. (1985), Grefenstette (1987), Cleveland e Smith (1989). Visando criar uma alternativa genérica que resolvesse este problema da inviabilidade, Bean (1994) propôs uma estratégia robusta de chaves aleatórias apresentada na próxima seção.

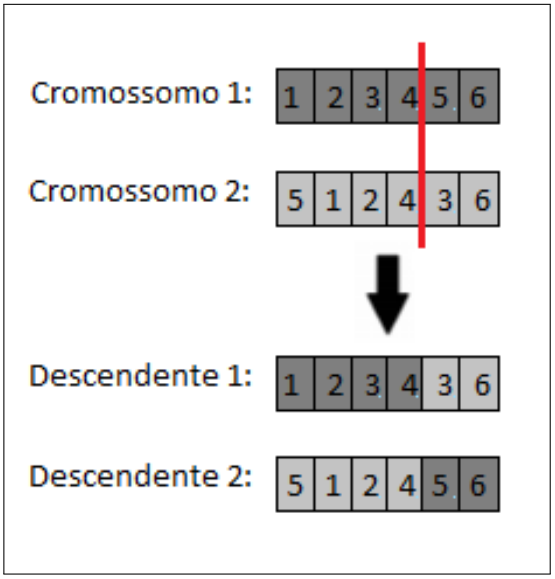


Figura 4 – Exemplo de cruzamento de um ponto gerando soluções inviáveis.

2.1.1 Algoritmos Genéticos de Chaves Aleatórias

Um Algoritmo Genético de Chaves Aleatórias, do inglês *Random-Key Genetic Algorithm (RKGA)*, é uma metaheurística evolucionária para problemas de otimização descrita por Bean (1994). O RKGA consiste em representar soluções como um vetor de n chaves

aleatórias, no qual cada chave é um número real definido por um valor gerado aleatoriamente segundo distribuição uniforme no intervalo contínuo [0,1).

As soluções (cromossomos) representadas pelos vetores de chaves aleatórias são submetidas a um decodificador. O decodificador é um algoritmo determinístico que recebe como entrada um cromossomo e mapeia-o numa solução do problema de otimização no espaço de soluções do problema. Em seguida, o decodificador calcula o custo (ou a aptidão) desta solução e retorna o valor deste custo para o BRKGA. Uma ilustração do mapeamento das chaves é demonstrada na Figura 5 onde do lado esquerdo está representado o espaço de chaves aleatórias e do lado direito o espaço de soluções do problema. O decodificador relaciona uma chave aleatória a um vetor solução e calcula sua aptidão.

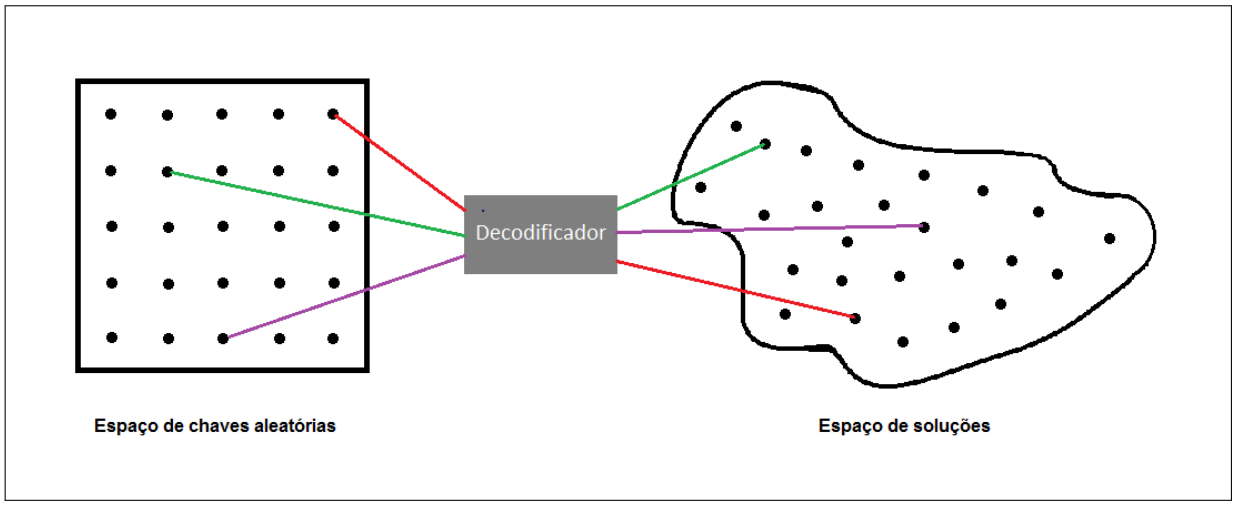


Figura 5 – Processo de mapeamento de chaves aleatórias em soluções por meio do decodificador.

Para exemplificar o procedimento de decodificação, consideremos um exemplo em que nosso espaço de soluções é determinado por vetores binários de tamanho 6 e possua uma restrição que obrigue que 3 posições dos vetores devam receber o valor 1. Um decodificador para este problema está descrito no Algoritmo 2.

Algoritmo 2 Exemplo de Decodificador

1: **método** DECODIFICADOR(*cromossomo*)
2: Ordene em ordem crescente o vetor inicial, representado pelo *cromossomo*;
3: Verifique no vetor inicial os índices dos 3 primeiros valores do vetor ordenado;
4: Defina um novo vetor de tamanho 6 e atribua o valor 1 nos 3 índices verificados no passo anterior;
5: Defina o valor 0 para as demais posições do novo vetor;
6: Calcule a aptidão do cromossomo a partir deste novo vetor;
7: **retorna** aptidão do cromossomo
8: **fim método**

Neste procedimento, o decodificador recebe como parâmetro o vetor de chaves aleatórias

(ou o cromossomo) na linha 1, ordena-o na linha 2 e na linha 3 verifica os índices ocupados pelos 3 primeiros valores do vetor ordenado no vetor inicial. Nas linhas 4 e 5, um novo vetor é gerado e recebe o valor 1 nos 3 índices do passo anterior e 0 nas demais posições. Este novo vetor é a decodificação do vetor de chaves aleatórias em uma solução do problema e como pode ser observado, o procedimento sempre gerará soluções viáveis. Este exemplo está ilustrado numericamente na Figura 6. Por fim, o decodificador calcula a aptidão do cromossomo na linha 6 a partir do valor da solução decodificada para o problema e retorna esta aptidão para o RKGA. Ainda utilizando o exemplo da Figura 6, suponhamos que a aptidão do indivíduo é determinada pela soma dos índices das posições do vetor decodificado que receberam valor 1. Neste caso, o decodificador ilustrado retornaria o valor 12 (2 + 4 + 6) para o RKGA. Desta forma, cada indivíduo da população sempre terá uma aptidão relacionada a ele e uma representação viável no espaço de soluções do problema.

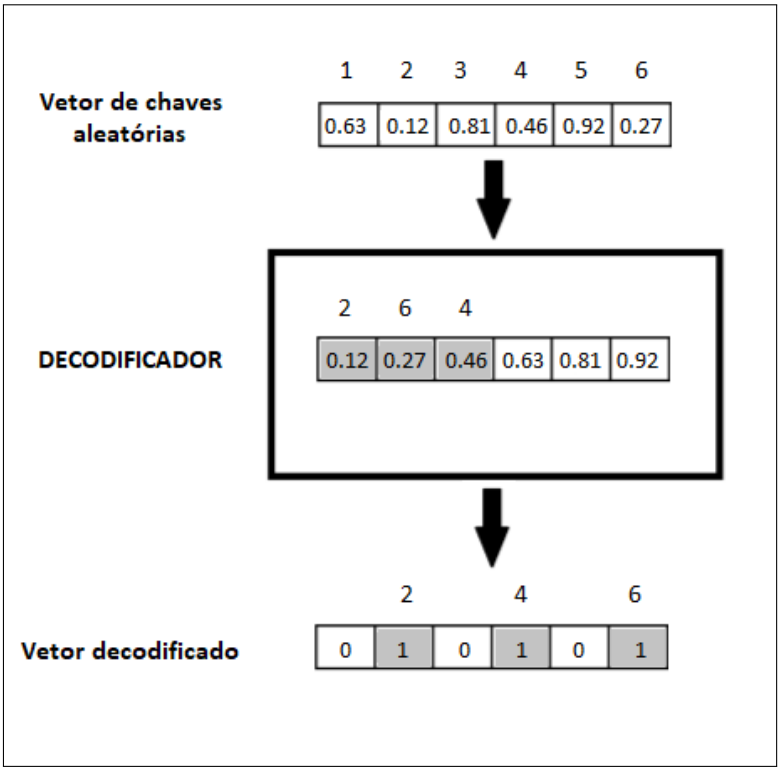


Figura 6 – Exemplo do processo de decodificação de uma chave aleatória no BRKGA.

Os operadores de variação (mutação e cruzamento) do RKGA são aplicados no vetor de chaves aleatórias e não no vetor decodificado. Este último só é utilizado no ambiente do decodificador para gerar a representação do problema e calcular a aptidão (ou *fitness*) do indivíduo. Com esta estratégia evita-se a produção de soluções inviáveis.

A evolução da população (ou conjunto de p vetores de chaves aleatórias) é feita com base no princípio darwinista, no qual os indivíduos mais aptos possuem mais chances de perpetuar suas características (material genético) pelas próximas gerações, ou seja, tem mais chances de serem selecionados para as operações de cruzamento e para serem sobreviventes.

Na figura 7 podemos ver que ao final de cada geração, os p vetores (ou indivíduos) da população são divididos em dois conjuntos: um conjunto menor $pe < p/2$, formado pelas melhores soluções e chamado de conjunto elite, e o restante da população chamado de conjunto não-elite. Todos os elementos do conjunto elite são copiados para a próxima geração, caracterizando o princípio darwinista do elitismo. Em seguida, uma quantidade pm, definida como parâmetro, de vetores mutantes é gerada para compor também a próxima geração, inserindo diversidade na população. Um mutante é simplesmente um vetor de chaves aleatórias geradas da mesma maneira que é gerada a população inicial. Por fim, para completar os p elementos da população posterior, os $p - pe - pm$ vetores restantes são gerados combinando pares de indivíduos da população atual, ambos escolhidos aleatoriamente, utilizando o cruzamento uniformemente parametrizado de Spears e Jong (1995), descrito na Figura 8.

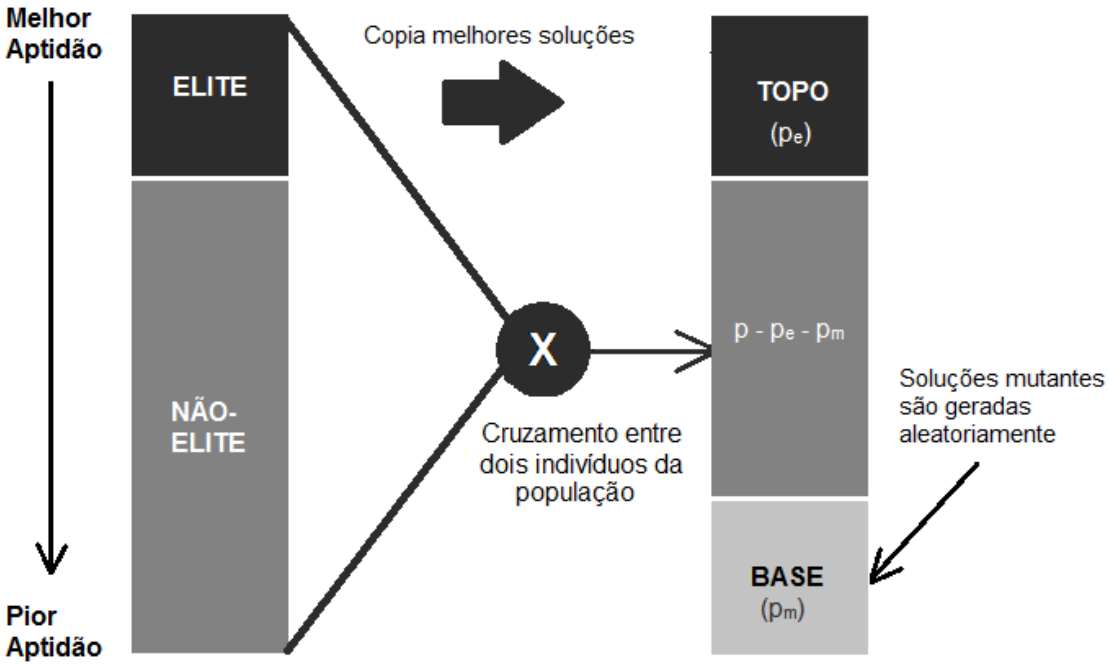


Figura 7 – Transição da geração k para a geração k+1 de um RKGA.

Neste tipo de cruzamento, é determinado um número aleatório para cada posição dos cromossomos, que será comparado com uma probabilidade ρ_a, definida como um parâmetro do algoritmo, de ser selecionado um alelo do primeiro pai (a) para a prole. No exemplo da Figura 8, considerando $\rho_a = 0.7$, se o número aleatório for menor que 0.7, a prole receberá o alelo do pai a. Caso contrário, receberá o alelo do pai b.

Por fim, o algoritmo é executado até que um critério de parada seja atendido e retornará a melhor solução encontrada para o problema.

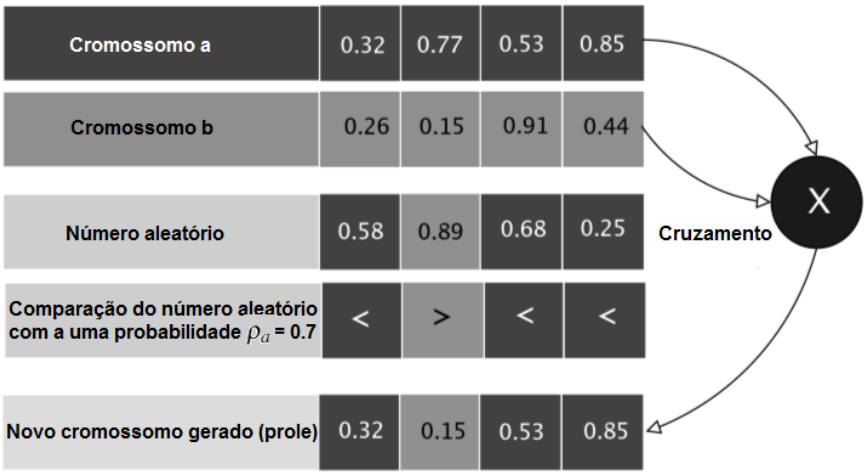


Figura 8 – Cruzamento uniformemente parametrizado.

2.1.2 Algoritmos Genéticos de Chaves Aleatórias Viciadas

O BRKGA é uma metaheurística proposta por Gonçalves e Resende (2011) como uma variante do RKGA, cuja diferença está na seleção dos pais para a realização do cruzamento. Enquanto no RKGA os dois pais são selecionados aleatoriamente da população p inteira, no BRKGA um dos pais é sempre selecionado do conjunto elite (pe) enquanto o outro pai é selecionado do conjunto não-elite. Esta característica torna o BRKGA mais enviesado ao elitismo. A repetição de um pai na seleção para o cruzamento é permitida, portanto, um mesmo indivíduo pode produzir mais de uma prole. Uma vez que é exigido que $pe < p/2$, a probabilidade de um indivíduo de elite ser selecionado para o cruzamento ($1/pe$) é maior que a de um indivíduo não-elitista ($1/p - pe$) e, portanto, os indivíduos de elite têm maior probabilidade de passar suas características para as gerações posteriores.

Na Figura 9 é ilustrada a dinâmica da evolução do BRKGA, que é bastante semelhante à do RKGA. A população é ordenada e dividida em conjunto elite e conjunto não-elite, o conjunto elite é copiado para a próxima geração e p_m indivíduos são gerados aleatoriamente para também comporem a próxima geração. A modificação acontece no procedimento de seleção dos pais para o cruzamento, em que o primeiro pai é sempre selecionado do conjunto elite.

O BRKGA, assim como o RKGA, utiliza o método de cruzamento uniformemente parametrizado de Spears e Jong (1995) descrito na seção anterior com a probabilidade ρ_a maior que 0.5. Dessa forma, como no BRKGA o primeiro pai (a) é sempre selecionado do conjunto elite, e sabendo que a probabilidade de cada gene do pai a ser transmitido para a prole é dada pelo parâmetro $\rho_a > 1/2$, a prole sempre terá maior probabilidade de herdar os genes do pai elitista, enquanto no RKGA isto não é uma regra.

Um pseudocódigo do funcionamento geral do BRKGA pode ser visto no Algoritmo 3. Inicialmente, a população P é inicializada na linha 2. Em seguida, inicia-se o processo

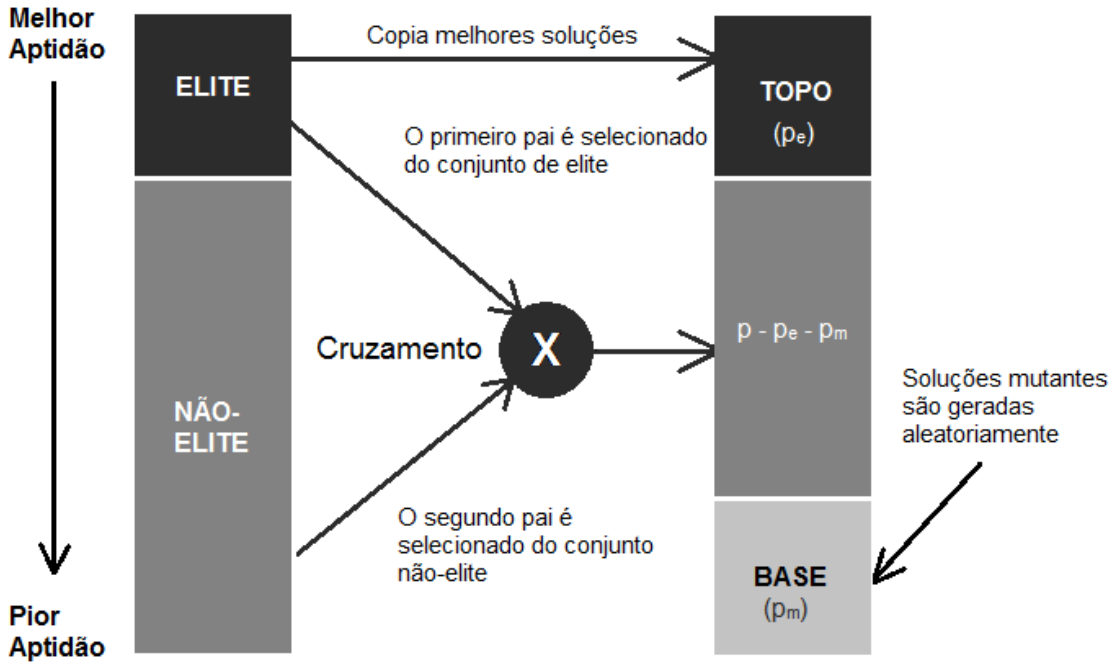


Figura 9 – Transição da geração k para a geração k+1 de um BRKGA.

evolucionário do BRKGA, descrito nas linhas 3 a 12. Na linha 4 é calculada a aptidão de cada indivíduo. Este procedimento de cálculo da aptidão ocorre por meio do decodificador, que receberá cada um dos cromossomos da população P, realizará o processo de decodificação e calculará a aptidão do indivíduo. Na linha 5 a população é ordenada de acordo com os valores de aptidão dos indivíduos e, em seguida, a população é dividida em conjunto elite e conjunto não-elite (linha 6). Na linha 7 os indivíduos do conjunto elite são copiados para a próxima geração, enquanto na linha 8 é realizada a seleção de pais para o cruzamento; e nas linhas 9 e 10 os operadores de cruzamento e mutação são aplicados. Por fim, a população é atualizada para a próxima geração (linha 11). Este procedimento evolucionário é repetido até que um determinado critério de parada seja satisfeito e o algoritmo retornará a melhor solução encontrada (linha 13).

O BRKGA, é baseado em um *framework* de metaheurísticas para propósitos gerais, no qual há uma divisão clara entre uma parte dependente e uma parte independente do problema. Neste *framework*, descrito na Figura 10, a parte independente do algoritmo não possui conhecimento acerca do problema a ser resolvido, limitando-se a realizar uma busca ao domínio das chaves aleatórias. Esta parte é composta pela geração das chaves aleatórias (cromossomos) e pela execução do processo evolucionário do algoritmo, onde os cromossomos são submetidos aos operadores genéticos. A conexão do BRKGA com o problema é feita pela parte dependente do algoritmo, na qual um decodificador produz as

Algoritmo 3 Pseudocódigo do BRKGA

```
1: método BRKGA
2:     Inicialize a população inicial P;
3:     enquanto Critério de parada não satisfeito faça
4:         Calcule a aptidão de cada indivíduo em P;          ▷ Decodificador
5:         Ordene a população P em ordem crescente de seus valores de aptidão;
6:         Particione P em dois conjuntos: elite e não-elite;
7:         Copie os indivíduos do conjunto elite da população atual para a próxima geração;
8:         Selecione um pai do conjunto elite para o cruzamento e outro pai da população
   inteira;
9:         Realize o cruzamento uniforme parametrizado;
10:        Realize a mutação;
11:        Atualize a próxima população;
12:    fim enquanto
13:    retorna Melhor indivíduo da população
14: fim método
```

soluções para o problema a partir das chaves aleatórias e calcula a aptidão destas soluções. Sendo assim, para especificar uma heurística BRKGA para um determinado problema é necessário apenas definir um decodificador que traduza as chaves aleatórias em soluções para o problema e o cálculo da aptidão.

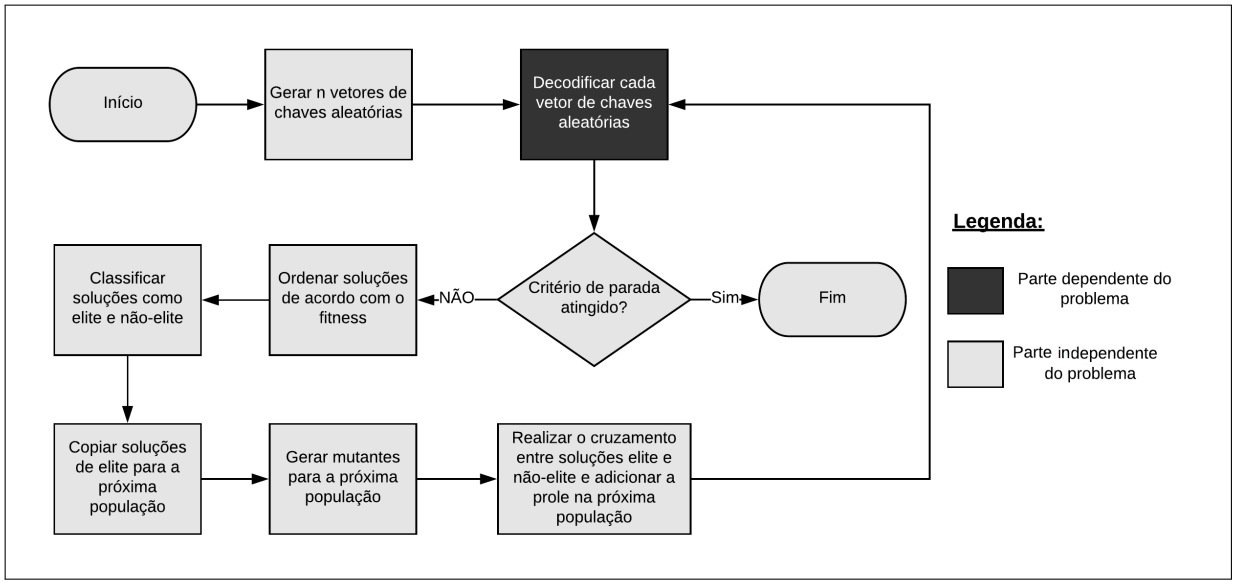


Figura 10 – Fluxograma do Algoritmo Genético de Chaves Aleatórias Viciadas.

A principal vantagem em usar um Algoritmo Genético de Chaves Aleatórias, seja o RKGA ou o BRKGA, em vez do Algoritmo Genético clássico está na possibilidade de reuso da maior parte do código em futuras implementações, já que o processo evolucionário não depende do problema a ser tratado, diferentemente do AG tradicional, em que é preciso definir diferentes operadores de cruzamento e mutação para cada problema a ser resolvido, a fim de evitar soluções inviáveis. Para o BRKGA, além desta vantagem do reuso, a mudança

inserida em relação ao RKGA, apesar de pequena, trouxe um desempenho considerável ao algoritmo de acordo com os experimentos demonstrados em (GONÇALVES; RESENDE, 2011) e (GONÇALVES; RESENDE; TOSO, 2014). Segundo os autores, a justificativa da melhoria é que o fato de um pai sempre ser selecionado do conjunto de elite para o cruzamento adiciona à abordagem uma característica de ganância (ou gula), semelhante ao que foi feito na heurística semi-gulosa de (HART; SHOGAN, 1987) e no GRASP (FEO; RESENDE, 1995), que propuseram a adição de procedimentos gulosos a um método construtivo aleatório puro, ambos resultando em soluções muito melhores, em média, do que uma construção puramente aleatória. Suas desvantagens, assim como nos algoritmos genéticos tradicionais, estão na quantidade grande de parâmetros a serem configurados e, por se tratar de um método populacional, exige um custo computacional alto, sendo mais recomendado para problemas mais complexos e multimodais. Uma observação importante é que, a depender da maneira como é construído o decodificador, o custo algorítmico pode aumentar, sendo a tarefa de projetá-lo essencial para que seja alcançado um bom desempenho.

2.2 AJUSTE AUTOMÁTICO DE PARÂMETROS

Boa parte dos algoritmos de otimização requer que uma grande quantidade de parâmetros seja ajustada para que possa ser alcançado um bom desempenho na sua execução. Muitas vezes estes ajustes são feitos de maneira rudimentar, até que se encontre uma configuração aceitável para a resolução do problema. Uma vez que a quantidade de parâmetros aumenta, o número de combinações possíveis destes parâmetros também aumenta, dificultando o seu ajuste.

Para atender este problema, diversas técnicas foram propostas ao longo dos anos, a fim de automatizar o processo de configuração de parâmetros da melhor maneira possível. Entre estes métodos encontra-se o *Iterated Racing*, que é baseado em procedimentos de corrida (*racing*) e tem sido utilizado com sucesso para configurar automaticamente vários algoritmos (LÓPEZ-IBÁÑEZ et al., 2016). Neste capítulo nos restringiremos à apresentação do método *Iterated Racing*.

2.2.1 *Iterated Racing*

O *Iterated Racing* é uma técnica de ajuste de parâmetros utilizado em diversos trabalhos recentes para diferentes problemas da literatura, tais como caixeiro viajante com janelas de tempo (LÓPEZ-IBÁÑEZ et al., 2013), alocação de *slots* (PELLEGRINI; CASTELLI; PESENTI, 2012), diagramas de fluxo (BENAVIDES; RITT, 2015), posicionamento de máquinas virtuais (STEFANELLO et al., 2015), empacotamento de caixas *online* (YARIMCAM et al., 2014), binarização de imagens (MESQUITA et al., 2015), problemas de roteamento e empacotamento combinados (CESCHIA; SCHAERF; STÜTZLE, 2013), seleção de rotas em tempo real (SAMA et al., 2016), rebalanceamento do compartilhamento de bicicletas (DELL et al., 2016),

planeamento energético (JACQUIN; JOURDAN; TALBI, 2014), organização de tabela de horário de aulas (NANNEN; EIBEN, 2006), discretização de séries temporais (ACOSTA-MESA et al., 2014), construção de máquinas de estado finito (CHIVILIKHIN; ULYANTSEV; SHALYTO, 2016), entre outros.

A corrida (*race*) foi inicialmente descrita por Maron e Moore (1997) como uma técnica de aprendizagem de máquina, que determina que um modelo é melhor que outro (ou o modelo que "ganha"a corrida sobre outro) por meio de limites estatísticos. Posteriormente ela foi adaptada para o contexto de configuração de parâmetros de algoritmos de otimização por Birattari et al. (2002).

O *Iterated Racing* é composto por três etapas:

1. Amostragem de novas configurações de acordo com uma distribuição normal truncada para parâmetros contínuos ou uma função de probabilidade discreta para parâmetros categóricos;

2. Seleção das melhores configurações das novas amostras por meio da técnica de corrida (*racing*);

3. E atualização da distribuição de amostragem a fim de que as melhores configurações tenham a maior probabilidade de serem selecionadas.

Esses três passos são repetidos até que um critério de parada seja atingido.

No *Iterated Racing*, cada parâmetro configurável está associado a uma distribuição amostral que é independente da distribuição amostral dos demais parâmetros. Além disso, esta técnica considera possíveis condições e restrições que possam existir entre os parâmetros. A distribuição utilizada para gerar amostras de parâmetros contínuos é uma normal truncada. Para parâmetros categóricos é utilizada uma função de probabilidade discreta definida em López-Ibáñez et al. (2016) e López-Ibáñez et al. (2016). Parâmetros ordinais são considerados como inteiros. A atualização da distribuição consiste em alterar os parâmetros de média e desvio padrão no caso da distribuição normal e o valor da probabilidade no caso da distribuição discreta. A adaptação da amostragem se dá a partir das atualizações das distribuições em cada iteração do algoritmo com base nas melhores configurações encontradas até o momento, as quais aumentam a probabilidade de amostragem dos parâmetros pertencentes a estas configurações mais promissoras, denotando um tipo de elitismo.

Depois que é criada uma amostra com novas configurações, as melhores configurações são selecionadas por meio de *racing*, que é executado até que se alcance um número mínimo de configurações sobreviventes, um máximo número de instâncias usadas ou um limite computacional B predefinido, que pode ser determinado pelo tempo computacional ou por um número máximo de experimentos, onde cada experimento implica em uma aplicação de uma configuração de parâmetros em uma instância.

Uma análise mais detalhada do *Iterated Racing* pode ser observado no pseudocódigo descrito no Algoritmo 4. Na linha 1, algoritmo recebe como entrada: um conjunto de instâncias I, amostrado do espaço de instâncias $\mathcal{I}$, sobre as quais o algoritmo a ser configurado será executado utilizando as configurações de parâmetros candidatas; o espaço de parâmetros X, que serão configurados automaticamente via *Iterated Racing*; uma função de custo C que determinará a qualidade de uma configuração; e o limite computacional B que pode ser definido como um tempo limite de execução ou por um número máximo de experimentos. Na primeira iteração, o conjunto inicial de configurações candidatas é gerado por meio de amostragem uniforme do espaço de parâmetros X (linha 2). Em seguida, as melhores configurações são determinadas a partir de um *race* (linha 3). Em cada iteração deste procedimento de *race*, as configurações são aplicadas a uma instância e são avaliadas de acordo com o valor médio da medida de custo C. Depois os resultados são comparados por meio de um teste estatístico, que pode ser o teste de Friedman ou o teste t de *Student*. Se houver evidências estatísticas suficientes de que algumas configurações candidatas obtiveram melhor desempenho que outras, as piores configurações são descartadas e as melhores continuam no *race* e serão executadas na próxima instância. Em cada uma das próximas iterações, definidas nas linhas 4 a 10, um novo conjunto de configurações candidatas é gerado por meio de uma distribuição amostral que foi atualizada a partir do conjunto de melhores soluções da iteração anterior (linha 7). Na linha 8 é formado um conjunto por meio da união das novas configurações geradas com as melhores configurações da iteração anterior e na linha 9, o *race* é executado novamente a fim de ser determinado o novo conjunto de melhores soluções. O procedimento é executado até que o limite computacional predefinido seja atingido e o conjunto de melhores configurações é retornado pelo algoritmo.

Algoritmo 4 Pseudocódigo do Iterated Racing

1: **método** IteratedRacing($I = \{I_1, I_2, ...\} \sim \mathcal{I}, X, C(\theta, i) \in \mathbb{R}, B$)
2: $\quad\Theta_i \leftarrow GerarAmostraUniforme(X)$
3: $\quad\Theta_{elite} \leftarrow Race(\Theta_1, B_1, C)$
4: $\quad j \leftarrow 1$
5: $\quad$**enquanto** $B^{used} \leq B$ **faça**
6: $\quad\quad j \leftarrow j + 1$
7: $\quad\quad\Theta_{new} \leftarrow GerarAmostra(X, \Theta_{elite})$;
8: $\quad\quad\Theta_j \leftarrow \Theta_{new} \cup \Theta_{elite}$
9: $\quad\quad\Theta_{elite} \leftarrow Race(\Theta_j, B_j)$;
10: $\quad$**fim enquanto**
11: $\quad$**retorna** Θ_{elite}
12: **fim método**

2.3 METAHEURÍSTICAS APLICADAS À PROBLEMAS DE OTIMIZAÇÃO GLOBAL

Nesta seção estão descritas algumas das principais metaheurísticas aplicadas à problemas de otimização global. Um resumo dos trabalhos relacionados pode ser visualizado na tabela 1.

Tabela 1 – Metaheurísticas aplicadas à problemas de otimização global

Referência	**Técnica**
(CORANA et al., 1987), (DEKKERS; AARTS, 1991), (GOFFE; FERRIER; ROGERS, 1994)	*Simulated Annealing* - Metaheurística baseada na termodinâmica, inspirada no processo de recozimento e resfriamento de metais.
(EBERHART; KENNEDY, 1995)	*Particle Swarm Optimization (PSO)* - Metaheurística baseada no comportamento coordenado de cardume de peixes e bandos de pássaros para a busca de alimentos.
(SOCHA; DORIGO, 2008), Dorigo e Birattari (2011)	*Ant Colony Optimization (ACO)* - Algoritmo baseado no comportamento inteligente de colônias de formigas para a busca de alimentos.
(KARABOGA; BASTURK, 2007)	Algoritmo de Colônias de Abelhas Artificiais baseado no comportamento inteligente das colmeias de abelhas.
(YANG; DEB, 2009)	*Cuckoo Search (CS)* - Algoritmo baseado no comportamento parasitário de ninhada de algumas espécies de pássaros cuco.
(ANDRE; SIARRY; DOGNON, 2001), (KIM; ABRAHAM; CHO, 2007)	Algoritmos Genéticos aplicados a problemas de otimização global.
(SILVA et al., 2013a)	Hibridização de um algoritmo genético de chaves aleatórias viciadas (BRKGA) por meio da inserção de um procedimento de busca local cuja vizinhança de uma solução é determinada pela projeção de pontos numa hiperesfera com centro na solução.

Continua na próxima página

REFERÊNCIA	**TÉCNICA**
(HIRSCH; PARDALOS; RESENDE, 2010)	*Continuous GRASP (C-GRASP)* - Uma adaptação do GRASP (FEO; RESENDE, 1995) para lidar com problemas de otimização contínuos. O GRASP é uma metaheurística para problemas de otimização combinatória de variáveis inteiras, que combina um método construtor para criar as soluções iniciais e uma busca local para tentar melhorar a qualidade das soluções. O C-GRASP utiliza a mesma abordagem, porém para espaços de busca contínuos.
(STORN; PRICE, 1997)	Evolução Diferencial - Algoritmo Evolucionário que gera novos indivíduos por meio da adição da diferença ponderada entre dois indivíduos a um terceiro indivíduo, gerando uma mutação.
(LARRAÑAGA; LOZANO, 2001)	Algoritmo de Estimação da Evolução - Metaheurística que utiliza técnicas de estimação da distribuição de probabilidade da população em vez de operadores genéticos.
(SUN; ZHANG; TSANG, 2005)	DE/EDA - Combinação de duas técicas de Computação Evolucionária: Evolução Diferencial (STORN; PRICE, 1997) e Algoritmo de Estimação da Evolução (LARRAÑAGA; LOZANO, 2001).

Uma das primeiras metaheurísticas idealizadas para o problema de otimização global foi o *Simulated Annealing (SA)* (METROPOLIS et al., 1953), inspirado no processo da termodinâmica de recozimento e resfriamento de metais, cuja principal vantagem em relação a outros métodos da época foi evitar a estagnação da busca em mínimos, haja vista que o algoritmo pode aceitar soluções piores do que a solução corrente de acordo com uma determinada probabilidade. Devido a esta característica, o SA tornou-se uma boa alternativa para problemas de otimização global multimodais e motivou sua aplicação em diferentes trabalhos nesta área, como em Corana et al. (1987) e Dekkers e Aarts (1991) em que foi aplicado a funções multimodais de variáveis contínuas e posteriormente em Goffe, Ferrier e Rogers (1994), onde foi utilizado para otimizar funções estatísticas aplicadas à economia.

Uma classe de algoritmos bioinspirados que também apresenta soluções de busca interessantes para resolver problemas de otimização global são os métodos baseados em enxames. Estes algoritmos se assemelham aos algoritmos evolucionários por empregarem o conceito de população de soluções como uma estratégia de exploração do espaço de busca. O precursor dentre estes algoritmos é o *Particle Swarm Optimizer (PSO)* proposto por Eberhart e Kennedy (1995), inspirado em estratégias de organização de cardumes

de peixes ou voos de pássaros para coordenar seu processo de busca. Posteriormente, Dorigo, Maniezzo e Colorni (1996) propôs o *Ant Colony Optimization (ACO)*, algoritmo baseado no processo de busca de alimentos de colônias de formigas aplicado a problemas de otimização combinatória. Posteriormente este algoritmo foi estendido para problemas de otimização global em domínios contínuos por Socha e Dorigo (2008) e também foi utilizado em Dorigo e Birattari (2011). Mais adiante foi idealizado o *Artificial Bee Colony (ABC)* (KARABOGA; BASTURK, 2007), inspirado no comportamento inteligente das colmeias de abelhas para problemas de otimização global multimodais. Estes três algoritmos de enxames são os principais representantes desta classe, porém na literatura podemos encontrar diversos outros métodos com ideias bioinspiradas. Dentre as propostas mais recentes está o *Cuckoo Search (CS)* (YANG; DEB, 2009), uma técnica de otimização global baseada no comportamento parasitário de ninhada realizado por algumas espécies de pássaros cuco, cujo procedimento de busca é coordenado por caminhadas aleatórias baseadas em voos de Levy, um padrão que segue regras da distribuição probabilística de Levy para explorar o espaço de busca do problema.

Como vimos na Seção 2.1, o algoritmo genético é uma metaheurística introduzida por Holland (1975) que baseia-se em ideias da teoria da evolução darwinista para encontrar boas soluções para problemas de otimização. Em Andre, Siarry e Dognon (2001) foi realizado um estudo do algoritmo genético clássico aplicado a um conjunto de problemas de otimização global e foi observado que ele apresentou pouca convergência em problemas maiores e multimodais, além de exigir um tempo computacional grande. Visando reduzir estes problemas, no mesmo trabalho Andre, Siarry e Dognon (2001) propôs algumas melhorias que consistiram em ajustes mais refinados dos parâmetros do AG, na inclusão de uma redução adaptativa no intervalo de cada variável e no uso de um fator de escala para o cálculo da probabilidade de cruzamento. As alterações propostas conseguiram melhorar consideravelmente o AG clássico em termos de convergência a ótimos globais.

Em Kim, Abraham e Cho (2007) também foi proposta uma variação do algoritmo genético para o problema de otimização global envolvendo a sua hibridização com um algoritmo baseado na alimentação bacteriana, o *Bacterial Foraging (BF)*, descrito em Gazi e Passino (2004). Foram realizados experimentos em funções de *benchmark* de otimização global e também em um problema real. A variação proposta conseguiu melhores resultados em relação à versão original do algoritmo genético.

Ainda no âmbito de algoritmos genéticos, em Silva et al. (2013a) foi desenvolvida uma hibridização do BRKGA descrito na Seção 2.1.2 com um procedimento de busca local para resolver problemas de otimização global em espaços contínuos. Nesta abordagem, a vizinhança de uma solução é determinada pela projeção de pontos do espaço de busca, discretizado em uma malha, em uma hiperesfera com centro na solução. Este procedimento de busca local foi aplicado também ao *Continuous Greedy Randomized Adaptive Search Procedure (C-GRASP)*, uma metaheurística proposta por Hirsch et al. (2007) e

posteriormente melhorado em Hirsch, Pardalos e Resende (2010), que consiste numa adaptação do GRASP para lidar com problemas de otimização contínuos. O GRASP é um método metaheurístico para problemas de otimização combinatória de variáveis inteiras, que combina um método construtor para criar as soluções iniciais e uma busca local para tentar melhorar a qualidade das soluções. O C-GRASP utiliza a mesma abordagem, porém para espaços de busca contínuos.

Outro algoritmo evolucionário para problemas de otimização global contínua, o algoritmo da Evolução Diferencial, do inglês *Differential Evolution (DE)*, foi proposto em Storn e Price (1997) com a iniciativa de reduzir a quantidade dos parâmetros de controle em relação a outros algoritmos evolucionários (como o AG), além de apresentar uma maior simplicidade para codificá-lo. Este algoritmo consiste em gerar novos indivíduos por meio da adição da diferença ponderada entre dois indivíduos a um terceiro indivíduo, gerando uma mutação. O DE apresentou bons resultados apesar da sua ideia simples, porém como é demonstrado em Sun, Zhang e Tsang (2005), a técnica apresenta uma tendência de maior convergência local. Como uma maneira de inserir um mecanismo para extrair e usar informações globais sobre o espaço de pesquisa, Sun, Zhang e Tsang (2005) combinou o o DE com outro algoritmo evolucionário, o Algoritmo de Estimação da Evolução, do inglês *Estimation of Distribution Algorithm (EDA)* (LARRAÑAGA; LOZANO, 2001), que utiliza técnicas para estimar a distribuição de probabilidade da população inteira em vez de operadores genéticos e pode ser aplicado também a problemas de otimização global. A combinação dos dois algoritmos permitiu um melhor balanceamento entre as características de convergência local do DE e da busca global do EDA.

Como é possível notar, diversos esforços são demandados na literatura para a busca de boas soluções para problemas de otimização global. Por se tratar de uma área bastante presente em situações reais, que geralmente dão origem a modelagens mais complexas, este tipo de problema dificilmente é passível de ser resolvido algoritmos exatos em um tempo tratável, o que favorece a aplicação de métodos aproximativos e heurísticos. Estes últimos, embora não sejam concebidos com o propósito de retornar sempre a solução exata, objetivam-se a encontrar boas soluções em um tempo tratável. Por esse motivo, o estudo de melhorias na convergência dos algoritmos heurísticos existentes e a proposta de novas soluções são constantes na comunidade científica, o que acabou despertando nosso interesse em contribuir com esta pesquisa.

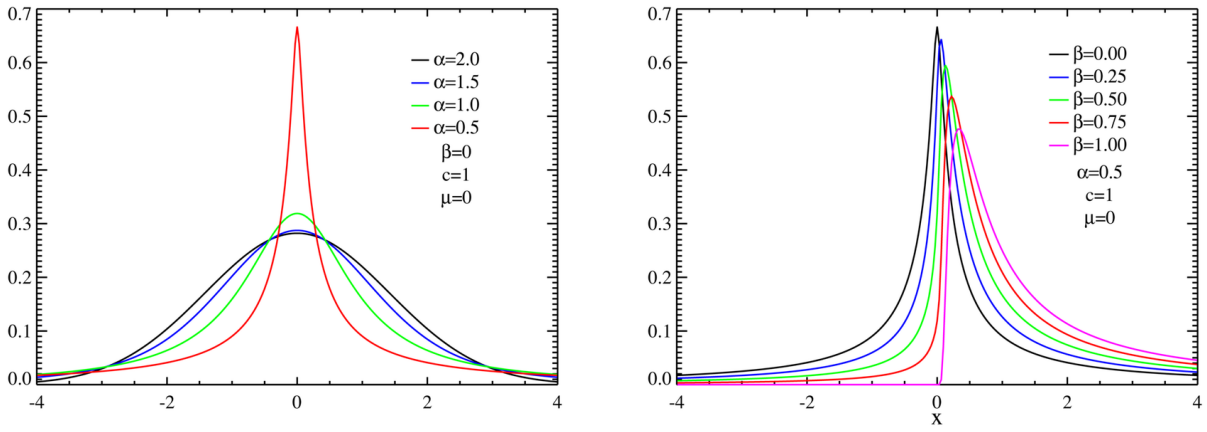
3 BRKGA COM DISTRIBUIÇÃO DE LEVY

3.1 DISTRIBUIÇÃO DE LEVY

A distribuição de Levy, também conhecida como distribuição Levy-estável, é uma família de distribuições concebida por Paul Lévy nos anos de 1920 e 1930 (PAUL, 1937), que permite diversas possibilidades de simetrias e curtoses, além de terem importantes propriedades matemáticas (GUERGELOT; ARTUSO, 2014) que favorecem sua aplicabilidade em problemas reais, principalmente na área de finanças.

Esta distribuição é definida por quatro parâmetros:

- O parâmetro α, chamado de índice da lei, índice de estabilidade ou ainda expoente característico, varia entre 0 e 2. Diferentes valores deste parâmetro podem representar diferentes distribuições, como pode ser verificado na Figura 11(a). Por exemplo: quando $\alpha = 2$, a distribuição transforma-se em uma Gaussiana e quando $\alpha = 1$, corresponde à distribuição de Cauchy.
- O parâmetro β, chamado de assimetria da lei, varia entre -1 e 1. Quando $\beta = 0$, a distribuição é simétrica, se $\beta > 0$ é assimétrica para direita, e se $\beta < 0$ é assimétrica para esquerda. Esta modificação pode ser verificada graficamente na Figura 11(b).
- O parâmetro γ ou c, conhecido como parâmetro de escala, pode ser qualquer número positivo e determina o espalhamento da distribuição.
- O parâmetro δ ou ou μ chamado de parâmetro de localização, translada a distribuição para direita se $\delta > 0$, e para esquerda se $\delta < 0$.



(a) Variação do parâmetro α na distribuição de Levy

(b) Variação do parâmetro β na distribuição de Levy

Figura 11 – Variação dos parâmetros α e β na distribuição de Levy

A distribuição de Levy está presente em outros trabalhos da área de otimização, como (LEE; YAO, 2004), (YANG; DEB, 2009), (YANG; DEB, 2010), (GANDOMI; YANG; ALAVI, 2013), (YANG; DEB, 2014), entre outros, os quais propõem métodos de busca bioinspirados, simulando comportamentos que seguem propriedades desta distribuição e apresentam bons resultados. A interferência da distribuição de Levy na melhoria dos resultados dos algoritmos é justificada principalmente pelas características de caudas pesadas, média e variância infinitas existentes em algumas de suas variações, que permitem uma melhor exploração do espaço de busca, evitando que os algoritmos tenham convergências prematuras em ótimos locais. Esta observação, alinhada ao estudo da distribuição, serviram de motivação para a investigação do impacto da sua aplicação na geração das chaves aleatórias do BRKGA para problemas de otimização global com restrições de caixa, principalmente pelos resultados promissores observados no método *Cuckoo Search* para a mesma categoria de problemas, o qual tem a distribuição de Levy como suporte.

3.2 INCLUSÃO DA DISTRIBUIÇÃO DE LEVY NO BRKGA

A proposta deste trabalho consiste em utilizar configurações da distribuição de Levy na geração de chaves aleatórias do Algoritmo Genético de Chaves Aleatórias Viciadas ou BRKGA. Como já citado anteriormente, esta alteração foi motivada devido aos bons resultados obtidos por algoritmos da literatura baseados em comportamentos bioinspirados que seguem a dinâmica de algumas variações da distribuição de Levy.

A alteração foi investigada em duas versões do BRKGA. A primeira delas foi o BRKGA clássico (GONÇALVES; RESENDE, 2011) e a segunda foi baseada na versão do BRKGA com hibridização por meio da adição de um procedimento de busca local (SILVA et al., 2013a). A modificação interferiu na geração da população inicial e na operação de mutação dos algoritmos, em que novos valores são gerados aleatoriamente para os alelos dos indivíduos.

Uma outra modificação importante foi realizada no espaço das chaves aleatórias para as variantes que utilizam a distribuição de Levy. No BRKGA tradicional, as chaves são geradas no intervalo [0, 1) enquanto nas versões com Levy, o intervalo é definido como parâmetro de entrada do algoritmo, que pode ser o mesmo intervalo das versões tradicionais ou não. Para o problema de otimização global com restrições de caixa abordado neste trabalho, o intervalo das chaves aleatórias foi determinado pelos limites superiores e inferiores das variáveis de decisão. Dessa forma, os números de Levy foram utilizados diretamente como variáveis para as funções a serem otimizadas, diminuindo um passo da decodificação.

Os algoritmos foram desenvolvidos em Python, utilizando como base o *framework* desenvolvido por Toso e Resende (2015), que propõe uma Application Programming Interface (API) para o BRKGA utilizando a linguagem de programação C++. Foi utilizado o módulo gerador de variáveis de Levy "*levy_stable.rvs*", disponível na biblioteca *Scipy* do Python, sendo necessário desenvolver mecanismos para que os resultados fossem gerados

dentro dos limites superiores e inferiores das variáveis de decisão. Para tanto, foi utilizada a operação de módulo em todas as situações necessárias.

Como a distribuição de Levy necessita dos quatro parâmetros descritos na Seção 3.1 para ser definida, as versões do BRKGA (GONÇALVES; RESENDE, 2011) e do BRKGA com busca local (SILVA et al., 2013a) que a utilizam precisaram ser acrescidas destes parâmetros. A descrição dos parâmetros pode ser verificada na Seção 3.1. Por este motivo, a utilização do ajuste automático de parâmetros por meio do *Iterated Racing* foi essencial.

Um pseudocódigo da implementação do procedimento gerador de números aleatórios segundo a distribuição de Levy pode ser verificado no Algoritmo 5.

Algoritmo 5 Pseudocódigo do Método Gerador de Números de Levy

```
1: método randLevy(i, l, u, α, β, γ, δ)
2:     random_number ← levy_stable.rvs(α, β, γ, δ);
3:     se random_number > u_i or random_number < l_i então
4:         se l_i = 0 então
5:             retorna |random_number| mod u_i;
6:         senão se l_i < 0 então
7:             se u_i = 0 então:
8:                 retorna −|random_number mod l_i|;
9:             senão se u_i < 0 então
10:                retorna u_i + (|random_number| mod (l_i − u_i));
11:            senão
12:                se random_number > 0 então
13:                    retorna random_number mod u_i;
14:                senão se random_number < 0 então
15:                    retorna random_number mod l_i;
16:                fim se
17:            fim se
18:        senão
19:            retorna l_i + (|random_number| mod (u_i − l_i));
20:        fim se
21:    fim se
22:    retorna random_number;
23: fim método
```

O método é iniciado na linha 1, recebendo como parâmetros: a posição i da variável aleatória no vetor; os vetores $\mathbf{l}$ e $\mathbf{u}$ dos limites inferiores e superiores, respectivamente; e os parâmetros α, β, γ e δ da distribuição de Levy. Na linha 2 é gerado um número aleatório seguindo a distribuição de Levy a partir dos valores dos parâmetros α, β, γ e δ. Em seguida, na linha 3 é verificado se o número aleatório gerado está dentro dos limites inferior e superior da variável de decisão do problema de otimização global com restrições de caixa. Caso não esteja, da linha 3 até a linha 21 são realizadas as verificações e ajustes do número para que ele se torne viável, consistindo em três situações:

- **Caso o limite inferior da variável seja igual a 0:** Na linha 4 é verificado se o

limite inferior da variável l_i é 0. Em caso afirmativo, na linha 5 o método retorna o valor de $|random\_number|$ **mod** l_i, ou seja, o resto da divisão do valor absoluto do número aleatório seguindo a distribuição de Levy pelo limite inferior da variável x_i.

- **Caso o limite inferior seja negativo:** Na linha 6 é verificado se o limite inferior l_i é negativo. Caso ele seja, o limite superior u_i deverá ser analisado, podendo atender a uma das três situações a seguir:
  - Nas linhas 7 e 8, se $u_i = 0$, será retornado o valor de $-|random\_number$ **mod** $l_i|$, ou seja, o resto do valor absoluto da divisão do número aleatório de Levy pelo limite inferior da variável x_i, multiplicado por -1;
  - Nas linhas 9 e 10, se o limite superior u_i é negativo, será retornado o valor de $u_i + (|random\_number|$ **mod** $(l_i - u_i))$, ou seja, o limite superior da variável x_i somado ao da divisão do valor absoluto do número aleatório de Levy pela subtração do limite inferior e do limite superior da variável;
  - E nas linhas 11 a 16, caso o limite superior seja positivo, se o número aleatório também é positivo (linha 12), o método retornará $random\_number$ **mod** $u_i|$, ou seja, o resto da divisão do número aleatório pelo limite superior u_i; se o número aleatório é negativo, retorna-se $random\_number$ **mod** $l_i|$, ou o resto da divisão do número aleatório pelo limite inferior l_i.
- **Caso o limite inferior seja positivo**: Nas linhas 18 e 19 o método retornará $l_i + (|random\_number|$ **mod** $(u_i - l_i))$, ou seja, o limite inferior da variável x_i somado ao resto da divisão do valor absoluto do número aleatório gerado via distribuição de Levy pela diferença entre os limites superior e inferior da variável de decisão.

O procedimento termina retornando o valor aleatório viável.

Como citado anteriormente, a inclusão da distribuição de Levy foi aplicada à duas versões do BRKGA disponíveis na literatura. A primeira delas foi o BRKGA clássico, proposto por Gonçalves e Resende (2011) e detalhado na Subseção 2.1.2. A segunda versão foi baseada em uma variante do BRKGA, proposta por Silva et al. (2013a), que adicionou ao algoritmo original um procedimento de busca local, baseado no trabalho proposto por Hirsch, Pardalos e Resende (2010) para a metaheurística C-GRASP. Esta busca local está detalhada a seguir.

BUSCA LOCAL

Em linhas gerais, para uma dada solução de entrada $\mathbf{x} \in \mathbb{R}^n$, o procedimento de busca local gera uma vizinhança (conjunto de soluções próximas a $\mathbf{x}$) na qual irá procurar por soluções cuja função objetivo é melhor do que a aplicada sobre a solução corrente $\mathbf{x}$. Se

uma solução melhor é encontrada, ela substitui a solução atual e o processo é repetido a partir da nova solução.

A busca local utilizada neste trabalho, utiliza a abordagem descrita em Hirsch, Pardalos e Resende (2010) e Silva et al. (2013a) a seguir: Seja $\bar{\mathbf{x}} \in \mathbb{R}^n$ a solução atual e h o parâmetro utilizado para discretizar o domínio contínuo S em uma malha. Define-se

$$S_h(\bar{\mathbf{x}}) = \{\mathbf{x} \in S \mid \mathbf{l} \leq \mathbf{x} \leq \mathbf{u},\ \mathbf{x} = \bar{\mathbf{x}} + \tau \cdot h,\ \tau \in \mathbb{Z}^n\} \tag{3.1}$$

como o conjunto de pontos em S que são passos inteiros (de tamanho h) partindo de $\bar{\mathbf{x}}$ (A Figura 12 (a) apresenta um exemplo desta estratégia de discretização em um espaço bidimensional). E seja

$$B_h(\bar{\mathbf{x}}) = \{\mathbf{x} \in S \mid \mathbf{x} = \bar{\mathbf{x}} + h \cdot (\mathbf{x}' - \bar{\mathbf{x}})/\|\mathbf{x}' - \bar{\mathbf{x}}\|,\ \mathbf{x}' \in S_h(\bar{\mathbf{x}}) \setminus \{\bar{\mathbf{x}}\}\} \tag{3.2}$$

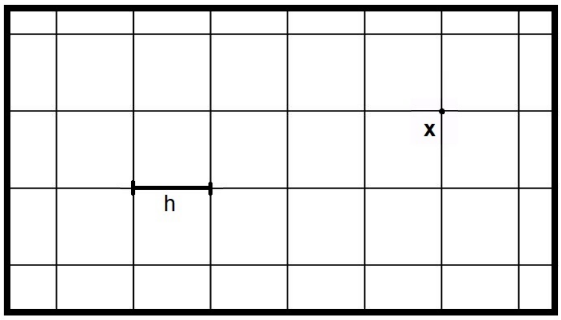
a projeção dos pontos de $S_h(\bar{\mathbf{x}}) \setminus \{\bar{\mathbf{x}}\}$ na hiperesfera com centro em $\bar{\mathbf{x}}$ e raio h (Figura 12 (b)). A h-vizinhança da solução $\bar{\mathbf{x}}$ é definida como o conjunto de pontos em $B_h(\bar{\mathbf{x}})$. As soluções do conjunto $B_h(\bar{\mathbf{x}})$ são avaliadas a fim de procurar uma solução cuja função objetivo seja melhor que a de $\bar{\mathbf{x}}$ (Figura 12 (c)). Caso seja encontrada uma solução melhor, $\bar{\mathbf{x}}$ será substituído por esta solução e o procedimento é repetido (Figura 12 (d)) até que seja encontrado um mínimo local ou um critério de parada seja atingido.

Para um melhor entendimento, o pseudocódigo detalhado da busca local é apresentada no Algoritmo 6.

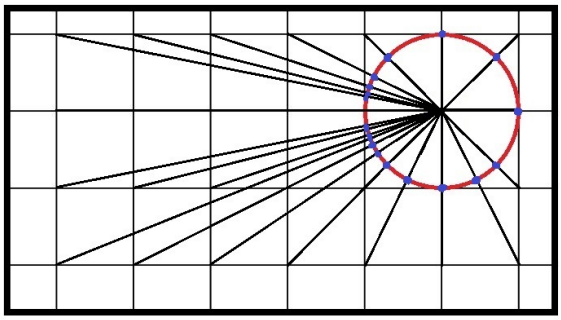
Na linha 1 do Algoritmo 6, a busca local recebe como parâmetros de entrada: uma solução corrente $\mathbf{x}$; o valor da função objetivo $f(\mathbf{x})$; o valor inicial do parâmetro de discretização hs; o valor limite de discretização he; o vetor de limites inferiores $\mathbf{l}$; o vetor de limites superiores $\mathbf{u}$; e a quantidade máxima de pontos vizinhos a serem analisados $MaxPointsToExamine$. O procedimento é iniciado a partir de uma solução $\mathbf{x} \in S \subseteq \mathbb{R}^n$. Nas linhas 2 a 5, a melhor solução atual da busca local $\mathbf{x}^*$ é inicializada a partir de $\mathbf{x}$, assim como o melhor valor da função objetivo $f(\mathbf{x}^*)$ é inicializada a partir de $f(\mathbf{x})$. Além disso, o parâmetro h é inicializado pelo parâmetro de discretização inicial hs, e a variável de que indica se houve melhoria $Impr$ é inicializada como $false$. Baseado no valor atual do parâmetro h e no número de pontos em $B_h(\mathbf{x}^*)$, o número total de pontos da malha $MaxGridPoints$ é calculado na linha 6 do pseudocódigo.

Diferente do que é descrito em Hirsch, Pardalos e Resende (2010), nas linhas 7-9 foi imposta uma restrição que limita o máximo número de pontos a serem avaliados na vizinhança ao valor do parâmetro $MaxPointsToExamine$, como ocorre em Silva et al. (2013a).

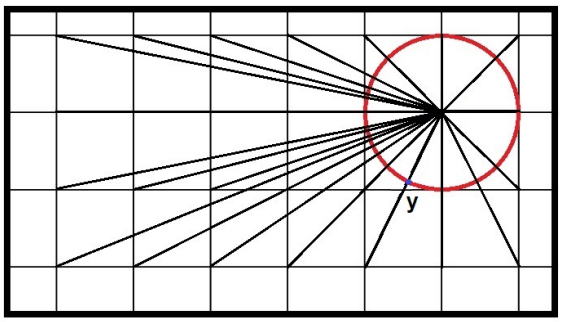
A partir do ponto $\mathbf{x}^*$, no *loop* entre as linhas 12 e 21 o algoritmo seleciona aleatoriamente $PointsToExamine$ pontos em $B_h(\mathbf{x}^*)$, um de cada vez (linha 14). Na linha 15, se a solução atual $\mathbf{x}$ selecionada de $B_h(\mathbf{x}^*)$ é viável e é melhor que $\mathbf{x}^*$, então $\mathbf{x}^*$ recebe o valor de $\mathbf{x}$ na linha 16, $f(\mathbf{x}^*)$ recebe o valor de $f(\mathbf{x})$ na linha 17, $Impr$ é alterada para $true$ na linha 18,



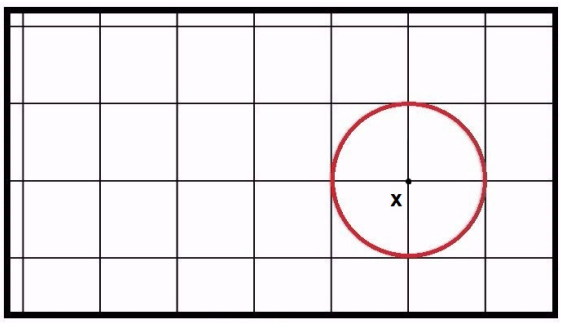
(a) O espaço contínuo de busca é discretizado utilizando passos de tamanho h em relação a uma solução corrente $\mathbf{x}$.



(b) A vizinhança da solução $\mathbf{x}$ é definida como a projeção, em relação a $\mathbf{x}$, dos pontos do espaço discretizado numa hiperesfera com centro em $\mathbf{x}$ e raio h. Neste exemplo, como trata-se de uma representação bidimensional, a vizinhança de $\mathbf{x}$ se dá pela projeção dos pontos em uma circunferência.



(c) Uma determinada quantidade de pontos vizinhos é avaliada a fim de que seja encontrada uma solução cujo resultado da função objetivo seja melhor que a de $\mathbf{x}$. Neste exemplo, suponhamos que seja encontrada uma solução $\mathbf{y}$ cuja função objetivo apresente um resultado melhor.



(d) Esta solução passa a ser o novo $\mathbf{x}$ corrente e o procedimento é repetido a partir dela.

Figura 12 – Exemplo do procedimento de busca local em um espaço bidimensional.

$NumPointsExamined$ é reinicializada como *zero* na linha 19 e o processo reinicia com $\mathbf{x}^*$ como a nova solução inicial. A variável $Impr$ é usada para determinar se a busca local conseguiu melhorar a solução ou não. Na linha 22, se esta variável estiver configurada como *true*, a solução melhorada $\mathbf{x}^*$ é retornada pelo procedimento nas linhas 23. Senão, na linha 25, a densidade da malha h é aumentada, sendo atualizada para a metade de h e o primeiro *loop* é reinicializado, se repetindo até que a densidade aumente até o limite he.

A busca local termina quando uma solução *h-mínima-local* é encontrada, sendo esta solução retornada pelo procedimento nas linhas 23 ou 28.

Cabe observar que em relação ao BRKGA original, o procedimento de busca local foi aplicado à prole gerada após o processo evolucionário, antes de ser iniciada uma nova geração.

A versão original do BRKGA foi utilizada como base para o desenvolvimento da sua

Algoritmo 6 Pseudocódigo da Busca Local

método LocalSearch($\mathbf{x}, f(\mathbf{x}), hs, he, \mathbf{l}, \mathbf{u}$, `MaxPointsToExamine`)
  $\mathbf{x}^* \leftarrow \mathbf{x}$;
  $f(\mathbf{x}^*) \leftarrow f(\mathbf{x})$;
  $h \leftarrow hs$
  $Impr \leftarrow false$
  $MaxGridPoints \leftarrow \prod_{i=1}^{n} \lceil (u_i - l_i)/h \rceil$;
  se $MaxPointsToExamine > MaxGridPoints$ **então**
    $MaxPointsToExamine \leftarrow MaxGridPoints$;
  fim se
  enquanto $h \leq he$ **faça**
    $NumPointsExamined \leftarrow 0$;
    enquanto $NumPointsExamined \leq MaxPointsToExamine$ **faça**
      $NumPointsExamined \leftarrow NumPointsExamined + 1$;
      $\mathbf{x} \leftarrow RandomlySelectElement(B_h(\mathbf{x}^*), \mathbf{l}, \mathbf{u}, h)$;
      se $\mathbf{l} \leq \mathbf{x} \leq \mathbf{u}$ **and** $f(\mathbf{x}) < f(\mathbf{x}^*)$ **então**
        $\mathbf{x}^* \leftarrow \mathbf{x}$;
        $f(\mathbf{x}^*) \leftarrow f(\mathbf{x})$;
        $Impr \leftarrow true$;
        $NumPointsExamined \leftarrow 0$;
      fim se
    fim enquanto
    se $Impr = true$ **então**
      retorna $\mathbf{x}^*$
    senão
      $h \leftarrow h/2$
    fim se
  fim enquanto
  retorna $\mathbf{x}^*$
fim método

variante com busca local e em seguida foi realizada a inclusão da distribuição de Levy na geração das chaves aleatórias das duas abordagens resultantes.

Para um melhor entendimento das variações utilizadas e desenvolvidas neste trabalho, o diagrama do *framework* do BRKGA com as modificações em destaque pode ser observado na Figura 13. Como descrito na legenda, os procedimentos em cinza permaneceram iguais ao BRKGA original. Para a variante com busca local foi acrescentado um novo procedimento no algoritmo logo após as operações do processo evolucionário. Por fim, os retângulos verdes representam os métodos que sofreram interferência após a inclusão da distribuição de Levy. Estes métodos são a inicialização da população, em que novos indivíduos são gerados aleatoriamente e o método de mutação, em que alguns alelos são alterados por valores aleatórios. Nas versões tradicionais do BRKGA, estes procedimentos de geração de números aleatórios são realizados por meio de distribuição uniforme.

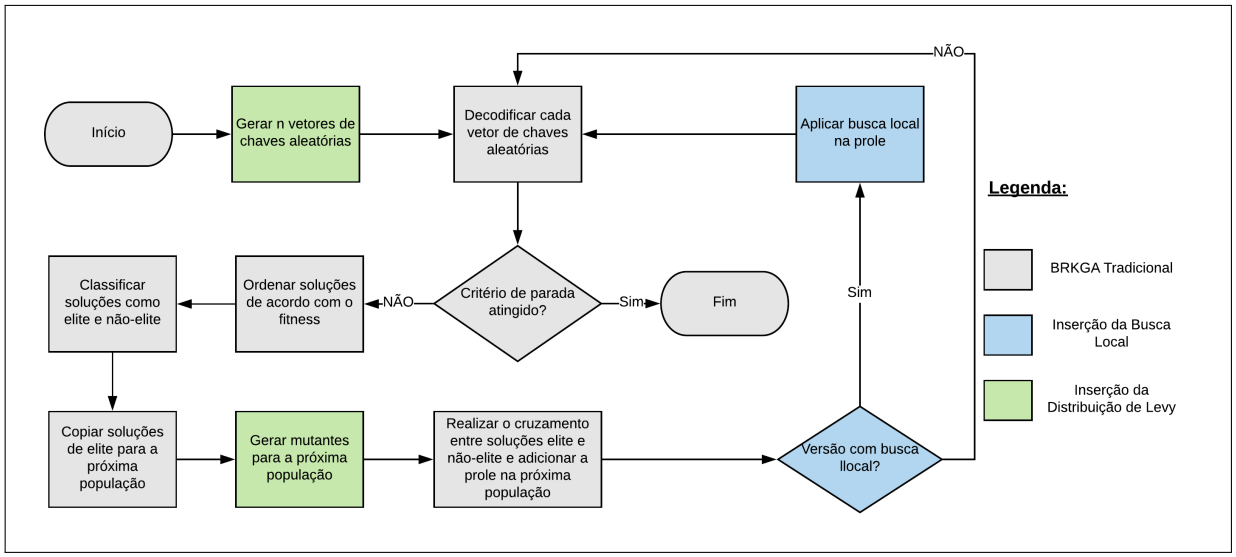


Figura 13 – Fluxograma do BRKGA com a inserção da busca local e da distribuição de Levy.

3.3 DECODIFICAÇÃO DAS CHAVES ALEATÓRIAS

O processo de decodificação das chaves aleatórias foi realizado com base no domínio das funções de *benchmark* utilizadas, mais especificamente a partir das restrições de caixa do problema, e o *fitness* foi calculado utilizando o resultado dos vetores decodificados aplicados às funções. Para as versões que utilizam distribuição de Levy, as chaves já foram geradas utilizando como intervalos os limites inferiores e superiores das variáveis de decisão do problema, enquanto nas versões sem a distribuição de Levy, as chaves aleatórias são geradas no intervalo entre [0,1). Devido a esta diferença, tornou-se necessário criar dois tipos de decodificadores diferentes:

- **Decodificador para as variantes já existentes do BRKGA:**

  Nas versões do BRKGA que não utilizam a distribuição de Levy, o decodificador primeiro precisa transformar o vetor de chaves aleatórias definidas no intervalo [0,1) em um vetor que represente a solução. Para isso utilizamos o seguinte método:

  Dado um cromossomo $\mathbf{c}$ de n chaves aleatórias definidas no intervalo [0, 1) e os vetores $\mathbf{l}$ e $\mathbf{u}$ que contêm os limites inferiores e superiores das n variáveis de decisão de uma função, cada variável x_i do vetor solução $\mathbf{x}$, será definida como:

$$x_i = l_i + c_i(u_i - l_i), \forall i = 1, ..., n \tag{3.3}$$

  Em seguida, o vetor $\mathbf{x}$ será substituído na função objetivo e o decodificador retornará o valor do $fitness$ do cromossomo. Este procedimento pode ser melhor compreendido no exemplo demonstrado na Figura 14. Neste exemplo, utilizamos uma instância da

função de otimização global *Sphere*, com 6 dimensões. As variáveis de decisão desta função estão compreendidas no domínio $[2.56, 5.12]^6$, ou seja, os limites inferiores e superiores das suas variáveis são respectivamente 2.56 e 5.12. O decodificador recebe como entrada um vetor de chaves aleatórias, decodifica-o em uma solução viável para o problema utilizando a Equação 3.3 e em seguida, o vetor decodificado é submetido à função de avaliação que retornará seu *fitness*.

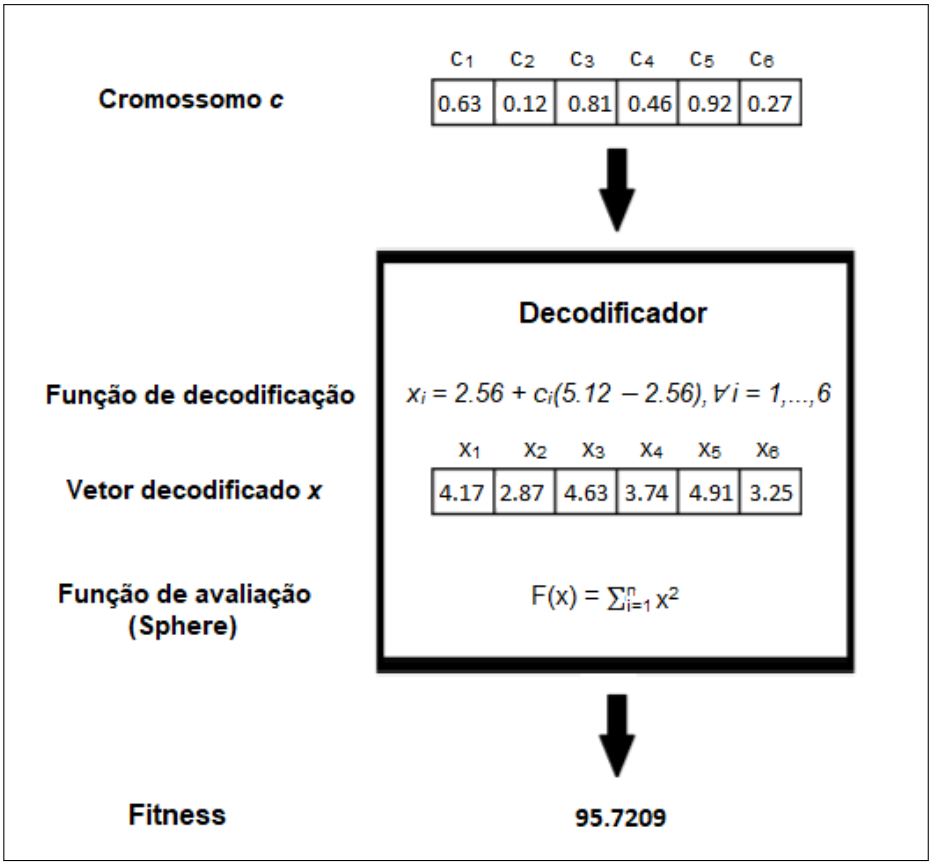


Figura 14 – Exemplo do processo de decodificação de uma chave aleatória do BRKGA tradicional para a função de otimização global Sphere.

- **Decodificador para as variantes do BRKGA que utilizam distribuição de Levy:**

  No caso do decodificador para as versões que utilizam distribuição de Levy, como as chaves aleatórias já foram geradas no domínio do problema, não foi necessário realizar nenhuma alteração no vetor retornado pelo algoritmo. Este vetor pode ser substituído diretamente na função a ser avaliada e o resultado é retornado como $fitness$. Na Figura 15 está ilustrado o mesmo exemplo utilizado para demonstrar a decodificação na versão tradicional. No caso, o vetor decodificado x é uma cópia do cromossomo c. Em seguida o vetor x é submetido à função de avaliação, que retorna seu *fitness*.

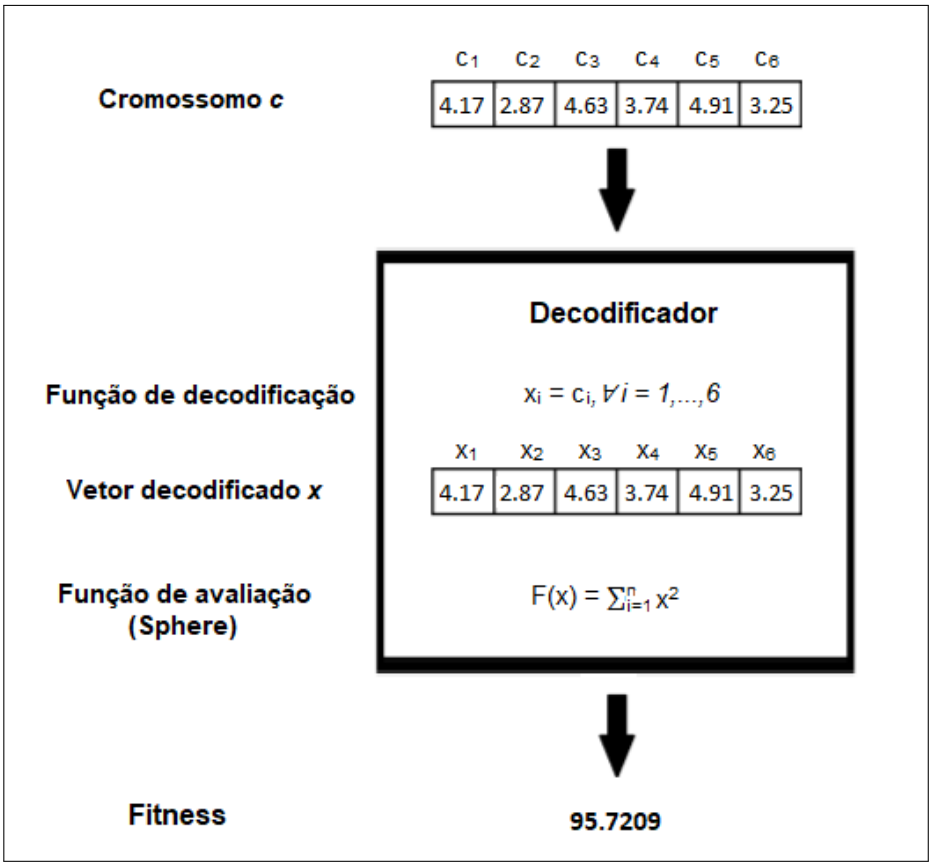


Figura 15 – Exemplo do processo de decodificação de uma chave aleatória do BRKGA com distribuição de Levy para a função de otimização global Sphere.

4 ANÁLISE EXPERIMENTAL

A análise experimental foi realizada considerando-se as quatro versões do BRKGA implementadas, que serão denominadas neste trabalho como:

- **BRKGA**: versão canônica do BRKGA, sem busca local e sem distribuição de Levy;
- **BRKGA-Levy**: versão do BRKGA sem busca local mas utilizando a distribuição de Levy;
- **BRKGA-LS**: versão do BRKGA com busca local e sem distribuição de Levy;
- **BRKGA-Levy-LS**: versão do BRKGA com busca local e com distribuição de Levy.

Para a execução dos algoritmos nos experimentos, seguiu-se o fluxograma demonstrado na Figura 16. Primeiro, as versões do BRKGA foram submetidas ao ajuste automático de parâmetros da distribuição de Levy e depois ao ajuste automático dos parâmetros dos algoritmos em si. Em seguida, os experimentos foram realizados utilizando funções de *benchmark* de otimização global da literatura. Por fim, os resultados foram organizados em tabelas e gráficos, analisados e comparados a fim de avaliar as contribuições propostas.

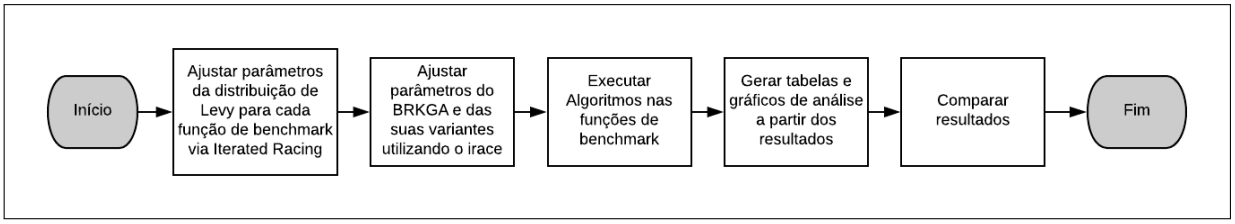


Figura 16 – Fluxograma da Execução dos Experimentos

As tabelas detalham os resultados em termos de qualidade das soluções e tempo de execução. Nas tabelas foram registrados os valores de mínimo, máximo, média e desvio-padrão alcançados pelos algoritmos em cada uma das funções, e na tabela de resultados foi registrado também o percentual de execuções que conseguiram atingir o critério de parada de qualidade descrito na seção 4.2.3.

A convergência dos algoritmos foi analisada por meio de gráficos de *Time to Target Plot (TTT-Plot)*. A técnica de *TTT-Plot* (RIBEIRO; ROSSETI, 2015) utiliza os tempos computacionais oriundos da aplicação dos algoritmos nas instâncias de teste para gerar e exibir a probabilidade de ser encontrado o alvo (*target*) previamente estipulado e geralmente definido pelo valor ótimo (ou melhor solução até então encontrada na literatura) ou próximo do ótimo em um determinado tempo de execução. Esta estimativa é fundamentada na observação do comportamento de muitas heurísticas baseadas em procedimentos de busca

local, tais como, Algoritmo Genético, *Simulated Annealing*, GRASP e Busca Tabu (Aiex, Resende e Ribeiro (2002), Battiti e Tecchiolli (1992), Dodd (1990), Eikelder et al. (1996), Osborne e Gillett (1991), Selman, Kautz e Cohen (1994)), em que o tempo variável aleatório de execução para encontrar o valor de uma solução é distribuído exponencialmente, ou se ajusta a uma distribuição exponencial de dois parâmetros deslocada, ou seja, a probabilidade de não ser encontrada uma solução alvo em t unidades de tempo é dada por $P(t) = e^{(-t-\mu)/\lambda}$, com $\lambda \in \mathrm{R}^+$ e $\mu \in \mathrm{R}$, onde λ é a média (e também o desvio padrão) dos dados da distribuição e μ é o deslocamento da distribuição em relação ao eixo das ordenadas. Esta hipótese é estudada e comprovada por (HOOS; STÜTZLE, 1998). No eixo das abscissas são representados os tempos computacionais, e no eixo das ordenadas, a probabilidade cumulativa. Os melhores resultados serão aqueles que alcançam a probabilidade 1.0 mais rapidamente, ou seja, aqueles representados pelas linhas que estiverem concentradas mais à esquerda e no topo do gráfico.

Os experimentos foram divididos em três fases de análise de resultados:

- A primeira fase consiste na comparação quanto à qualidade dos resultados das duas versões do BRKGA existentes na literatura (BRKGA e BRKGA-LS) e de duas versões do Algoritmo Genético clássico disponibilizadas em Andre, Siarry e Dognon (2001), a fim de justificar a escolha do BRKGA ao invés de uma abordagem canônica do AG;

- A segunda fase corresponde à análise principal à qual este trabalho se propõe: a comparação entre as abordagens do BRKGA com a distribuição de Levy e suas versões originais com o intuito de verificar a veracidade da hipótese de melhoria do BRKGA via distribuição de Levy para problemas de otimização global com restrições de caixa;

- Na terceira fase, a melhor dentre as abordagens do BRKGA ainda foi comparada com uma implementação do algoritmo C-GRASP (HIRSCH et al., 2007), (HIRSCH; PARDALOS; RESENDE, 2010). O C-GRASP é uma metaheurística também aplicada a problemas de otimização global contínua com restrições de caixa e utiliza um procedimento de busca local semelhante ao que foi inserido no BRKGA em Silva et al. (2013a). Além disso, este algoritmo apresentou contribuições significativas para o estado da arte da otimização global, tornando a comparação com seus resultados um bom indicador a respeito da relevância das contribuições do presente trabalho para a literatura.

Na segunda e terceira fase foram realizados testes estatísticos sobre os resultados e tempos computacionais obtidos pelos algoritmos para avaliar a significância das modificações propostas. Nos casos em que foram comparadas mais de duas abordagens, foi aplicado o teste não-paramétrico de *Friedman* (FRIEDMAN, 1937), que utiliza os *rankings*

dos dados ao invés de seus valores brutos para confirmar se existe diferença estatística entre pelo menos um par de amostras. Caso confirmada a diferença, os resultados do teste de *Friedman* são submetidos ao pós teste *Nemenyi* (NEMENYI, 1963), que faz a comparação par a par entre as amostras a fim de identificar em quais pares de abordagens encontram-se esta diferença estatística. Nos casos em que apenas duas amostras foram comparadas, foi utilizado o teste de *Wilcoxon Signed-Rank* pareado (WILCOXON, 1945), que compara se as medidas de posição de duas amostras dependentes são iguais ou se há diferença estatística entre elas.

4.1 AMBIENTE COMPUTACIONAL

A linguagem de programação utilizada para o desenvolvimento do trabalho foi Python, versão 2.7, sendo necessária a inclusão das bibliotecas *Futures*, *SciPy* e *NumPy*, que não são nativas da linguagem. Os algoritmos foram desenvolvidos com base na API para o BRKGA proposta em Toso e Resende (2015).

Para o ajuste dos parâmetros, foi utilizada a biblioteca *irace* (LóPEZ-IBáñEZ et al., 2016), da suíte de software R (R Core Team, 2015).

Os experimentos foram executados em um servidor Ubuntu 14.04.5 LTS com processador Intel Xeon E5-2603 v3, de 1.60GHz e 12 núcleos, HD de 2 TB e memória RAM de 32 GB.

Os gráficos para a análise da convergência foram gerados utilizando a biblioteca *tttplot* (RIBEIRO; Rosseti., 2016), também disponível no R (R Core Team, 2015).

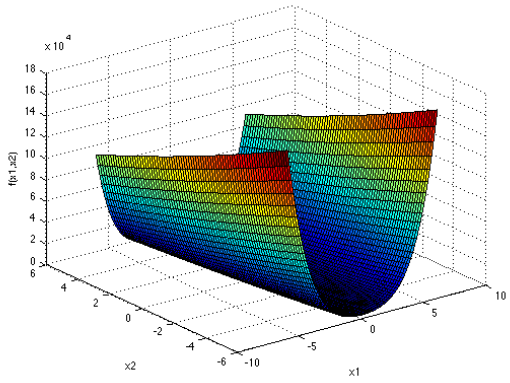
4.2 PROJETO EXPERIMENTAL

4.2.1 Funções de *Benchmark*

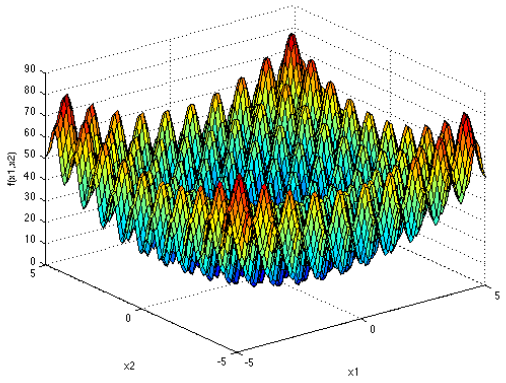
Para a validação do trabalho, foi utilizado um conjunto de 27 funções de *benchmark* de otimização global de variáveis contínuas, disponíveis em (JAMIL; YANG, 2013) e utilizadas em (HIRSCH; PARDALOS; RESENDE, 2010), além de vários outros trabalhos da literatura. Estas funções são clássicas para a avaliação de algoritmos, além de serem funções-base para as principais competições de otimização global, tais como a *IEEE CEC Competition*, que trata-se de uma competição entre algoritmos evolucionários realizada no *Congress on Evolutionary Computation (CEC)*.

As funções foram particionadas em grupos de acordo com suas características de **tamanho** e **modalidade**. O **tamanho** determina a quantidade de variáveis de decisão, que pode ser fixa ou não, dependendo das características da função. Esta possibilidade de variação de tamanho foi aqui chamada de **escalabilidade**. Por exemplo, a função *Beale* é classificada como não-escalável pois possui tamanho fixo, com apenas duas variáveis de decisão, enquanto a função *Ackley* pode variar o seu tamanho, sendo classificada como escalável. A modalidade classifica as funções em unimodais e multimodais. Uma função **multimodal** possui em sua curva múltiplos pontos de mínimo ou máximo enquanto a

função **unimodal** possui apenas um ponto mínimo ou máximo. Exemplos gráfico de função unimodal e multimodal podem ser vistos nas Figuras 17(a) e (b) respectivamente.



(a) Exemplo de função unimodal - Função Rosenbrock



(b) Exemplo de função multimodal - Função Rastrigin

Figura 17 – Funções exemplificando a característica de modalidade.

As características de escalabilidade e modalidade das funções selecionadas neste trabalho podem ser observadas na Tabela 2. Para informações mais detalhadas, tais como melhor vetor-solução e melhor valor da função objetivo encontrados pela literatura, definição da função e domínio, conferir o Apêndice A.

4.2.2 Ajustes dos Parâmetros

Os parâmetros dos algoritmos e da distribuição de Levy foram ajustados automaticamente utilizando o pacote *irace* do Comprehensive R Archive Network (CRAN), que implementa o método *Iterated Racing* descrito na Seção 2.2.1. Este procedimento de ajustes constituiu uma fase de pré-processamento para que o algoritmo pudesse ser executado nas funções.

Para executar o *irace*, primeiro foi necessário criar um conjunto de arquivos de configuração, os quais definem os parâmetros a serem ajustados, os tipos de cada parâmetro (variando entre real, inteiro ou categórico), os limites inferior e superior de cada parâmetro (ou os valores que eles podem assumir, caso sejam categóricos), uma configuração inicial (caso exista), as regras e restrições que eventualmente podem existir entre os parâmetros, o conjunto de instâncias que foi utilizado para o ajuste, uma função de custo que avalia cada combinação de parâmetros, o critério de parada do ajuste, entre outras informações. Os exemplos dos arquivos de configuração do *irace* estão devidamente explanados e disponíveis no Apêndice B.

Inicialmente, todas as funções foram submetidas ao *irace* para determinar a melhor combinação dos parâmetros da distribuição de Levy para cada uma. Os resultados obtidos neste procedimento podem ser verificados na Tabela 3.

Tabela 2 – Funções de Benchmark

Funções	**Escalabilidade**	**Modalidade**
Ackley	Escalável	Multimodal
Alpine 1	Escalável	Multimodal
Alpine 2	Escalável	Multimodal
Bartels Conn	Não-Escalável	Multimodal
Beale	Não-Escalável	Unimodal
Bohachevsky	Não-Escalável	Multimodal
Booth	Não-Escalável	Unimodal
Branin	Não-Escalável	Multimodal
Colville	Não-Escalável	Multimodal
Easom	Não-Escalável	Multimodal
Goldstein e Price	Não-Escalável	Multimodal
Griewank	Escalável	Multimodal
Levy	Escalável	Unimodal
Matyas	Não-Escalável	Unimodal
Perm	Escalável	Unimodal
Perm0	Escalável	Unimodal
Powell	Escalável	Unimodal
Power Sum	Não-Escalável	Unimodal
Rastrigin	Escalável	Multimodal
Rosenbrock	Escalável	Unimodal
Shekel	Escalável	Multimodal
Shubert	Não-Escalável	Multimodal
Six-Hump CamelBack	Não-Escalável	Multimodal
Sphere	Escalável	Multimodal
Sum of Squares	Escalável	Unimodal
Trid	Não-Escalável	Unimodal
Zakharov	Escalável	Multimodal

Tabela 3 – Ajuste automático dos parâmetros da distribuição de Levy para cada função de benchmark

Funções	**Parâmetros**				**Tempo de Execução** (m = minutos; s = segundos)
Ackley	$\alpha = 0.0489$	$\beta = 0.1406$	$\gamma = -15$	$\delta = 17.3453$	46m28.623s
Alpine 1	$\alpha = 0.0076$	$\beta = -0.1829$	$\gamma = -10$	$\delta = 19.3963$	40m46.147s
Alpine 2	$\alpha = 0.1349$	$\beta = -0.3791$	$\gamma = 8$	$\delta = 9.0285$	39m29.892s
Bartels Conn	$\alpha = 1.9727$	$\beta = -0.6742$	$\gamma = 441$	$\delta = 529.6006$	18m42.578s
Beale	$\alpha = 1.9014$	$\beta = 0.6417$	$\gamma = 2$	$\delta = 1.9089$	21m33.472s
Bohachevsky	$\alpha = 0.8778$	$\beta = 0.7827$	$\gamma = 37$	$\delta = 126.7566$	21m7.728s
Booth	$\alpha = 0.9409$	$\beta = 0.1378$	$\gamma = -3$	$\delta = 0.7772$	23m35.619s
Branin	$\alpha = 1.1597$	$\beta = -0.8297$	$\gamma = 12$	$\delta = 2.7782$	21m47.961s
Colville	$\alpha = 0.5548$	$\beta = 0.5572$	$\gamma = 0$	$\delta = 3.0912$	31m40.747s
Easom	$\alpha = 1.2359$	$\beta = -0.0039$	$\gamma = -5$	$\delta = 150.9143$	23m13.468s
Goldstein e Price	$\alpha = 0.488$	$\beta = -0.0396$	$\gamma = 1$	$\delta = 2.8133$	23m21.808s
Griewank	$\alpha = 0.2558$	$\beta = 0.3776$	$\gamma = 1$	$\delta = 253.7533$	40m3.186s
Levy	$\alpha = 1.2618$	$\beta = -0.6721$	$\gamma = 2$	$\delta = 17.4728$	52m58.782s
Matyas	$\alpha = 0.0194$	$\beta = 0.6192$	$\gamma = 6$	$\delta = 6.6508$	18m52.812s
Perm	$\alpha = 1.0338$	$\beta = -0.6417$	$\gamma = 6$	$\delta = 8.7062$	49m56.547s
Perm0	$\alpha = 0.3055$	$\beta = 0.5311$	$\gamma = 0$	$\delta = 15.618$	44m57.183s
Powell	$\alpha = 0.6871$	$\beta = 0.6942$	$\gamma = -2$	$\delta = 8.616$	36m14.844s
Power Sum	$\alpha = 1.449$	$\beta = -0.6781$	$\gamma = 1$	$\delta = 1.4174$	24m27.772s
Rastrigin	$\alpha = 0.5673$	$\beta = -0.3584$	$\gamma = 0$	$\delta = 0.0338$	41m52.012s
Rosenbrock	$\alpha = 1.656$	$\beta = -0.7724$	$\gamma = -5$	$\delta = 4.5895$	43m55.731s
Shekel	$\alpha = 0.9016$	$\beta = 0.8483$	$\gamma = 8$	$\delta = 0.8976$	38m3.272s
Shubert	$\alpha = 1.6105$	$\beta = 0.3809$	$\gamma = 4$	$\delta = 18.432$	25m7.262s
Six-Hump CamelBack	$\alpha = 1.5467$	$\beta = -0.0117$	$\gamma = 4$	$\delta = 7.1295$	25m50.046s
Sphere	$\alpha = 0.0186$	$\beta = 0.0742$	$\gamma = 0$	$\delta = 4.4863$	40m1.382s
Sum of Squares	$\alpha = 1.5477$	$\beta = 0.3432$	$\gamma = -5$	$\delta = 11.6944$	58m46.486s
Trid	$\alpha = 1.3613$	$\beta = -0.514$	$\gamma = 85$	$\delta = 172.2426$	20m39.657s
Zakharov	$\alpha = 1.6429$	$\beta = -0.1782$	$\gamma = 0$	$\delta = 0.4253$	45m4.353s

Foi possível perceber que os parâmetros da distribuição variaram muito de uma função para outra. A partir desta observação podemos dizer que os parâmetros da distribuição de Levy são sensíveis ao problema a ser resolvido, tornando-se um fator decisivo para que sejam obtidos resultados de qualidade e com bom desempenho pelos algoritmos.

Em seguida, cada um dos algoritmos teve seus parâmetros ajustados utilizando o mesmo método de ajuste. Os resultados obtidos no ajuste automático dos parâmetros de cada uma das versões do BRKGA via *Iterated Racing* podem ser visualizados na Tabela 4.

[1] O parâmetro *maxgen* corresponde à quantidade máxima de gerações.
[2] O parâmetro *maxpls* corresponde ao parâmetro *MaxPointsToExamine* da Busca Local.

Tabela 4 – Ajuste automático dos parâmetros das versões do BRKGA

BRKGA	**Parâmetros**	**Tempo de Execução** (m = minutos; s = segundos)
BRKGA	p=87 p_e=0.2619 p_m=0.2709 ρ_a=0.5617 maxgen[1]=425	561m26.141s
BRKGA-Levy	p=100 p_e=0.1748 p_m=0.5356 ρ_a=0.5759 maxgen[1]=483	1430m54.893s
BRKGA-LS	p=83 p_e=0.3146 p_m=0.5536 ρ_a=0.6425 maxgen[1]=406 hs=0.767 he=0.0215 maxpls[2]=31	3979m51.881s
BRKGA-Levy-LS	p=57 p_e=0.3449 p_m=0.119 ρ_a=0.2719 maxgen[1]=311 hs=0.8515 he=0.0125 maxpls[2]=6	5129m52.951s

4.2.3 Execução dos Experimentos

Cada função foi executada 30 vezes para cada algoritmo, com as configurações obtidas pela execução do *irace*.

As sementes utilizadas para as 30 execuções foram retiradas das casas decimais do número π e estão disponíveis na Tabela 5.

Tabela 5 – Sementes do gerador de números aleatórios utilizadas.

1415926535	8979323846	2643383279	5028841971	6939937510	5820974944
5923078164	8628034825	3421170679	8214808651	3282306647	5058223172
5359408128	4811174502	8410270193	8521105559	6446229489	5493038196
4428810975	6659334461	2847564823	3786783165	2712019091	4564856692
3460348610	4543266482	1339360726	7245870066	4881520920	9628292540

O *gap* de otimalidade entre os resultados e os valores ótimos conhecidos das funções foi definido pela função $GAP = |f(\mathbf{x}) - f(\mathbf{x}^*)|$, onde $\mathbf{x}$ é a melhor solução até então encontrada pela heurística e $\mathbf{x}^*$ é a solução mínima global conhecida.

Como critério de parada de qualidade, consideramos que uma heurística conseguiu atender um determinado problema quando

$$GAP \leq \begin{cases} \epsilon, & if \quad f(x^*) = 0 \\ \epsilon \cdot f(x^*), & if \quad f(x^*) \neq 0 \end{cases} \tag{4.1}$$

onde $\epsilon = 0.001$, tal como definido em Hirsch, Pardalos e Resende (2010).

Ao final foi calculado o valor médio dos resultados obtidos e dos tempos computacionais despendidos, assim como o valor mínimo, máximo, desvio padrão e o percentual de iterações que atingiram o critério de parada de qualidade.

Estes resultados encontram-se compilados, resumidos e comentados na Seção 4.3. As tabelas com os resultados detalhados das funções e dos tempos computacionais podem ser verificadas nos Apêndice C e D, respectivamente.

4.3 RESULTADOS

4.3.1 Comparação do BRKGA com o AG Clássico

A primeira fase de análise dos algoritmos foi constituída para a comparação entre: (a) as abordagens do BRKGA já existentes na literatura (versões BRKGA tradicional e BRKGA-LS); (b) os resultados disponíveis no trabalho de Andre, Siarry e Dognon (2001), referentes ao algoritmo genético original, o qual chamamos de AG; e (c) à versão abrangendo as melhorias propostas por Andre, Siarry e Dognon (2001), identificada por AGI. Esta comparação foi realizada apenas para confirmar se o BRKGA realmente incorporou melhorias ao AG clássico. Foram selecionadas 5 funções de *benchmark* do trabalho de Andre, Siarry e Dognon (2001) que também fazem parte do conjunto de funções utilizadas no presente trabalho, dentre as quais uma delas (*Shekel*) possui 3 versões, totalizando portanto 7 funções. Os resultados encontram-se descritos na Tabela 6. Como foram utilizados os resultados disponíveis no trabalho de Andre, Siarry e Dognon (2001), na tabela não estão inclusos os desvios-padrões nem os percentuais de iterações que atingiram o critério de parada de qualidade (pois não estão disponíveis na referência). A comparação foi realizada com relação aos valores das médias dos erros relativos, definidos como $E_r = |f(x) - f(x^*)|/|f(x^*)|$ e das médias dos erros absolutos, equivalentes à função *GAP*, definidos como $E_a = |f(x) - f(x^*)|$.

Como verificamos na Tabela 6, as duas versões do BRKGA alcançaram resultados melhores na maioria das funções, exceto em duas funções (duas versões da função Shekel), em que a versão melhorada do algoritmo genético clássico (AGI) conseguiu superar o BRKGA tradicional. O BRKGA-LS alcançou os melhores resultados dos erros médios absolutos e relativos em todas as funções analisadas.

Estas análises motivaram ainda mais a escolha desta metaheurística para a pesquisa realizada neste trabalho.

4.3.2 Comparação Entre as Abordagens do BRKGA

A segunda fase da análise dos resultados deu-se pela comparação entre as variações do BRKGA que foram desenvolvidas neste trabalho, a fim de verificar se foi possível obter melhorias significativas com as modificações propostas.

Primeiro as funções foram divididas em grupos de acordo com suas características de tamanho e modalidade. Para tanto, foram criados 6 grupos:

- **Grupo 1:** Funções com 2 variáveis de decisão e unimodais;
- **Grupo 2:** Funções com 2 variáveis de decisão e multimodais;
- **Grupo 3:** Funções com 4 variáveis de decisão e unimodais;
- **Grupo 4:** Funções com 4 variáveis de decisão e multimodais;

Tabela 6 – Comparação entre o BRKGA, o BRKGA-LS e os resultados do AG e do AGI descritos no trabalho de (ANDRE; SIARRY; DOGNON, 2001).

Funções	**n**	**Resultados**	**Métodos**			
			BRKGA	**BRKGA-LS**	**AG**	**AGI**
Branin	2	Mínimo	0.39789	0.39789	0.39789	0.39791
		Máximo	0.39891	0.39789	0.43755	0.40183
		Erro médio relativo	**0.001**	**0.000**	0.76	0.48
		Erro médio absoluto	**0.000**	**0.000**	0.003	0.002
Goldstein e Price	2	Mínimo	3.00009	3.00001	3.00000	3.00028
		Máximo	3.00261	3.00134	13.06440	3.02959
		Erro médio relativo	**0.000**	**0.000**	7.62	0.43
		Erro médio absoluto	**0.001**	**0.000**	0.229	0.013
Six-Hump CamelBack	2	Mínimo	-1.03161	-1.03163	-1.03163	-1.03163
		Máximo	-1.03063	-1.03161	-1.02138	-1.02138
		Erro médio relativo	**0.000**	**0.000**	0.44	0.44
		Erro médio absoluto	**0.000**	**0.000**	0.005	0.005
Shubert	2	Mínimo	-186.72974	-186.73091	-186.73100	-186.72802
		Máximo	-186.54330	-186.71885	-175.90800	-184.87532
		Erro médio relativo	**0.001**	**0.000**	0.32	0.49
		Erro médio absoluto	**0.11**	**0.003**	0.596	0.921
Shekel (5)	4	Mínimo	-10.15249	-10.15295	-10.13490	-10.14866
		Máximo	-5.05518	-10.14804	-1.42461	-9.69071
		Erro médio relativo	**0.034**	**0.000**	59.76	0.71
		Erro médio absoluto	0.344	**0.002**	6.067	**0.072**
Shekel (7)	4	Mínimo	-10.40221	-10.40241	-10.16770	-10.38253
		Máximo	-5.08763	-10.39767	-1.60922	-5.12876
		Erro médio relativo	**0.017**	**0.000**	46.68	1.58
		Erro médio absoluto	0.183	**0.002**	4.856	**0.165**
Shekel (10)	4	Mínimo	-10.53173	-10.53085	-10.40340	-10.51404
		Máximo	-10.51966	-10.52555	-1.82221	-10.43177
		Erro médio relativo	**0.001**	**0.000**	48.65	0.70
		Erro médio absoluto	**0.011**	**0.007**	5.126	0.074

- **Grupo 5:** Funções com 10 variáveis de decisão e unimodais;
- **Grupo 6:** Funções com 10 variáveis de decisão e multimodais.

Os resultados desta análise podem sem verificados na Tabela 7, onde estão registrados, para cada grupo, os valores dos *GAPs* mínimos, máximos, médios, os desvios-padrões (D.P) e os percentuais de iterações que atingiram o critério de parada de qualidade (%QUAL) definido pela Equação 4.1.

Como é possível observar, a versão BRKGA-Levy-LS, que utiliza tanto o procedimento de busca local quanto a distribuição de Levy na geração das chaves aleatórias apresentou os melhores resultados em termos de qualidade, em relação às demais versões, em quase

Tabela 7 – Comparação entre as abordagens do BRKGA

Grupo de Funções	**GAP**	**Métodos**			
		BRKGA	**BRKGA-Levy**	**BRKGA-LS**	**BRKGA-Levy-LS**
Grupo 1 (n = 2)	Mínimo	0.00000	0.00000	0.00000	0.00000
UNIMODAIS	Máximo	4.00206	0.00194	8.94870	0.00027
	Média	0.40105	0.00058	0.89487	0.00004
	D.P.	1.20034	0.00053	2.99144	0.01653
	%QUAL	82.67	94.67	90.00	**100.00**
Grupo 2 (n=2)	Mínimo	0.00018	0.00000	0.00000	0.00000
MULTIMODAIS	Máximo	1.05594	1.04077	1.00001	0.00195
	Média	0.08587	0.08317	0.12606	0.00024
	D.P.	0.27043	0.26687	1.00001	0.04413
	%QUAL	71.67	84.05	81.19	**100.00**
Grupo 3 (n = 4)	Mínimo	0.00071	0.00059	0.00004	0.00003
UNIMODAIS	Máximo	20.08117	0.56966	100.75711	0.00106
	Média	2.62142	0.08015	12.59490	0.00042
	D.P.	6.60228	0.18564	10.03778	0.03251
	%QUAL	30.42	58.33	87.50	**98.33**
Grupo 4 (n=4)	Mínimo	0.00059	0.00000	0.00002	0.00000
MULTIMODAIS	Máximo	0.39462	0.05113	351.01688	0.00182
	Média	0.10504	0.01101	31.91171	0.00063
	D.P.	0.13552	0.01714	18.73544	0.04266
	%QUAL	52.73	84.24	90.61	**100.00**
Grupo 5 (n = 10)	Mínimo	0.01498	0.00083	0.00012	0.00033
UNIMODAIS	Máximo	3.95e+12	4.46e+11	1.61e+10	5.52e+09
	Média	5.65e+11	6.38e+10	2.3e+09	7.89e+08
	D.P.	1.38e+12	1.56e+11	1.27e+05	7.43e+04
	%QUAL	1.90	21.43	57.14	**71.43**
Grupo 6 (n = 10)	Mínimo	0.00386	0.00013	0.00014	0.00062
MULTIMODAIS	Máximo	1302.18217	24.96266	1.49e+06	21.25683
	Média	236.95022	4.16527	2.49e+05	3.55018
	D.P.	478.29324	9.30088	1222.40893	4.61051
	%QUAL	12.22	**76.67**	65.00	70.00

todos os grupos de funções, exceto na execução das funções multimodais maiores (10 variáveis), na qual a versão BRKGA-Levy obteve resultados melhores.

Fazendo uma análise por grupo de funções, é possível observar que, para as funções pequenas (2 dimensões) pertencentes aos grupos 1 e 2, as versões que utilizam distribuição de Levy (BRKGA-Levy e BRKGA-Levy-LS) obtiveram os melhores resultados em termos de qualidade das soluções. Esta característica se repetiu também no grupo 6, composto pelas funções maiores (10 dimensões) e multimodais, porém, neste caso, a versão BRKGA-Levy conseguiu resultados melhores que as demais.

É importante enfatizar a relevância da adição da distribuição de Levy para as versões do BRKGA disponíveis na literatura. Na comparação entre as duas versões sem busca local (BRKGA e BRKGA-Levy), a versão que usa a distribuição obteve resultados melhores

que a versão original em termos de qualidade para todos os grupos de funções. Analisando separadamente também as duas versões com busca local (BRKGA-LS e BRKGA-Levy-LS), a versão que utiliza a distribuição de Levy apresentou resultados melhores na tabela. O BRKGA original apresentou os piores resultados em todos os casos analisados.

Desempenho e Convergência

Para analisar o desempenho e a convergência das abordagens do BRKGA, foram utilizados gráficos de *TTT-Plots*, dos quais alguns foram anexados nesta seção, além das tabelas de resultados disponíveis no Apêndice D.

Para demonstrar esta análise, foi selecionada uma função de cada um dos grupos definidos na Seção 4.3.2 para cada versão do BRKGA estudada neste trabalho.

Os gráficos podem ser analisados na Figura 18.

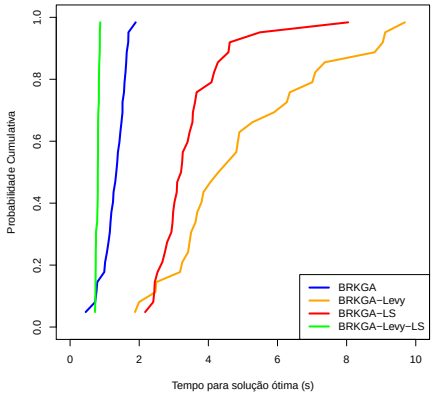
Nas funções selecionadas, a versão BRKGA-Levy-LS conseguiu o melhor desempenho para todos os grupos, conseguindo alcançar o *target* com 100% de probabilidade em menor tempo de execução do que as demais. Na Figura 18 (a), a qual apresenta o gráfico de *TTT-Plot* de uma função do grupo 1, a versão tradicional do BRKGA obteve um bom desempenho, alcançando o target com 100% de probabilidade em aproximadamente 2 segundos, sendo superada apenas pela versão BRKGA-Levy-LS. A versão que utiliza apenas a distribuição de Levy (BRKGA-Levy) obteve um pior resultado em termos de desempenho, necessitando de 30 segundos para alcançar o *target* com probabilidade de 100%, apesar de ter melhorado a qualidade das soluções do BRKGA.

Na Figura 18 (b), que demonstra o gráfico de uma função do grupo 2, a versão BRKGA-LS obteve o melhor resultado em termos de desempenho enquanto, mais uma vez, a versão BRKGA-Levy obteve o pior desempenho, necessitando de mais de 200 segundos para que o target fosse alcançado com probabilidade de 100%.

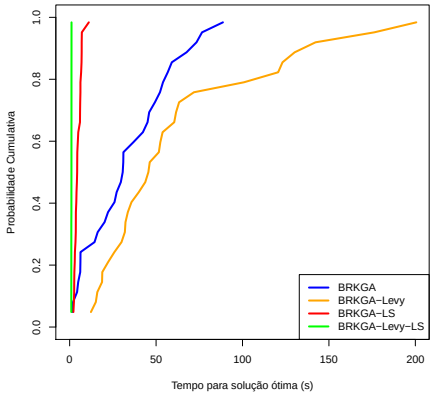
Nas Figuras 18 (c) e (d), que representam funções de dimensão média (4 dimensões) dos grupos 3 e 4, o desempenho da versão BRKGA-LS foi o que mais se aproximou da versão BRKGA-Levy-LS, porém esta segunda ainda conseguiu resultados melhores. A versão tradicional do BRKGA apresentou os piores desempenhos.

Na Figura 18 (e), a qual apresenta o gráfico de uma função do grupo 5, a versão BRKGA-Levy obteve um desempenho ruim e a versão BRKGA-LS conseguiu bons resultados em termos de melhor desempenho e convergência, porém com uma diferença grande em relação ao BRKGA-Levy-LS (aproximadamente 500 segundos para alcançar o target com 100% de probabilidade).

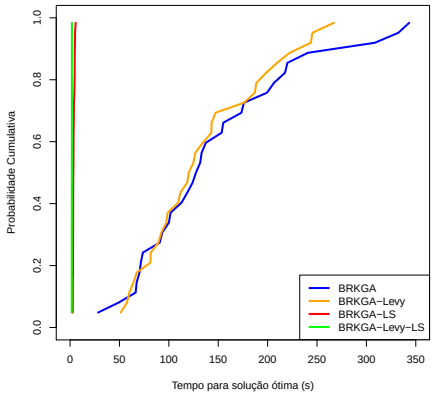
Por fim, na Figura 18 (f) está demonstrado o gráfico de uma função do grupo 6. Para esta função, as versões BRKGA-Levy-LS, BRKGA-LS e BRKGA-Levy obtiveram resultados melhores e mais próximos entre si, enquanto a versão tradicional do BRKGA obteve os piores resultados.



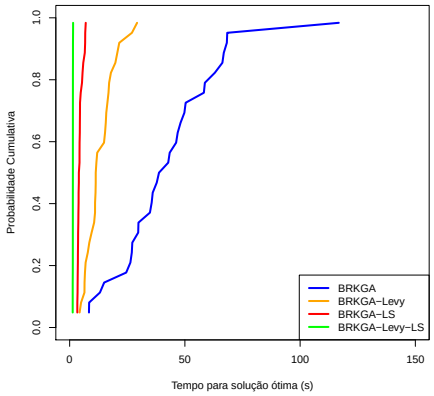
(a) TTT-Plot da execução das versões do BRKGA para a Função Unimodal Powell, com n=2



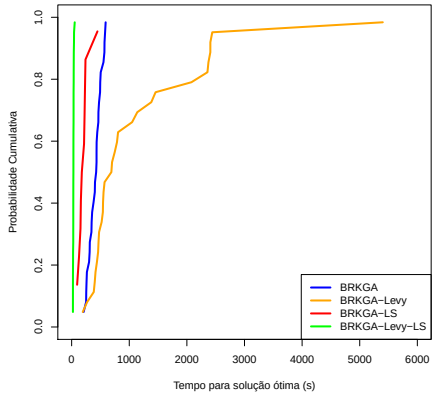
(b) TTT-Plot da execução das versões do BRKGA para a Função Multimodal Six-Hump CamelBack, com n=2



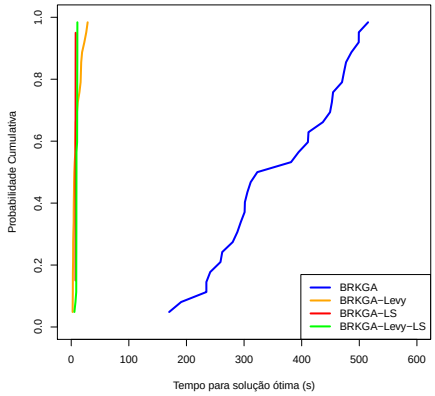
(c) TTT-Plot da execução das versões do BRKGA para a Função Unimodal Sum of Squares, com n=4



(d) TTT-Plot da execução das versões do BRKGA para a Função Multimodal Alpine 2, com n=4



(e) TTT-Plot da execução das versões do BRKGA para a Função Unimodal Levy, com n=10



(f) TTT-Plot da execução das versões do BRKGA para a Função Multimodal Sphere, com n=10

Figura 18 – Exemplos de *TTT-Plots* das versões do BRKGA para algumas funções testadas.

A partir da análise das tabelas e dos *TTT-Plots* das funções testadas, foi possível perceber que a inserção da distribuição de Levy melhorou a qualidade dos resultados do BRKGA tradicional, porém, ocasionou um aumento no tempo de execução do algoritmo em alguns casos de teste. Já ao combinar o uso da distribuição de Levy com o procedimento de busca local no BRKGA-Levy-LS, o ganho de desempenho foi bastante significativo, inclusive em comparação com a versão do BRKGA que utiliza apenas a busca local (BRKGA-LS).

Testes Estatísticos

- **Qualidade das soluções:**

  Para confirmar se realmente os resultados das quatro versões do BRKGA em termos de qualidade das soluções e de desempenho foram significativamente diferentes, foi utilizado o teste não-paramétrico de *Friedman* com nível de significância de 0.05.

  Para o teste da qualidade, as hipóteses foram definidas da seguinte maneira:

  $$H_0: \quad \text{Os algoritmos apresentaram a mesma qualidade;}$$
  $$H_1: \quad \text{Os algoritmos apresentaram qualidades diferentes;}$$

  Como hipótese nula, definimos que as abordagens possuem qualidades iguais, logo, se o resultado do *p-value* for menor que 0.05, significa que a hipótese nula foi rejeitada e podemos afirmar com 95% de certeza que houve diferença de qualidade entre pelo menos um par das quatro amostras analisadas.

  O detalhamento do resultado do teste está descrito na Tabela 8, na qual o ranking mais baixo indica a melhor abordagem e o mais alto, a pior.

Tabela 8 – Teste de Friedman para a qualidade das soluções

Teste de Friedman	
F-Value:	42.783321
p-Value:	0.000000
Rankings Médios	
Técnica	Ranking
BRKGA	3.562500
BRKGA-Levy	2.803571
BRKGA-LS	1.946429
BRKGA-Levy-LS	**1.687500**

Como podemos ver, o *p-value* retornado na execução do teste obteve resultado 0.0, confirmando que os algoritmos apresentam qualidades estatisticamente diferentes. O

teste também gerou um *rankeamento*, no qual o BRKGA-Levy-LS ficou em primeiro lugar.

Como a hipótese nula do teste de *Friedman* foi rejeitada, foi realizado o pós teste *Nemenyi*, que compara os blocos de dados um a um a fim de mensurar a significância da diferença entre eles. A partir do valor do *p-value* de cada comparação podemos determinar as diferenças entre os resultados dos algoritmos. Valores maiores que 0.05 significam que os algoritmos obtiveram desempenhos iguais. Valores menores que 0.05 implicam em diferenças significativas dos desempenhos. O resultado do teste se encontra na Tabela 9.

Tabela 9 – Pós-Teste de Nemenyi para a qualidade das soluções

Pós-Teste Nemenyi	
BRKGA X BRKGA-Levy-LS	
Z-value:	7.685213
p-Value:	0.000000
p-value ajustado	9.1926466439e-14
BRKGA-LS X BRKGA	
Z-value:	6.623922
p-Value:	0.000000
p-value ajustado:	2.09874784218e-10
BRKGA-Levy X BRKGA-Levy-LS	
Z-value:	4.574532
p-Value:	0.000005
p-value ajustado:	2.86371953071e-05
BRKGA-LS X BRKGA-Levy	
Z-value:	3.513240
p-Value:	0.000443
p-value ajustado:	0.00265606191806
BRKGA-Levy X BRKGA	
Z-value:	3.110681
p-Value:	0.001867
p-value ajustado:	0.0111993694666
BRKGA-LS X BRKGA-Levy-LS	
Z-value:	1.061291
p-Value:	0.288558
p-value ajustado:	1

De acordo com os resultados dos testes, é possível confirmar a contribuição da distribuição de Levy para a qualidade dos resultados da versão tradicional do

BRKGA, em que o *p-value* da comparação da versão BRKGA com a versão BRKGA-Levy foi de 0.001867, menor que o nível de significância 0.05.

No teste *Nemenyi*, o BRKGA-Levy e o BRKGA-Levy-LS apresentaram qualidades similares, com o resultado do $p-value = 0.288558 \geq 0.05$, indicando que a diferença estatística entre as duas abordagens não é significativa. Esta situação ocorreu devido à natureza do experimento realizado, que foi planejado para que os algoritmos tentassem atingir o critério de parada de qualidade, com base na proximidade da solução ótima ou da melhor solução até então encontrada na literatura para os problemas, não levando em consideração a eficiência para alcançar tais resultados. Haja vista que, tanto o BRKGA-LS quanto o BRKGA-Levy-LS conseguem convergir aos ótimos globais na maioria das funções, evidentemente ambas alcançaram resultados estatisticamente equivalentes para a qualidade das soluções. Para diferenciar o grau de eficiência da convergência entre o BRKGA-LS e o BRKGA-Levy-LS foi realizado um novo experimento mais adequado, em que os algoritmos foram testados mediante um tempo de execução limite, determinado com base nos tempos médios de execução alcançados pelo BRKGA-Levy-LS. Para os grupos 1 e 2 foi utilizado um tempo limite de 1 segundo. Para os grupos 3 e 4 o tempo limite foi configurado como 4 segundos e para os grupos 5 e 6, o tempo limite de execução foi de 10 segundos. Os resultados obtidos neste experimentos podem ser analisados na Tabela 10. Como podemos ver, o BRKGA-Levy-LS alcançou os melhores resultados médios em todos os grupos de funções, sem exceção. Para verificar a diferença estatística entre as duas abordagens, foi realizado o teste de *Wilcoxon Signed-Rank* pareado que retornou o $p-value = 4.6334739512045839e-28 \leq 0.05$, confirmando a diferença estatística e validando a hipótese de que a distribuição de Levy também contribuiu significativamente com a melhoria do algoritmo BRKGA-LS.

- **Tempos de Execução:**

  Também foram executados os testes estatísticos para os tempos de execução das abordagens. As hipóteses para o teste de *Friedman* foram definidas da seguinte maneira:

  $$H_0: \quad \text{Os algoritmos apresentaram o mesmo desempenho;}$$

  $$H_1: \quad \text{Os algoritmos apresentaram desempenhos diferentes;}$$

  O resultado do teste está detalhado na Tabela 11, onde podemos observar que a diferença estatística entre os resultados é verificada ou não a partir do *p-value*.

  Ainda de acordo com a saída do teste de *Friedman*, podemos observar que o BRKGA-Levy-LS ficou em primeiro lugar no ranking médio dos desempenhos.

Tabela 10 – Comparação entre as abordagens BRKGA-LS e BRKGA-Levy-LS

Grupo de Funções	**GAP**	**Métodos**	
		BRKGA-LS	**BRKGA-Levy-LS**
Grupo 1 (n = 2)	Mínimo	0.00000	0.00000
	Máximo	8.97203	0.00070
	Média	0.89721	**0.00008**
	D.P.	2.69161	0.00021
Grupo 2 (n=2)	Mínimo	0.00000	0.00000
	Máximo	23.28808	0.33333
	Média	1.80389	**0.02625**
	D.P.	6.20208	0.08865
Grupo 3 (n = 4)	Mínimo	0.00003	0.00003
	Máximo	100.88374	0.64874
	Média	12.64017	**0.09554**
	D.P.	33.35299	0.21073
Grupo 4 (n=4)	Mínimo	0.00001	0.00000
	Máximo	635.80501	1.23121
	Média	64.05088	**0.15208**
	D.P.	190.58801	0.36630
Grupo 5 (n = 10)	Mínimo	0.00066	0.00076
	Máximo	1.141e+13	2.865e+12
	Média	1.63e+12	**4.093e+11**
	D.P.	3.993e+12	1.002e+12
Grupo 6 (n = 10)	Mínimo	0.00015	0.00062
	Máximo	3468203.89387	21.25683
	Média	579435.14978	**4.97629**
	D.P.	1291900.27810	7.74088

Tabela 11 – Teste de Friedman para o desempenho dos algoritmos.

Teste de Friedman	
F-Value:	35.058479
p-Value:	0.000000
Rankings Médios	
Técnica	Ranking
BRKGA	2.750000
BRKGA-Levy	3.142857
BRKGA-LS	2.785714
BRKGA-Levy-LS	**1.321429**

Assim como no teste da qualidade, como a hipótese nula foi rejeitada fez-se necessária a realização do pós-teste *Nemenyi* para verificar a diferença estatística dos desempenhos par a par. O resultado deste teste está descrito na Tabela 12

Por meio da análise do pós-teste para o desempenho dos algoritmos, verificamos que o BRKGA-Levy-LS obteve diferença significativa em relação às demais abordagens, obtendo o resultado do *p-value* menor que 0.05 em todas as comparações e confirmando a contribuição de melhoria no desempenho do algoritmo. As comparações par a par entre os outros 3 algoritmos (BRKGA, BRKGA-Levy e BRKGA-LS) obtiveram resultados de *p-value* maiores que o nível de significância de 0.05, demonstrando que a diferença entre eles em termos de desempenho não são relevantes.

4.3.3 Comparação com o Algoritmo *Continuous-GRASP*

A terceira fase da análise experimental consistiu na comparação da versão do BRKGA que apresentou os melhores resultados nos experimentos da segunda fase, no caso o BRKGA-Levy-LS, com uma outra técnica da literatura, o *Continuous GRASP* ou apenas *C-GRASP* (HIRSCH; PARDALOS; RESENDE, 2010).

No presente trabalho, foi utilizada a implementação do C-GRASP de Silva et al. (2017), que utilizou Python embarcado em C++ para implementar a biblioteca do C-GRASP descrita em Silva et al. (2013b), a qual por sua vez propõe a utilização de Python embarcado em C). Por fim, parâmetros utilizados para executar o C-GRASP foram ajustados de acordo com as configurações sugeridas em Hirsch, Pardalos e Resende (2010).

Foram selecionadas 10 funções para esta análise, de tal forma que os testes fossem realizados em funções de dimensão 2, 4 e 10. Foram selecionadas funções em comum entre o conjunto utilizado neste trabalho e as funções utilizadas em Hirsch, Pardalos e Resende (2010).

Assim como nos experimentos com as versões do BRKGA, C-GRASP também foi executado 30 vezes para cada função e seus resultados em termos de qualidade estão deta-

Tabela 12 – Pós-Teste de Nemenyi para o desempenho dos algoritmos.

Pós-Teste Nemenyi	
BRKGA-Levy X BRKGA-Levy-LS	
Z-value:	7.465636
p-Value:	0.000000
p-value ajustado:	4.96935825822e-13
BRKGA-LS X BRKGA-Levy-LS	
Z-value:	6.001785
p-Value:	0.000000
p-value ajustado:	1.17095679819e-08
BRKGA X BRKGA-Levy-LS	
Z-value:	5.855400
p-Value:	0.000000
p-value ajustado	2.85517223197e-08
BRKGA-Levy X BRKGA	
Z-value:	1.610235
p-Value:	0.107347
p-value ajustado:	0.644079221963
BRKGA-LS X BRKGA-Levy	
Z-value:	1.463850
p-Value:	0.143235
p-value ajustado:	0.859409445148
BRKGA-LS X BRKGA	
Z-value:	0.146385
p-Value:	0.883617
p-value ajustado:	1

lhados na Tabela 13. Como podemos analisar, BRKGA-Levy-LS atingiu bons resultados, conseguindo alcançar o critério de parada de qualidade em 100% das execuções em todos os experimentos. Já o C-GRASP, embora também tenha apresentado bons resultados, apenas em 50% das funções analisadas o critério de qualidade foi alcançado nas 30 execuções.

Tabela 13 – Comparação entre o BRKGA-Levy-LS e o C-GRASP.

Funções	**n**	**Resultados**	**Métodos**	
			C-GRASP	**BRKGA-Levy-LS**
Bohachevsky	2	Mínimo	0.00003	0.00000
		Máximo	0.00093	0.00029

Continua na próxima página

Funções	**n**	**Resultados**	**Métodos**	
			C-GRASP	**BRKGA-Levy-LS**
		Média	0.00047	0.00010
		D.P.	0.00029	0.00010
		%QUAL	**100.00**	**100.00**
Booth	2	Mínimo	0.00000	0.00000
		Máximo	0.00071	0.00001
		Média	0.00025	0.00000
		D.P.	0.00019	0.00000
		%QUAL	**100.00**	**100.00**
Branin	2	Mínimo	0.39789	0.39789
		Máximo	0.39826	0.39789
		Média	0.39801	0.39789
		D.P.	0.00011	0.0000
		%QUAL	**100.00**	**100.00**
Colville	4	Mínimo	0.00045	0.00015
		Máximo	0.00245	0.00100
		Média	0.00105	0.00059
		D.P.	0.00044	0.00020
		%QUAL	43.33	**100.00**
Perm0	4	Mínimo	0.00007	0.00013
		Máximo	0.00100	0.00096
		Média	0.00049	0.00043
		D.P.	0.00028	0.00024
		%QUAL	**100.00**	**100.00**
Power Sum	4	Mínimo	0.00006	0.00011
		Máximo	0.00095	0.00097
		Média	0.00057	0.00062
		D.P.	0.00022	0.00022
		%QUAL	**100.00**	**100.00**
Rastrigin	10	Mínimo	0.15991	0.00061

Continua na próxima página

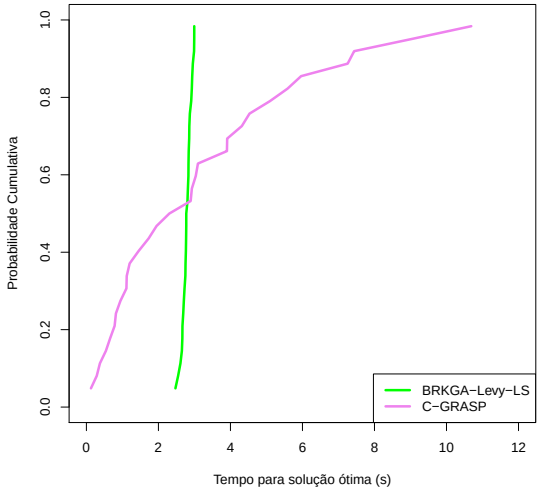
Funções	**n**	**Resultados**	**Métodos**	
			C-GRASP	**BRKGA-Levy-LS**
		Máximo	0.45480	0.00099
		Média	0.34767	0.00087
		D.P.	0.00010	0.00010
		%QUAL	0.00	**100.00**
Sum of Squares	10	Mínimo	0.00351	0.00052
		Máximo	0.00924	0.00096
		Média	0.00605	0.00077
		D.P.	0.00136	0.00013
		%QUAL	0.00	**100.00**
Trid	10	Mínimo	-124.66656	-209.97797
		Máximo	-124.66580	-209.79125
		Média	-124.66630	-209.81793
		D.P.	0.00015	0.04203
		%QUAL	0.00	**100.00**
Zakharov	10	Mínimo	0.00085	0.00041
		Máximo	0.00370	0.00099
		Média	0.00263	0.00075
		D.P.	0.00059	0.00015
		%QUAL	3.33	**100.00**

Desempenho e Convergência

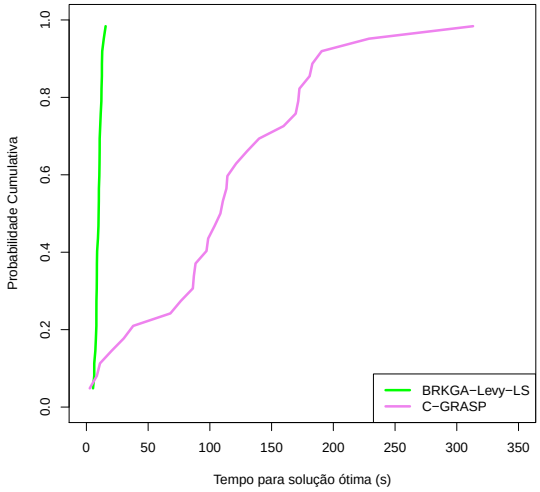
Assim como na comparação entre as abordagens do BRKGA, utilizamos gráficos de *TTT-Plot* juntamente com as tabelas de tempos computacionais como apoio (Tabela 20) para comparar o desempenho e a convergência entre as técnicas BRKGA-Levy-LS e C-GRASP nas funções experimentadas.

Para exemplificar a análise realizada, foram escolhidas 3 funções de tamanhos diferentes dentre as 10 testadas. Os *TTT-Plots* para estas funções podem ser visualizados nas Figuras 19 (a), (b) e (c).

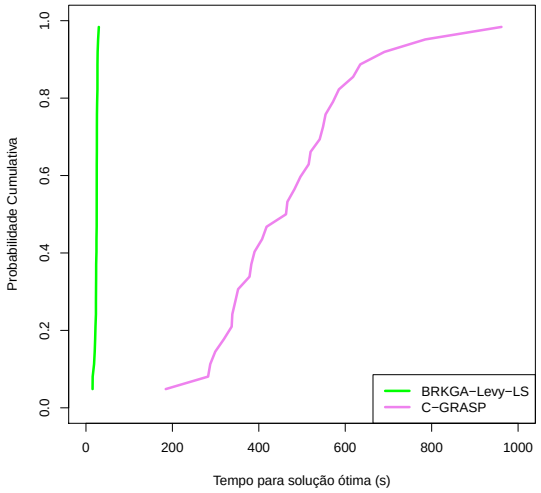
Nos três gráficos é possível verificar que o BRKGA-Levy-LS conseguiu convergir mais rápido do que o C-GRASP. No gráfico da Figura 19 (a), podemos observar que com 1 segundo o C-GRASP consegue alcançar o *target* com aproximadamente 40% de probabilidade enquanto o BRKGA-Levy-LS necessita de pouco mais de 2 segundos. Porém,



(a) TTT-Plot da execução do BRKGA-Levy-LS e do C-GRASP para a Função Bohachevsky, com n=2



(b) TTT-Plot da execução do BRKGA-Levy-LS e do C-GRASP para a Função Colville, com n=4



(c) TTT-Plot da execução do BRKGA-Levy-LS e do C-GRASP para a Função Zakharov, com n=10

Figura 19 – Exemplos de TTT-Plots da execução do BRKGA-Levy-LS e do C-GRASP em algumas funções testadas.

o BRKGA-LS alcança a probabilidade de 100% com aproximadamente 3 segundos enquanto o C-GRASP necessita de pouco mais de 10 segundos. Nos gráficos das Figuras 19 (b) e (c), a diferença entre o desempenho do C-GRASP e do BRKGA-Levy-LS é ainda maior, onde observamos que o C-GRASP precisou de um tempo computacional elevado para alcançar o *target* com 100% de probabilidade enquanto o BRKGA alcançou esta convergência em poucos segundos.

A partir da análise das tabelas e dos *TTT-Plots* de todas as funções, foi observado que o BRKGA-Levy-LS obteve melhor desempenho na maioria dos testes realizados, obtendo resultados piores que o C-GRASP em apenas 3 das 10 funções testadas (*Booth*, *Branin* e *Perm0*).

Testes Estatísticos

- **Qualidade das soluções:**

  Para fins de validação da diferença estatística entre a qualidade dos resultados e o desempenho das duas metaheurísticas comparadas, foi realizado o teste estatístico de *Wilcoxon Signed-Rank* pareado, considerando um nível de significância de 0.05.

  Assumimos como hipótese nula que os algoritmos apresentam a mesma qualidade dos resultados e como hipótese alternativa o contrário.

  $$H_0: \quad \text{Os algoritmos apresentam a mesma qualidade.}$$
  $$H_1: \quad \text{Os algoritmos apresentam qualidades diferentes.}$$

  O teste obteve como resultado o $p - value = 6.829e - 38 < 0.05$, confirmando a diferença estatística entre os resultados das duas abordagens. Portanto, o algoritmo BRKGA-Levy-LS apresentou uma qualidade de resultados significativamente melhor que o C-GRASP com 95% de certeza.

- **Tempos de Execução**

  Da mesma maneira, foi realizado o teste estatístico de *Wilcoxon Signed-Rank* pareado para o desempenho computacional entre as duas abordagens. No caso, assumimos como hipótese nula que os algoritmos apresentam o mesmo desempenho e como hipótese alternativa o contrário.

  $$H_0: \quad \text{Os algoritmos apresentam o mesmo desempenho.}$$
  $$H_1: \quad \text{Os algoritmos apresentam desempenhos diferentes.}$$

  O *p-value* do teste foi igual a 1.497e-08, o que confirma a diferença estatística entre os desempenhos das duas metaheurísticas com nível de significância de 0.05.

Dessa forma, podemos afirmar com 95% de certeza que o BRKGA-Levy-LS também superou o C-GRASP em termos de desempenho.

5 CONCLUSÕES E TRABALHOS FUTUROS

Este trabalho teve como principal objetivo analisar a hipótese de que a utilização da distribuição de Levy como alternativa para a geração de chaves aleatórias do BRKGA é adequada para problemas de otimização global com restrições de caixa.

Os algoritmos foram validados utilizando 27 funções de *benchmark* de otimização global, clássicas para a avaliação de algoritmos na literatura, além de serem funções-base para as principais competições de otimização global, tais como a *IEEE CEC Competition*. As abordagens do BRKGA desenvolvidas foram comparadas entre si e também com a metaheurística C-GRASP.

Por meio da análise dos experimentos, conclui-se que a distribuição de Levy introduziu contribuições relevantes para a melhoria da qualidade dos resultados do BRKGA original e inclusive quando inserida na versão com busca local (BRKGA-LS), as modificações trouxeram melhorias significativas para a eficiência da convergência do algoritmo aos mínimos globais.

Por fim, foi realizada a comparação da melhor versão do BRKGA com o C-GRASP, a fim de avaliar a relevância deste trabalho para a área de otimização global. Verificou-se que, nos experimentos realizados, os resultados do BRKGA-Levy-LS conseguiram se equiparar ou superar os do C-GRASP, tanto em termos de qualidade quanto em desempenho, confirmando a relevância da abordagem aqui proposta para o estado da arte da otimização global.

Quanto aos trabalhos futuros, podemos citar:

- A aplicação do algoritmo a outros tipos de problemas para os quais o BRKGA também pode ser aplicado e analisar o impacto da utilização da distribuição de Levy em outras metaheurísticas da literatura;
- A implementação de melhorias na busca local com o intuito de aperfeiçoar o desempenho e a convergência do algoritmo, baseadas em duas ideias:
  - Adaptação da densidade de discretização do espaço de busca de acordo com o intervalo de domínio das variáveis de decisão;
  - Utilização de informações de direção e sentido do espaço de busca em que são encontradas melhorias em relação a uma solução atual durante o procedimento da busca local.

REFERÊNCIAS

ACOSTA-MESA, H.-G.; RECHY-RAMÍREZ, F.; MEZURA-MONTES, E.; CRUZ-RAMÍREZ, N.; JIMÉNEZ, R. H. Application of time series discretization using evolutionary programming for classification of precancerous cervical lesions. *Journal of biomedical informatics*, Elsevier, v. 49, p. 73–83, 2014.

AIEX, R. M.; RESENDE, M. G.; RIBEIRO, C. C. Probability distribution of solution time in grasp: An experimental investigation. *Journal of Heuristics*, Springer, v. 8, n. 3, p. 343–373, 2002.

ANDRE, J.; SIARRY, P.; DOGNON, T. An improvement of the standard genetic algorithm fighting premature convergence in continuous optimization. *Advances in engineering software*, Elsevier, v. 32, n. 1, p. 49–60, 2001.

BATTITI, R.; TECCHIOLLI, G. Parallel biased search for combinatorial optimization: genetic algorithms and tabu. *Microprocessors and Microsystems*, Elsevier, v. 16, n. 7, p. 351–367, 1992.

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

BELLMAN, R. *Dynamic programming*. [S.l.]: Courier Corporation, 2013.

BENAVIDES, A. J.; RITT, M. Iterated local search heuristics for minimizing total completion time in permutation and non-permutation flow shops. In: *ICAPS*. [S.l.: s.n.], 2015. p. 34–41.

BIRATTARI, M.; STÜTZLE, T.; PAQUETE, L.; VARRENTRAPP, K. A racing algorithm for configuring metaheuristics. In: MORGAN KAUFMANN PUBLISHERS INC. *Proceedings of the 4th Annual Conference on Genetic and Evolutionary Computation*. [S.l.], 2002. p. 11–18.

CESCHIA, S.; SCHAERF, A.; STÜTZLE, T. Local search techniques for a routing-packing problem. *Computers & industrial engineering*, Elsevier, v. 66, n. 4, p. 1138–1149, 2013.

CHARLES, D. On the origin of species by means of natural selection. *Murray, London*, 1859.

CHIVILIKHIN, D.; ULYANTSEV, V.; SHALYTO, A. A. Modified ant colony algorithm for constructing finite state machines from execution scenarios and temporal formulas. *Automation and Remote Control*, Springer, v. 77, n. 3, p. 473–484, 2016.

CLEVELAND, G. A.; SMITH, S. F. Using genetic algorithms to schedule flow shop releases. In: *Proceedings of the 3rd International Conference on Genetic Algorithms*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1989. p. 160–169. ISBN 1-55860-066-3. Disponível em: <http://dl.acm.org/citation.cfm?id=645512.657259>.

CORANA, A.; MARCHESI, M.; MARTINI, C.; RIDELLA, S. Minimizing multimodal functions of continuous variables with the "simulated annealing" algorithm corrigenda for this article is available here. *ACM Transactions on Mathematical Software (TOMS)*, ACM, v. 13, n. 3, p. 262–280, 1987.

DEKKERS, A.; AARTS, E. Global optimization and simulated annealing. *Mathematical programming*, Springer, v. 50, n. 1, p. 367–393, 1991.

DELL, M.; IORI, M.; NOVELLANI, S.; STÜTZLE, T. et al. A destroy and repair algorithm for the bike sharing rebalancing problem. *Computers & Operations Research*, Elsevier, v. 71, p. 149–162, 2016.

DODD, N. Slow annealing versus multiple fast annealing runs—an empirical investigation. *Parallel Computing*, Elsevier, v. 16, n. 2-3, p. 269–272, 1990.

DORIGO, M.; BIRATTARI, M. Ant colony optimization. In: *Encyclopedia of machine learning*. [S.l.]: Springer, 2011. p. 36–39.

DORIGO, M.; MANIEZZO, V.; COLORNI, A. Ant system: optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, IEEE, v. 26, n. 1, p. 29–41, 1996.

EBERHART, R.; KENNEDY, J. A new optimizer using particle swarm theory. In: IEEE. *Micro Machine and Human Science, 1995. MHS'95., Proceedings of the Sixth International Symposium on*. [S.l.], 1995. p. 39–43.

EIKELDER, H. T.; VERHOEVEN, M.; VOSSEN, T.; AARTS, E. A probabilistic analysis of local search. In: *Meta-Heuristics*. [S.l.]: Springer, 1996. p. 605–618.

FEO, T. A.; RESENDE, M. G. Greedy randomized adaptive search procedures. *Journal of global optimization*, Springer, v. 6, n. 2, p. 109–133, 1995.

FRIEDMAN, M. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the american statistical association*, Taylor & Francis, v. 32, n. 200, p. 675–701, 1937.

GANDOMI, A. H.; YANG, X.-S.; ALAVI, A. H. Cuckoo search algorithm: a metaheuristic approach to solve structural optimization problems. *Engineering with computers*, Springer, v. 29, n. 1, p. 17–35, 2013.

GAZI, V.; PASSINO, K. M. Stability analysis of social foraging swarms. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, IEEE, v. 34, n. 1, p. 539–557, 2004.

GOFFE, W. L.; FERRIER, G. D.; ROGERS, J. Global optimization of statistical functions with simulated annealing. *Journal of econometrics*, Elsevier, v. 60, n. 1-2, p. 65–99, 1994.

GOLDBERG, D. E. *Genetic algorithms*. [S.l.]: Pearson Education India, 2006.

GOLDBERG, D. E.; LINGLE, R. et al. Alleles, loci, and the traveling salesman problem. In: LAWRENCE ERLBAUM, HILLSDALE, NJ. *Proceedings of an international conference on genetic algorithms and their applications*. [S.l.], 1985. v. 154, p. 154–159.

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

GONÇALVES, J. F.; RESENDE, M. G.; TOSO, R. F. An experimental comparison of biased and unbiased random-key genetic algorithms. *Pesquisa Operacional*, SciELO Brasil, v. 34, n. 2, p. 143–164, 2014.

GREFENSTETTE, J.; GOPAL, R.; ROSMAITA, B.; GUCHT, D. V. Genetic algorithms for the traveling salesman problem. In: *Proceedings of the first International Conference on Genetic Algorithms and their Applications*. [S.l.: s.n.], 1985. p. 160–168.

GREFENSTETTE, J. J. *Incorporating problem specific knowledge into genetic algorithms*. London: [s.n.], 1987. 42–60 p.

GUERGELOT, M. A.; ARTUSO, A. R. Distribuição de lévy aplicado aos retornos do indíce bovespa e das ações da vale e petrobrás. *Revista CIATEC-UPF*, v. 6, n. 2, p. 48–65, 2014.

HART, J. P.; SHOGAN, A. W. Semi-greedy heuristics: An empirical study. *Operations Research Letters*, Elsevier, v. 6, n. 3, p. 107–114, 1987.

HIRSCH, M. J.; MENESES, C.; PARDALOS, P. M.; RESENDE, M. G. Global optimization by continuous grasp. *Optimization Letters*, Springer, v. 1, n. 2, p. 201–212, 2007.

HIRSCH, M. J.; PARDALOS, P. M.; RESENDE, M. G. Speeding up continuous grasp. *European Journal of Operational Research*, Elsevier, v. 205, n. 3, p. 507–521, 2010.

HOLLAND, J. H. *Adaptation in Natural and Artificial Systems*. Ann Arbor, MI: University of Michigan Press, 1975. Second edition, 1992.

HOOS, H. H.; STÜTZLE, T. Some surprising regularities in the behaviour of stochastic local search. *CP*, v. 98, p. 470, 1998.

JACQUIN, S.; JOURDAN, L.; TALBI, E.-G. Dynamic programming based metaheuristic for energy planning problems. In: SPRINGER. *European Conference on the Applications of Evolutionary Computation*. [S.l.], 2014. p. 165–176.

JAMIL, M.; YANG, X.-S. A literature survey of benchmark functions for global optimisation problems. *International Journal of Mathematical Modelling and Numerical Optimisation*, Inderscience Publishers Ltd, v. 4, n. 2, p. 150–194, 2013.

KARABOGA, D.; BASTURK, B. A powerful and efficient algorithm for numerical function optimization: artificial bee colony (abc) algorithm. *Journal of global optimization*, Springer, v. 39, n. 3, p. 459–471, 2007.

KIM, D. H.; ABRAHAM, A.; CHO, J. H. A hybrid genetic algorithm and bacterial foraging approach for global optimization. *Information Sciences*, Elsevier, v. 177, n. 18, p. 3918–3937, 2007.

KIRKPATRICK, S.; GELATT, C. D.; VECCHI, M. P. Optimization by simulated annealing. *science*, American Association for the Advancement of Science, v. 220, n. 4598, p. 671–680, 1983.

LARRAÑAGA, P.; LOZANO, J. A. *Estimation of distribution algorithms: A new tool for evolutionary computation*. [S.l.]: Springer Science & Business Media, 2001. v. 2.

LAWLER, E. L.; WOOD, D. E. Branch-and-bound methods: A survey. *Operations research*, INFORMS, v. 14, n. 4, p. 699–719, 1966.

LEE, C.-Y.; YAO, X. Evolutionary programming using mutations based on the lévy probability distribution. *IEEE Transactions on Evolutionary Computation*, IEEE, v. 8, n. 1, p. 1–13, 2004.

LÓPEZ-IBÁÑEZ, M.; BLUM, C.; OHLMANN, J. W.; THOMAS, B. W. The travelling salesman problem with time windows: Adapting algorithms from travel-time to makespan optimization. *Applied Soft Computing*, Elsevier, v. 13, n. 9, p. 3806–3815, 2013.

LÓPEZ-IBÁÑEZ, M.; DUBOIS-LACOSTE, J.; CÁCERES, L. P.; BIRATTARI, M.; STÜTZLE, T. The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, Elsevier, v. 3, p. 43–58, 2016.

LóPEZ-IBáñEZ, M.; DUBOIS-LACOSTE, J.; Pérez Cáceres, L.; STüTZLE, T.; BIRATTARI, M. The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, v. 3, p. 43–58, 2016.

MARON, O.; MOORE, A. W. The racing algorithm: Model selection for lazy learners. In: *Lazy learning*. [S.l.]: Springer, 1997. p. 193–225.

MESQUITA, R. G.; SILVA, R. M.; MELLO, C. A.; MIRANDA, P. B. Parameter tuning for document image binarization using a racing algorithm. *Expert Systems with Applications*, Elsevier, v. 42, n. 5, p. 2593–2603, 2015.

METROPOLIS, N.; ROSENBLUTH, A. W.; ROSENBLUTH, M. N.; TELLER, A. H.; TELLER, E. Equation of state calculations by fast computing machines. *The journal of chemical physics*, AIP, v. 21, n. 6, p. 1087–1092, 1953.

NANNEN, V.; EIBEN, A. E. A method for parameter calibration and relevance estimation in evolutionary algorithms. In: ACM. *Proceedings of the 8th annual conference on Genetic and evolutionary computation*. [S.l.], 2006. p. 183–190.

NEMENYI, P. *Distribution-free Multiple Comparisons*. [s.n.], 1963. Disponível em: <https://books.google.com.br/books?id=_oozmgEACAAJ>.

OSBORNE, L. J.; GILLETT, B. E. A comparison of two simulated annealing algorithms applied to the directed steiner problem on networks. *ORSA Journal on Computing*, INFORMS, v. 3, n. 3, p. 213–225, 1991.

PAUL, L. Théorie de l'addition des variables aléatoires. *Gauthiers-Villars, Paris*, 1937.

PELLEGRINI, P.; CASTELLI, L.; PESENTI, R. Metaheuristic algorithms for the simultaneous slot allocation problem. *IET Intelligent Transport Systems*, IET, v. 6, n. 4, p. 453–462, 2012.

R Core Team. *R: A Language and Environment for Statistical Computing*. Vienna, Austria, 2015. Disponível em: <https://www.R-project.org/>.

RIBEIRO, C. A. M. based on the work of; Rosseti. *tttplot: Time to Target Plot*. [S.l.], 2016. R package version 1.1.1. Disponível em: <https://CRAN.R-project.org/package=tttplot>.

RIBEIRO, C. C.; ROSSETI, I. tttplots-compare: A perl program to compare time-to-target plots or general runtime distributions of randomized algorithms. *Optimization Letters*, Springer, v. 9, n. 3, p. 601–614, 2015.

SAMA, M.; PELLEGRINI, P.; D'ARIANO, A.; RODRIGUEZ, J.; PACCIARELLI, D. Ant colony optimization for the real-time train routing selection problem. *Transportation Research Part B: Methodological*, Elsevier, v. 85, p. 89–108, 2016.

SELMAN, B.; KAUTZ, H. A.; COHEN, B. Noise strategies for improving local search. In: *AAAI*. [S.l.: s.n.], 1994. v. 94, p. 337–343.

SILVA, R. M.; MOURA, M. A.; RESENDE, M. G.; FACÓ, J. L.; FALCÃO, R. M. Generalized grasp for mixed-integer nonlinear optimization. In: *Proceedings of the 12th Metaheuristics International Conference*. [S.l.: s.n.], 2017. p. 496–504.

SILVA, R. M.; RESENDE, M. G.; PARDALOS, P. M.; FACÓ, J. L. Biased random-key genetic algorithm for nonlinearly-constrained global optimization. In: IEEE. *Evolutionary Computation (CEC), 2013 IEEE Congress on*. [S.l.], 2013. p. 2201–2206.

SILVA, R. M.; RESENDE, M. G.; PARDALOS, P. M.; HIRSCH, M. J. A python/c library for bound-constrained global optimization with continuous grasp. *Optimization Letters*, Springer, v. 7, n. 5, p. 967–984, 2013.

SOCHA, K.; DORIGO, M. Ant colony optimization for continuous domains. *European journal of operational research*, Elsevier, v. 185, n. 3, p. 1155–1173, 2008.

SPEARS, W. M.; JONG, K. D. D. *On the virtues of parameterized uniform crossover*. [S.l.], 1995.

STEFANELLO, F.; AGGARWAL, V.; BURIOL, L. S.; GONÇALVES, J. F.; RESENDE, M. G. A biased random-key genetic algorithm for placement of virtual machines across geo-separated data centers. In: ACM. *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*. [S.l.], 2015. p. 919–926.

STORN, R.; PRICE, K. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*, Springer, v. 11, n. 4, p. 341–359, 1997.

SUN, J.; ZHANG, Q.; TSANG, E. P. De/eda: A new evolutionary algorithm for global optimization. *Information Sciences*, Elsevier, v. 169, n. 3, p. 249–262, 2005.

TOSO, R. F.; RESENDE, M. G. A c++ application programming interface for biased random-key genetic algorithms. *Optimization Methods and Software*, Taylor & Francis, v. 30, n. 1, p. 81–93, 2015.

WILCOXON, F. Individual comparisons by ranking methods. *Biometrics bulletin*, JSTOR, v. 1, n. 6, p. 80–83, 1945.

YANG, X.-S.; DEB, S. Cuckoo search via lévy flights. In: IEEE. *Nature & Biologically Inspired Computing, 2009. NaBIC 2009. World Congress on*. [S.l.], 2009. p. 210–214.

YANG, X.-S.; DEB, S. Engineering optimisation by cuckoo search. *International Journal of Mathematical Modelling and Numerical Optimisation*, Inderscience Publishers, v. 1, n. 4, p. 330–343, 2010.

YANG, X.-S.; DEB, S. Cuckoo search: recent advances and applications. *Neural Computing and Applications*, Springer, v. 24, n. 1, p. 169–174, 2014.

YARIMCAM, A.; ASTA, S.; ÖZCAN, E.; PARKES, A. J. Heuristic generation via parameter tuning for online bin packing. In: IEEE. *Evolving and Autonomous Learning Systems (EALS), 2014 IEEE Symposium on*. [S.l.], 2014. p. 102–108.

APÊNDICE A – FUNÇÕES DE BENCHMARK

(A_n) **Função Ackley** (ackley)
Definição: $A_n(x) = -20e^{-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}} - e^{\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)} + 20 + e$
Domínio: $[-15, 30]^n$
Mínimo Global: $x^* = (0, ..., 0)$; $A_n(x^*) = 0$

($A1$) **Função Alpine 1** (alpine1)
Definição: $A1(x) = \sum_{i=1}^{n} |x_i \sin(x_i) + 0.1x_i|$
Domínio: $[-10, 10]^n$
Mínimo Global: $x^* = (0, ..., 0)$; $A1(x^*) = 0$

($A2$) **Função Alpine 2** (alpine2)
Definição: $A2(x) = \prod_{i=1}^{n} \sqrt{x_i} \sin(x_i)$
Domínio: $[0, 10]^n$
Mínimo Global: $x^* = (7.917...7.917)$; $A2(x^*) = 2.808^n$

(BA) **Função Bartels Conn** (bartels)
Definição: $BA(x) = |x_1^2 + x_2^2 + x_1x_2| + |\sin(x_1)| + |\cos(x_2)|$
Domínio: $[-500, 500]^2$
Mínimo Global: $x^* = (0, 0)$; $BA(x^*) = 1$

(BE) **Função Beale** (beale)
Definição: $BE(x) = (1.5 - x_1 - x_1x_2)^2 + (2.25 - x_1 - x_1x_2^2)^2 + (2.625 - x_1 - x_1x_2^3)^2$
Domínio: $[-4.5, 4.5]^2$
Mínimo Global: $x^* = (3, \frac{1}{2})$; $BE(x^*) = 0$

(B_2) **Função Bohachevsky** (bohachevsky)
Definição: $B_2(x) = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1) - 0.4\cos(4\pi x_2) + 0,7$
Domínio: $[-50, 100]^2$
Mínimo Global: $x^* = (0, 0)$; $B_2(x^*) = 0$

(BO) **Função Booth** (booth)
Definição: $BO(x) = (x_1 + 2x_2 - 7)^2 + (2x_1 + x_2 - 5)^2$
Domínio: $[-10, 10]^2$
Mínimo Global: $x^* = (1, 3)$; $BO(x^*) = 0$

(BR) **Função Branin** (branin)

Definição: $BR(x) = (x_2 - \frac{5}{4\pi^2}x_1^2 + \frac{1}{\pi}5x_1 - 6)^2 10(1 - \frac{1}{8\pi})\cos(x_1) + 10$
Domínio: $[-5, 15]^2$
Mínimo Global (um de vários): $x^* = (\pi, 2.275)$; $BR(x^*) = 0.397887$

(CV) **Função Colville** (colville)
Definição: $CV(x) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2 + 90(x_4 - x_3^2)^2 + (1 - x_3)^2 +$
$10.1\left[(x_2 - 1)^2 + (x_4 - 1)^2\right] + 19.8(x_2 - 1)(x_4 - 1)$
Domínio: $[-10, 10]^4$
Mínimo Global: $x^* = (1, 1, 1, 1)$; $CV(x^*) = 0$

(EA) **Função Easom** (easom)
Definição: $EA(x) = -\cos(x_1)\cos(x_2)e^{-(x_1-\pi)^2-(x_2-\pi)^2}$
Domínio: $[-100, 100]^2$
Mínimo Global: $x^* = (\pi, \pi)$; $EA(x^*) = -1$

(GP) **Função Goldstein e Price** (goldstein)
Definição: $GP(x) = \left[1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)\right]$
$[30 + (2x_1 - 3x_2)^2(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2]$
Domínio: $[-2, 2]^2$
Mínimo Global: $x^* = (0, -1)$; $GP(x^*) = 3$

(GR_n) **Função Griewank** (griewank)
Definição: $GR_n(x) = \sum_{i=1}^{n} \frac{x_i^2}{4000} - \prod_{i=1}^{n} \cos(\frac{x_i}{\sqrt{i}}) + 1$
Domínio: $[-300, 600]^n$
Mínimo Global: $x^* = (0, ..., 0)$; $GR_n(x^*) = 0$

(L_n) **Função Levy** (levy)
Definição: $L_n(x) = \sin^2(\pi y_1) + \sum_{i=1}^{n-1}\left[(y_i - 1)^2(1 + 10\sin^2(\pi y_i + 1))\right]$
$+(y_n - 1)^2(1 + 10\sin^2(2\pi y_n))$
Domínio: $[-10, 10]^n$
Mínimo Global: $x^* = (1, ..., 1)$; $L_n(x^*) = 0$
Parâmetros: $y_i = 1 + \frac{x_i - 1}{4}.\forall i = 1, ..., n$

(M) **Função Matyas** (matyas)
Definição: $M(x) = 0.26(x_1^2 + x_2^2) - 0.48x_1x_2$
Domínio: $[-5, 10]^2$
Mínimo Global: $x^* = (0, 0)$; $M(x^*) = 0$

($P_{n,\beta}$) **Função Perm** (perm)

Definição: $P_{n,\beta}(x) = \sum_{k=1}^{n} \left[\sum_{i=1}^{n} (i^k + \beta)((\frac{x_i}{i})^k - 1)\right]^2$
Domínio: $[-n, n]^n$
Mínimo Global: $x^* = (1, 2, ..., n)$; $P_{n,\beta}(x^*) = 0$

($P^0_{n,\beta}$) **Função Perm$_0$** (perm0)
Definição: $P^0_{n,\beta}(x) = \sum_{k=1}^{n} \left[\sum_{i=1}^{n} (i + \beta)((x_i^k - (\frac{1}{i})^k))\right]^2$
Domínio: $[-n, n]^n$
Mínimo Global: $x^* = (1, \frac{1}{2}, ..., \frac{1}{n})$; $P^0_{n,\beta}(x^*) = 0$

(PW_n) **Função Powell** (powell)
Definição: $PW_n(x) =$
$\sum_{i=1}^{\frac{n}{4}} \left[(x_{4i-3} + 10x_{4i-2})^2 + 5(x_{4i-1} - x_{4i})^2 + (x_{4i-2} - 2x_{4i-1})^4 + 10(x_{4i-3} - x_{4i})^4\right]$
Domínio: $[-4, 5]^n$
Mínimo Global: $x^* = (3, -1, 0, 1, 3, ..., 3, -1, 0, 1)$; $PW_n(x^*) = 0$

($PS_{n,b}$) **Função Power Sum** (powersum)
Definição: $PS_{n,b}(x) = \sum_{k=1}^{n} ((\sum_{i=1}^{n} x_i^k) - b_k)^2$
Domínio: $[0, n]^n$
Mínimo Global: $(n = 4, b = \{8, 18, 44, 114\}) : x^* = (1, 2, 2, 3)\,; PS_{4,\{8,18,44,114\}}(x^*) = 0$

(RA_n) **Função Rastrigin** (rastrigin)
Definição: $RA_n(x) = 10n + \sum_{i=1}^{n} (x_i^2 - 10\cos(2\pi x_i))$
Domínio: $[-2.56, 5.12]^n$
Mínimo Global: $x^* = (0, ..., 0)$; $RA_n(x^*) = 0$

(R_n) **Função Rosenbrock** (rosenbrock)
Definição: $R_n(x) = \sum_{j=1}^{n-1} \left[100(x_j^2 - x_{j+1})^2 + (x_j - 1)^2\right]$
Domínio: $[-10, 10]^n$
Mínimo Global: $x^* = (1, ..., 1)$; $R_n(x^*) = 0$

($S_{4,m}$) **Função Shekel** (shekel - shekel5, shekel7, shekel10)
Definição: $S_{4,m}(x) = -\sum_{i=1}^{m} \left[(x - a_i)^T (x - a_i) + c_i\right]^{-1}$
Domínio: $[0, 10]^4$
Mínimo Global: $x^* = (4, 4, 4, 4)$;
$S_{4,5}(x^*) = -10.15319538, S_{4,7}(x^*) = -10.40281868, \quad e \quad S_{4,10}(x^*) = -10.53628349$
Parâmetros:

$$a = \begin{bmatrix} 4.0 & 4.0 & 4.0 & 4.0 \\ 1.0 & 1.0 & 1.0 & 1.0 \\ 8.0 & 8.0 & 8.0 & 8.0 \\ 6.0 & 6.0 & 6.0 & 6.0 \\ 7.0 & 3.0 & 7.0 & 3.0 \\ 2.0 & 9.0 & 2.0 & 9.0 \\ 5.0 & 5.0 & 3.0 & 3.0 \\ 8.0 & 1.0 & 8.0 & 1.0 \\ 6.0 & 2.0 & 6.0 & 2.0 \\ 7.0 & 2.6 & 7.0 & 3.6 \end{bmatrix}$$

$c = [0.1, 0.2, 0.2, 0.4, 0.4, 0.6, 0.3, 0.7, 0.5, 0.5]$

(SH) **Função Shubert** (shubert)
Definição: $SH(x) = \left[\sum_{i=1}^{5} i \cos\left[(i+1)\, x_1 + i\right]\right] \left[\sum_{i=1}^{5} i \cos\left[(i+1)\, x_2 + 1\right]\right]$
Domínio: $[-10, 10]^2$
Mínimo Global (um de vários): $x^* = (5.48242188, 4.85742188)$; $SH(x^*) = -186.7309$

(CA) **Função Six-Hump CamelBack** (sixhump)
Definição: $CA(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$
Domínio: $[-5, 5]^2$
Mínimo Global (um de vários): $x^* = (0.08984375, -0.71289062)$; $CA(x^*) = -1.03162801$

(SP_n) **Função Sphere** (sphere)
Definição: $SP_n(x) = \sum_{i=1}^{n} x_i^2$
Domínio: $[-2.56, 5.12]^n$
Mínimo Global: $x^* = (0, ..., 0)$; $SP_n(x^*) = 0$

(SS_n) **Função Sum of Squares** (sumsquares)
Definição: $SS_n(x) = \sum_{i=1}^{n} ix_i^2$
Domínio: $[-5, 10]^n$
Mínimo Global: $x^* = (0, ..., 0)$; $SS_n(x^*) = 0$

(T_n) **Função Trid** (trid)
Definição: $T_n(x) = \sum_{i=1}^{n} (x_i - 1)^2 - \sum_{i=2}^{n} x_i x_{i-1}$
Domínio: $[-n^2, n^2]^n$
Mínimo Global: $x_i^* = i\,(n + 1 - i)\,, \forall i = 1, ..., n; T_6(x^*) = -50 \quad e \quad T_{10}(x^*) = -210$

(Z_n) **Função Zakharov** (zakharov)
Definição: $Z_n(x) = \sum_{i=1}^{n} x_i^2 + (\sum_{i=1}^{n} 0.5ix_i)^2 + (\sum_{i=1}^{n} 0.5ix_i)^4$
Domínio: $[-5, 10]^n$
Mínimo Global: $x^* = (0, ..., 0)$; $Z_n(x^*) = 0$

APÊNDICE B – ARQUIVOS DE CONFIGURAÇÃO DO IRACE

B.1 EXEMPLOS DE ARQUIVOS DE CONFIGURAÇÃO PARA O AJUSTE DOS PARÂMETROS DA DISTRIBUIÇÃO DE LEVY

- Parâmetros

```
# name          switch          type    values
alfa            "--alfa "       r       (0, 2)
beta            "--beta "       r       (-1, 1)
loc             "--loc "        i       (-15, 30)
scale           "--scale "      r       (0, 45)
```

Figura 20 – **Exemplo de arquivo de parâmetros do irace utilizado no ajuste dos parâmetros de Levy para a função Ackley.** Este arquivo deve conter as informações acerca dos parâmetros do algoritmo a serem ajustados pelo irace. Na primeira coluna são registrados os nomes dos parâmetros, a segunda coluna deve conter a diretiva utilizada para indicar o parâmetro na linha de comando do algoritmo, caso exista. A terceira coluna deve indicar os tipos de cada parâmetro, e a quarta coluna contém o domínio ao qual pertence o parâmetro. Existe uma quinta coluna que é opcional e determina as condições de existência de cada parâmetro, e deve-se utilizar a sintaxe da linguagem de programação R para descrevê-las.

- Restrições

```
alfa == 0
```

Figura 21 – **Exemplo de arquivo de restrições do irace.** Neste arquivo, as restrições dos parâmetros devem ser escritas, uma por linha, utilizando a sintaxe da linguagem de programação R, que devem descrever situações que não podem acontecer. Por exemplo, neste caso das configurações dos parâmetros da distribuição de Levy, o parâmetro $alfa$ não poderia receber o valor 0, portanto, esta restrição foi registrada no arquivo.

- Instâncias

```
2
5
7
10
```

Figura 22 – **Exemplo de arquivo de instâncias do irace para a função Ackley.** O arquivo de instâncias deve ser organizado com uma instância por linha. No caso da configuração dos parâmetros de Levy para cada função, foram utilizadas como instâncias as dimensões da função que seriam testadas.

- Função de Avaliação

```
#!/bin/bash

EXE=python
FIXED_PARAMS=../../main.py
FUN=ackley

CONFIG_ID=$1
INSTANCE_ID=$2
SEED=$3
INSTANCE=$4
shift 4 || exit 1
CONFIG_PARAMS=$*

STDOUT=c${CONFIG_ID}-${INSTANCE_ID}.stdout
STDERR=c${CONFIG_ID}-${INSTANCE_ID}.stderr

$EXE ${FIXED_PARAMS} --fun ${FUN} --seed ${SEED} --n $INSTANCE ${CONFIG_PARAMS} 1> ${STDOUT} 2> ${STDERR}

error() {
    echo "`TZ=UTC date`: error: $@"
    exit 1
}

if [ -s "${STDOUT}" ]; then
    COST=$(tail -n 1 ${STDOUT} | grep -e '^[[:space:]]*[+-]\?[0-9]' | cut -f1)
    echo "$COST"
    rm -f "${STDOUT}" "${STDERR}"
    exit 0
else
    error "${STDOUT}: No such file or directory"
fi
```

Figura 23 – **Exemplo de arquivo de função de avaliação do irace para a função Ackley.** Neste arquivo deve ser criado um script executável para calcular o custo de cada configuração candidata.

- Cenário

```
parameterFile = "./parameters.txt"

execDir = "./exec-dir/"

logFile = "./irace.Rdata"

trainInstancesDir = ""

trainInstancesFile = "instances.txt"

targetRunner = "./target-runner"

targetRunnerRetries = 5

maxExperiments = 300

digits = 4

forbiddenFile = "../forbidden.txt"
```

Figura 24 – **Exemplo de arquivo de cenário do irace.** Este arquivo deve relacionar todos os parâmetros do irace aos seus arquivos de configuração criados, além de outros detalhes da execução, como os limites computacionais de execução (neste caso foi utilizada a quantidade máxima de experimentos), quantidade de tentativas caso aconteça algum problema em uma execução, quantidade de dígitos dos números decimais, entre outros.

B.2 EXEMPLOS DE ARQUIVOS DE CONFIGURAÇÃO PARA O AJUSTE DOS PARÂMETROS DO BRKGA

- Parâmetros

```
# name          switch          type    values
p               "--p "          i       (10, 100)
pe              "--pe "         r       (0, 1)
pm              "--pm "         r       (0, 1)
rho             "--rho  "       r       (0. 1)
MAX_GENS        "--maxgen "     i       (10, 500)
n               "--n "          i       (5, 10)
```

Figura 25 – **Exemplo de arquivo de parâmetros do irace para o ajuste das configurações da versão tradicional do BRKGA.** Onde devem ser informados os nomes dos parâmetros, as diretivas de identificação do parâmetro utilizadas na linha de comando do algoritmo, o tipo dos parâmetros e o domínio a qual eles pertencem. Parâmetros que não serão ajustados devem ser descritos com tipo "c", que corresponde a "categórico"e na coluna do domínio deve-se informar o valor fixo que este parâmetro deverá receber nas execuções do algoritmo.

- Restrições

```
pe+pm > 1
pe*p<1
```

Figura 26 – **Exemplo de arquivo de restrições do irace para os parâmetros do BRKGA tradicional.**

- Instâncias

```
ackley1
alpine1
alpine2
brown
pathological
```

Figura 27 – **Exemplo de arquivo de instâncias do irace para o ajuste dos parâmetros do BRKGA tradicional.** Neste caso, as instâncias utilizadas para o ajuste dos parâmetros foram as funções de benchmark.

- Função de Avaliação

```
#!/bin/bash

EXE=python
FIXED_PARAMS=../../main.py

CONFIG_ID=$1
INSTANCE_ID=$2
SEED=$3
INSTANCE=$4
shift 4 || exit 1
CONFIG_PARAMS=$*

STDOUT=c${CONFIG_ID}-${INSTANCE_ID}.stdout
STDERR=c${CONFIG_ID}-${INSTANCE_ID}.stderr

$EXE ${FIXED_PARAMS} --seed ${SEED} --fun $INSTANCE ${CONFIG_PARAMS} 1> ${STDOUT} 2> ${STDERR}

error() {
    echo "`TZ=UTC date`: error: $@"
    exit 1
}

if [ -s "${STDOUT}" ]; then
    COST=$(tail -n 1 ${STDOUT} | grep -e '^[[:space:]]*[+-]\?[0-9]' | cut -f1)
    echo "$COST"
    rm -f "${STDOUT}" "${STDERR}"
    exit 0
else
    error "${STDOUT}: No such file or directory"
fi
```

Figura 28 – **Exemplo de arquivo de função de avaliação do irace para a configuração de parâmetros do BRKGA tradicional.**

- Cenário

```
parameterFile = "./parameters.txt"

execDir = "./exec-dir/"

logFile = "./irace-brkga1.Rdata"

trainInstancesDir = ""

trainInstancesFile = "instances.txt"

targetRunner = "./target-runner"

targetRunnerRetries = 5

forbiddenFile = "forbidden.txt"
```

Figura 29 – **Exemplo de arquivo de cenário do irace para a configuração de parâmetros do BRKGA tradicional.**

APÊNDICE C – RESULTADOS DETALHADOS

Tabela 14 – Resultados das funções para n=2

Funções	**n**	**Resultados**	**Métodos - Valor da Solução**			
			BRKGA	BRKGA-Levy	BRKGA-LS	BRKGA-Levy-LS
ackley	2	Mín	0.00033	0.00000	0.00022	0.00000
		Máx	0.04593	0.00085	0.00099	0.00055
		Méd	0.01310	0.00004	0.00063	0.00007
		DP	0.01203	0.00016	0.00026	0.00016
		%Qual	3.33	**100.00**	**100.00**	**100.00**
alpine1	2	Mín	0.00037	0.00000	0.00001	0.00000
		Máx	0.00099	0.00000	0.00033	0.00000
		Méd	0.00076	0.00000	0.00017	0.00000
		DP	0.00018	0.00000	0.00008	0.00000
		%Qual	**100.00**	**100.00**	**100.00**	**100.00**
alpine2	2	Mín	-7.88510	-7.88533	-7.88560	-7.88560
		Máx	-7.87772	-7.87777	-7.88556	-7.88559
		Méd	-7.88127	-7.88239	-7.88559	-7.88560
		DP	0.00229	0.00227	0.00001	0.00000
		%Qual	**100.00**	**100.00**	**100.00**	**100.00**
bartels	2	Mín	1.00072	1.00251	1.00000	1.00004
		Máx	1.34335	1.12152	1.00008	1.00093
		Méd	1.05594	1.04077	1.00001	1.00034
		DP	0.06459	0.03451	0.00002	0.00026
		%Qual	0.00	0.00	36.67	**100.00**
beale	2	Mín	0.00002	0.00001	0.00000	0.00000
		Máx	0.00097	0.00091	0.00001	0.00016
		Méd	0.00053	0.00048	0.00000	0.00005
		DP	0.00027	0.00028	0.00000	0.00005
		%Qual	**100.00**	**100.00**	**100.00**	**100.00**
bohachevsky	2	Mín	0.00019	0.00034	0.00000	0.00000
		Máx	0.08286	0.06370	0.00007	0.00029
		Méd	0.00745	0.01293	0.00002	0.00010
		DP	0.01507	0.01602	0.00002	0.00008
		%Qual	40.00	20.00	**100.00**	**100.00**
booth	2	Mín	0.00008	0.00003	0.00000	0.00000
		Máx	0.00232	0.00152	0.00001	0.00001
		Méd	0.00062	0.00055	0.00000	0.00000
		DP	0.00043	0.00034	0.00000	0.00000
		%Qual	96.67	90.00	**100.00**	**100.00**
branin	2	Mín	0.39789	0.39789	0.39789	0.39789
		Máx	0.39891	0.39885	0.39789	0.39789
		Méd	0.39834	0.39836	0.39789	0.39789
		DP	0.00033	0.00029	0.00000	0.00000
		%Qual	73.33	80.00	**100.00**	**100.00**
easom	2	Mín	-0.99996	-0.99999	-1.00000	-1.00000

Continua na próxima página

Funções	**n**	**Resultados**	**Métodos - Valor da Solução**			
			BRKGA	BRKGA-Levy	BRKGA-LS	BRKGA-Levy-LS
		Máx	-0.95261	-0.99696	-0.99991	-0.99993
		Méd	-0.99744	-0.99934	-0.99998	-0.99998
		DP	0.00851	0.00061	0.00002	0.00002
		%Qual	80.00	86.67	**100.00**	**100.00**
goldstein	2	Mín	3.00009	3.00014	3.00001	3.00000
		Máx	3.00261	3.00285	3.00134	3.00060
		Méd	3.00122	3.00125	3.00028	3.00019
		DP	0.00078	0.00084	0.00028	0.00016
		%Qual	**100.00**	**100.00**	**100.00**	**100.00**
griewank	2	Mín	0.00019	0.00002	0.00000	0.00000
		Máx	0.01180	0.00740	0.00007	0.00005
		Méd	0.00614	0.00128	0.00003	0.00001
		DP	0.00352	0.00206	0.00002	0.00001
		%Qual	10.00	90.00	**100.00**	**100.00**
levy	2	Mín	0.00005	0.00000	0.00000	0.00000
		Máx	0.00100	0.00097	0.00000	0.00000
		Méd	0.00039	0.00038	0.00000	0.00000
		DP	0.00033	0.00027	0.00000	0.00000
		%Qual	**100.00**	**100.00**	**100.00**	**100.00**
matyas	2	Mín	0.00002	0.00000	0.00000	0.00000
		Máx	0.00096	0.00000	0.00000	0.00000
		Méd	0.00048	0.00000	0.00000	0.00000
		DP	0.00026	0.00000	0.00000	0.00000
		%Qual	**100.00**	**100.00**	**100.00**	**100.00**
perm	2	Mín	0.00001	0.00002	0.00000	0.00000
		Máx	0.00098	0.00096	0.00002	0.00010
		Méd	0.00060	0.00046	0.00001	0.00003
		DP	0.00026	0.00028	0.00001	0.00003
		%Qual	**100.00**	**100.00**	**100.00**	**100.00**
perm0	2	Mín	0.00006	0.00013	0.00000	0.00000
		Máx	0.00094	0.00097	0.00000	0.00002
		Méd	0.00046	0.00055	0.00000	0.00000
		DP	0.00027	0.00030	0.00000	0.00000
		%Qual	**100.00**	**100.00**	**100.00**	**100.00**
powell	2	Mín	0.00000	0.00000	0.00000	0.00000
		Máx	0.00000	0.00000	0.00000	0.00000
		Méd	0.00000	0.00000	0.00000	0.00000
		DP	0.00000	0.00000	0.00000	0.00000
		%Qual	**100.00**	**100.00**	**100.00**	**100.00**
rastrigin	2	Mín	0.00005	0.00001	0.00005	0.00001
		Máx	0.00155	0.00093	0.00099	0.00030
		Méd	0.00052	0.00035	0.00055	0.00009
		DP	0.00035	0.00028	0.00031	0.00008
		%Qual	96.67	**100.00**	**100.00**	**100.00**
rosenbrock	2	Mín	0.00011	0.00004	0.00000	0.00001
		Máx	0.03434	0.01551	0.00004	0.00080

Continua na próxima página

Funções	**n**	**Resultados**	**Métodos - Valor da Solução**			
			BRKGA	BRKGA-Levy	BRKGA-LS	BRKGA-Levy-LS
		Méd	0.00491	0.00194	0.00001	0.00027
		DP	0.00744	0.00310	0.00001	0.00024
		%Qual	30.00	56.67	**100.00**	**100.00**
shubert	2	Mín	-186.72974	-186.72252	-186.73091	-186.73088
		Máx	-186.54330	-186.54593	-186.71885	-186.71655
		Méd	-186.62101	-186.62691	-186.72769	-186.72759
		DP	0.05388	0.05452	0.00318	0.00359
		%Qual	**100.00**	**100.00**	**100.00**	**100.00**
sixhump	2	Mín	-1.03161	-1.03162	-1.03163	-1.03163
		Máx	-1.03063	-1.03060	-1.03161	-1.03162
		Méd	-1.03115	-1.03107	-1.03162	-1.03163
		DP	0.00031	0.00033	0.00001	0.00000
		%Qual	**100.00**	**100.00**	**100.00**	**100.00**
sphere	2	Mín	0.00003	0.00000	0.00000	0.00000
		Máx	0.00096	0.00000	0.00000	0.00000
		Méd	0.00046	0.00000	0.00000	0.00000
		DP	0.00029	0.00000	0.00000	0.00000
		%Qual	**100.00**	**100.00**	**100.00**	**100.00**
sumsquares	2	Mín	0.00000	0.00001	0.00000	0.00000
		Máx	0.00097	0.00100	0.00001	0.00000
		Méd	0.00042	0.00047	0.00000	0.00000
		DP	0.00029	0.00027	0.00000	0.00000
		%Qual	**100.00**	**100.00**	**100.00**	**100.00**
trid	2	Mín	2.00013	-1.99999	6.93952	-2.00000
		Máx	2.00804	-1.99805	6.95287	-2.00000
		Méd	2.00206	-1.99902	6.94870	-2.00000
		DP	0.00175	0.00058	0.00283	0.00000
		%Qual	0.00	**100.00**	0.00	**100.00**
zakharov	2	Mín	0.00004	0.00003	0.00468	0.00000
		Máx	0.00099	0.00096	4.92933	0.00000
		Méd	0.00053	0.00052	0.76068	0.00000
		DP	0.00025	0.00026	1.10414	0.00000
		%Qual	**100.00**	**100.00**	0.00	**100.00**

Tabela 15 – Resultados das funções para n=4

Funções	**n**	**Resultados**	**Métodos - Valor da Solução**			
			BRKGA	BRKGA-Levy	BRKGA-LS	BRKGA-Levy-LS
ackley	4	Mín	0.00681	0.00000	0.00061	0.00000
		Máx	0.59433	0.00096	0.00100	0.00100
		Méd	0.09704	0.00029	0.00082	0.00069
		DP	0.11040	0.00034	0.00012	0.00034
		%Qual	0.00	**100.00**	96.67	**100.00**
alpine1	4	Mín	0.00050	0.00000	0.00035	0.00000
		Máx	0.00337	0.00030	0.00099	0.00082
		Méd	0.00136	0.00002	0.00078	0.00046
		DP	0.00075	0.00006	0.00017	0.00036
		%Qual	50.00	**100.00**	**100.00**	**100.00**
alpine2	4	Mín	-62.17049	-62.16739	-62.18230	-62.18227
		Máx	-62.12075	-62.12188	-62.17770	-62.17452
		Méd	-62.13987	-62.14056	-62.18041	-62.18088
		DP	0.01317	0.01222	0.00128	0.00144
		%Qual	**100.00**	**100.00**	**100.00**	**100.00**
colville	4	Mín	0.00550	0.00075	0.00009	0.00015
		Máx	1.06308	0.21083	0.00100	0.00100
		Méd	0.39462	0.05113	0.00063	0.00059
		DP	0.34316	0.05342	0.00026	0.00020
		%Qual	0.00	3.33	**100.00**	**100.00**
griewank	4	Mín	0.01568	0.00014	0.00000	0.00000
		Máx	0.21812	0.01763	0.00007	0.00096
		Méd	0.06414	0.00737	0.00003	0.00016
		DP	0.05040	0.00474	0.00002	0.00025
		%Qual	0.00	23.33	**100.00**	**100.00**
levy	4	Mín	0.00022	0.00013	0.00001	0.00001
		Máx	0.00200	0.00100	0.00013	0.00075
		Méd	0.00071	0.00065	0.00005	0.00013
		DP	0.00037	0.00027	0.00002	0.00016
		%Qual	93.33	**100.00**	**100.00**	**100.00**
perm	4	Mín	0.01893	0.00298	0.00015	0.00025
		Máx	1.06744	0.28209	0.00100	0.00491
		Méd	0.24689	0.04869	0.00069	0.00106
		DP	0.23236	0.06701	0.00028	0.00106
		%Qual	0.00	0.00	**100.00**	86.67
perm0	4	Mín	0.00049	0.00010	0.00001	0.00013
		Máx	0.06390	0.00450	0.00035	0.00096
		Méd	0.01048	0.00146	0.00008	0.00043
		DP	0.01372	0.00112	0.00007	0.00024
		%Qual	20.00	56.67	**100.00**	**100.00**
powell	4	Mín	0.00025	0.00005	0.00000	0.00003
		Máx	0.01055	0.00100	0.00017	0.00085
		Méd	0.00285	0.00080	0.00004	0.00037
		DP	0.00266	0.00022	0.00003	0.00023
		%Qual	43.33	**100.00**	**100.00**	**100.00**

Continua na próxima página

Funções	**n**	**Resultados**	**Métodos - Valor da Solução**			
			BRKGA	BRKGA-Levy	BRKGA-LS	BRKGA-Levy-LS
powersum	4	Mín	0.00037	0.00066	0.00003	0.00011
		Máx	0.04706	0.02307	0.00099	0.00097
		Méd	0.01061	0.00754	0.00050	0.00062
		DP	0.01215	0.00628	0.00027	0.00022
		%Qual	6.67	10.00	**100.00**	**100.00**
rastrigin	4	Mín	0.00004	0.00002	0.00023	0.00015
		Máx	0.08571	0.00099	0.00098	0.00098
		Méd	0.00954	0.00060	0.00063	0.00060
		DP	0.01725	0.00025	0.00024	0.00020
		%Qual	26.67	**100.00**	**100.00**	
rosenbrock	4	Mín	0.01077	0.00754	0.00026	0.00007
		Máx	1.77087	1.81750	0.00098	0.00100
		Méd	0.61768	0.56966	0.00065	0.00065
		DP	0.47181	0.53501	0.00022	0.00024
		%Qual	0.00	0.00	**100.00**	**100.00**
shekel5	4	Mín	-10.15249	-10.15300	-10.15295	-10.15307
		Máx	-5.05518	-10.14341	-10.14804	-10.15123
		Méd	-9.80899	-10.14759	-10.15120	-10.15240
		DP	1.26444	0.00270	0.00110	0.00047
		%Qual	93.33	**100.00**	**100.00**	**100.00**
shekel7	4	Mín	-10.40221	-10.40180	-10.40241	-10.40275
		Máx	-5.08763	-10.39246	-10.39767	-10.40099
		Méd	-10.21963	-10.39622	-10.40072	-10.40195
		DP	0.95299	0.00296	0.00117	0.00041
		%Qual	96.67	**100.00**	**100.00**	**100.00**
shekel10	4	Mín	-10.53173	-10.52904	-10.53085	-10.53172
		Máx	-10.51966	-10.52145	-10.52555	-10.52995
		Méd	-10.52534	-10.52503	-10.52904	-10.53077
		DP	0.00282	0.00248	0.00139	0.00052
		%Qual	96.67	**100.00**	**100.00**	**100.00**
sphere	4	Mín	0.00016	0.00000	0.00001	0.00000
		Máx	0.00100	0.00000	0.00003	0.00002
		Méd	0.00059	0.00000	0.00002	0.00000
		DP	0.00028	0.00000	0.00001	0.00000
		%Qual	**100.00**	**100.00**	**100.00**	**100.00**
sumsquares	4	Mín	0.00002	0.00008	0.00001	0.00000
		Máx	0.00508	0.00095	0.00007	0.00007
		Méd	0.00101	0.00059	0.00004	0.00003
		DP	0.00100	0.00024	0.00002	0.00002
		%Qual	80.00	**100.00**	**100.00**	**100.00**
trid	4	Mín	4.00727	-15.99574	84.69425	-16.00000
		Máx	4.21537	-15.98405	84.79821	-15.99993
		Méd	4.08117	-15.98822	84.75711	-15.99997
		DP	0.05689	0.00319	0.02509	0.00002
		%Qual	0.00	**100.00**	0.00	**100.00**
zakharov	4	Mín	0.00037	0.00009	267.01783	0.00001

Continua na próxima página

Funções	**n**	**Resultados**	**Métodos - Valor da Solução**			
			BRKGA	BRKGA-Levy	BRKGA-LS	BRKGA-Levy-LS
		Máx	0.08524	0.00100	400.20473	0.00008
		Méd	0.01161	0.00074	351.01688	0.00003
		DP	0.02065	0.00023	24.56349	0.00002
		%Qual	16.67	**100.00**	0.00	**100.00**

Tabela 16 – Resultados das funções para n=10

Funções	**n**	**Resultados**	**Métodos - Valor da Solução**			
			BRKGA	BRKGA-Levy	BRKGA-LS	BRKGA-Levy-LS
ackley	10	Mín	0.11435	0.00000	0.00096	0.00079
		Máx	2.67060	0.00095	0.00361	0.00282
		Méd	0.72499	0.00029	0.00224	0.00132
		DP	0.63345	0.00030	0.00077	0.00050
		%Qual	0.00	**100.00**	10.00	20.00
alpine2	10	Mín	-30482.82633	-30476.43280	-30488.26868	-30485.72925
		Máx	-29968.50655	-30460.85242	-30468.03187	-30460.79622
		Méd	-30373.77322	-30466.19482	-30480.27478	-30469.90065
		DP	143.50496	4.54308	6.90577	5.98063
		%Qual	40.00	**100.00**	**100.00**	**100.00**
griewank	10	Mín	0.13827	0.00148	0.00000	0.00986
		Máx	1.05123	0.07211	0.01232	0.08367
		Méd	0.63222	0.02632	0.00212	0.04069
		DP	0.27906	0.01566	0.00402	0.02035
		%Qual	0.00	0.00	**80.00**	0.00
levy	10	Mín	0.00071	0.00042	0.00008	0.00006
		Máx	0.09331	0.00197	0.00016	0.00096
		Méd	0.01498	0.00083	0.00012	0.00033
		DP	0.02054	0.00027	0.00002	0.00022
		%Qual	13.33	93.33	**100.00**	**100.00**
perm	10	Mín	9.066e+10	1.503e+10	5.474e+07	1.083e+07
		Máx	3.424e+13	3.126e+12	8.446e+10	3.316e+10
		Méd	3.955e+12	4.464e+11	1.61e+10	5.523e+09
		DP	6.954e+12	6.518e+11	2.612+10	8.259e+09
		%Qual	0.00	0.00	0.00	0.00
perm0	10	Mín	0.00214	0.00102	0.00033	0.00030
		Máx	14.32278	2.08934	0.00099	0.00099
		Méd	3.11288	0.17575	0.00064	0.00067
		DP	3.35317	0.37084	0.00020	0.00021
		%Qual	0.00	0.00	**100.00**	**100.00**
powell	10	Mín	0.00547	0.00093	0.00043	0.00019
		Máx	0.24116	0.02387	0.00095	0.00100
		Méd	0.07579	0.00520	0.00064	0.00076
		DP	0.06885	0.00576	0.00017	0.00021
		%Qual	0.00	10.00	**100.00**	**100.00**
rastrigin	10	Mín	0.01245	0.00037	0.00067	0.00061
		Máx	2.62970	0.00098	0.00099	0.00099
		Méd	0.77385	0.00078	0.00088	0.00087
		DP	0.79294	0.00016	0.00011	0.00010
		%Qual	0.00	**100.00**	**100.00**	**100.00**
rosenbrock	10	Mín	0.98979	0.34747	0.00213	0.01297
		Máx	62.73592	8.39747	0.01465	0.30394
		Méd	21.05119	5.67609	0.00761	0.16700
		DP	19.97842	2.56608	0.00372	0.06992
		%Qual	0.00	0.00	0.00	0.00

Continua na próxima página

Funções	**n**	**Resultados**	**Métodos - Valor da Solução**			
			BRKGA	BRKGA-Levy	BRKGA-LS	BRKGA-Levy-LS
sphere	10	Mín	0.00040	0.00000	0.00008	0.00023
		Máx	0.03726	0.00093	0.00020	0.00098
		Méd	0.00386	0.00013	0.00014	0.00062
		DP	0.00680	0.00024	0.00004	0.00021
		%Qual	33.33	**100.00**	**100.00**	**100.00**
sumsquares	10	Mín	0.00142	0.00061	0.00081	0.00052
		Máx	0.32001	0.02404	0.00100	0.00096
		Méd	0.07002	0.00274	0.00090	0.00077
		DP	0.08757	0.00461	0.00005	0.00013
		%Qual	0.00	43.33	**100.00**	**100.00**
trid	10	Mín	11.05613	-209.81871	2845.82802	-209.97797
		Máx	62.95359	-196.05364	2846.49703	-209.79125
		Méd	21.87275	-206.14956	2846.18048	-209.81793
		DP	10.62007	2.99100	0.20547	0.04203
		%Qual	0.00	3.33	0.00	**100.00**
zakharov	10	Mín	167.80842	0.00078	731579.35812	0.00041
		Máx	3158.74413	0.00542	2306004.83203	0.00099
		Méd	1302.18217	0.00145	1494283.59178	0.00075
		DP	787.29228	0.00112	417713.34209	0.00015
		%Qual	0.00	60.00	0.00	**100.00**

APÊNDICE D – TEMPOS COMPUTACIONAIS DETALHADOS

Tabela 17 – Tempo computacional das execuções para n=2

Funções	**n**	**Resultados**	**Métodos - tempo(s)**			
			BRKGA	BRKGA-Levy	BRKGA-LS	BRKGA-Levy-LS
ackley	2	Mín	183.32753	2.15500	7.33851	0.82968
		Máx	1658.01611	8.59255	492.78440	2.72454
		Méd	584.15224	5.22667	91.95146	**1.02415**
		DP	319.46938	1.77217	105.57695	0.34791
alpine1	2	Mín	5.50223	1.95933	2.66174	0.89718
		Máx	192.08094	8.46161	15.05902	1.08226
		Méd	66.18242	5.03741	5.04727	**0.97143**
		DP	45.90920	1.84129	2.42908	0.04386
alpine2	2	Mín	0.66446	2.29554	1.98526	0.81478
		Máx	50.09830	12.50328	11.85042	0.95706
		Méd	10.87388	6.07695	4.40744	**0.89344**
		DP	11.29532	2.30140	1.98610	0.04679
griewank	2	Mín	169.74726	22.15597	15.29420	0.89930
		Máx	1661.03121	3759.50816	871.21644	3.32117
		Méd	582.84195	620.03321	148.49215	**1.45896**
		DP	283.70951	710.06758	174.26732	0.66721
levy	2	Mín	2.02489	4.63584	3.00436	1.03442
		Máx	71.55185	108.55801	15.92555	1.23464
		Méd	18.26725	23.12165	5.87431	**1.11734**
		DP	14.08305	22.18556	2.58511	0.05222
perm	2	Mín	0.59810	2.21254	2.29857	1.13713
		Máx	65.15052	47.61730	14.16876	1.28316
		Méd	18.21289	12.80549	5.77045	**1.23553**
		DP	15.10965	11.13394	2.16094	0.04100
perm0	2	Mín	1.20207	2.62101	2.57778	1.06154
		Máx	139.70534	83.30617	15.24805	1.19937
		Méd	21.72110	17.86671	5.75862	**1.14228**
		DP	28.12617	17.94281	2.59354	0.05245
powell	2	Mín	0.45073	1.87773	2.17153	0.71927
		Máx	1.89796	9.68388	8.04455	0.87301
		Méd	1.29936	4.99927	3.48313	**0.79720**
		DP	0.32757	2.16882	1.12566	0.04434
rastrigin	2	Mín	18.76229	1.84616	1.90953	0.80733
		Máx	337.62626	15.49049	62.32523	0.98188
		Méd	143.46103	5.48943	16.26001	**0.88985**
		DP	79.16985	3.17806	14.76315	0.04746
rosenbrock	2	Mín	58.39392	17.10510	3.17805	0.95289
		Máx	804.81791	3064.85563	16.19831	1.68230
		Méd	394.15163	971.83060	6.63678	**1.20049**
		DP	177.97581	719.24503	2.74570	0.21208
sphere	2	Mín	0.51061	2.04189	2.06411	0.82609

Continua na próxima página

Funções	**n**	**Resultados**	**Métodos - tempo(s)**			
			BRKGA	BRKGA-Levy	BRKGA-LS	BRKGA-Levy-LS
		Máx	38.07353	10.17058	13.70939	0.96218
		Méd	12.61429	5.11159	4.69206	**0.89181**
		DP	10.01582	1.96409	2.13346	0.04537
sumsquares	2	Mín	4.24444	6.53620	1.67053	0.90712
		Máx	116.46090	102.00075	13.73918	1.06285
		Méd	31.33502	39.89692	4.67233	**1.00555**
		DP	24.66071	28.41520	2.27369	0.04166
trid	2	Mín	128.67873	3.29587	1434.00388	0.93869
		Máx	1251.05237	59.59026	14460.89387	1.13749
		Méd	617.22717	18.65807	4299.75832	**1.04161**
		DP	302.77647	15.85051	2816.75068	0.05416
zakharov	2	Mín	2.60565	2.47507	1556.99674	0.92127
		Máx	52.21952	16.69880	11655.83802	1.07497
		Méd	15.74423	7.08174	4239.48228	**1.00738**
		DP	12.76691	3.36557	2289.84396	0.04374
beale	2	Mín	5.49279	3.78276	2.29167	1.03582
		Máx	182.66112	213.96162	14.23900	1.18931
		Méd	56.03516	62.62697	5.73391	**1.11806**
		DP	49.63001	60.11953	2.79117	0.04241
bartels	2	Mín	325.18560	451.57338	8.01687	6.46999
		Máx	1827.27420	3994.10262	37442.10870	11.71190
		Méd	681.71678	1483.79355	11797.74196	**8.27753**
		DP	344.51093	744.89337	10955.95183	1.75644
bohachevsky	2	Mín	88.40482	418.13872	3.01262	2.47226
		Máx	826.64615	2992.50495	18.38690	2.99537
		Méd	452.47471	1194.51604	7.20453	**2.79167**
		DP	184.53724	645.80228	3.25386	0.12885
booth	2	Mín	10.81919	79.66400	2.04784	0.96618
		Máx	723.08461	2790.83770	10.90982	1.14644
		Méd	169.91833	736.78136	4.52028	**1.05554**
		DP	169.89605	596.43363	1.82783	0.04904
branin	2	Mín	1.88712	9.44471	1073.99305	138.26324
		Máx	1032.86562	1452.66379	12183.33369	491.51452
		Méd	**208.84466**	232.98375	3941.61751	253.29673
		DP	325.01386	394.96587	2501.06533	82.67079
easom	2	Mín	13.29218	63.38272	2.02487	0.88925
		Máx	735.13784	2489.29485	11.97140	1.91061
		Méd	353.00509	720.07268	4.59308	**1.25004**
		DP	175.40783	520.23706	1.99822	0.26284
goldstein	2	Mín	13.69633	17.37068	2.20568	1.06114
		Máx	216.27731	506.67162	12.98906	1.22903
		Méd	87.65588	185.41832	4.76407	**1.13869**
		DP	52.64882	138.58616	2.27625	0.04898
matyas	2	Mín	0.57014	2.32000	2.22679	1.21292
		Máx	45.49864	9.83176	16.19731	1.40465
		Méd	10.32113	5.18719	5.17782	**1.31335**

Continua na próxima página

Funções	**n**	**Resultados**	**Métodos - tempo(s)**			
			BRKGA	BRKGA-Levy	BRKGA-LS	BRKGA-Levy-LS
		DP	9.55620	2.06222	2.83885	0.05167
shubert	2	Mín	1.51856	15.24198	2.64906	1.02236
		Máx	124.50429	177.66579	13.64996	1.16892
		Méd	36.22368	68.17935	5.42578	**1.09215**
		DP	24.90751	38.47798	2.40978	0.05209
sixhump	2	Mín	1.44241	12.30445	2.16111	0.94343
		Máx	88.68272	200.55535	11.08909	1.08358
		Méd	34.26380	62.58211	4.78729	**1.01474**
		DP	23.99079	48.79184	1.93315	0.04484

Tabela 18 – Tempo computacional das execuções para n=4

Funções	**n**	**Resultados**	**Métodos - tempo(s)**			
			BRKGA	BRKGA-Levy	BRKGA-LS	BRKGA-Levy-LS
ackley	4	Mín	136.28587	1.48861	81.84383	1.33356
		Máx	510.35782	14.32886	2029.67926	138.61362
		Méd	296.19656	**5.14019**	776.44451	54.53875
		DP	84.90801	3.16078	495.63880	34.08967
alpine1	4	Mín	63.39961	1.64676	3.48100	1.31348
		Máx	341.54549	6.42247	38.39766	3.17571
		Méd	188.68225	3.12247	20.37956	**1.76964**
		DP	60.33515	1.31726	7.82026	0.60667
alpine2	4	Mín	8.37390	4.31154	3.31902	1.31955
		Máx	116.78633	29.26257	6.97942	1.49861
		Méd	42.88419	13.33686	4.47284	**1.39577**
		DP	22.07797	6.09806	1.09504	0.04007
griewank	4	Mín	118.34559	82.69010	206.58913	6.82716
		Máx	448.49145	2927.18629	2791.37501	88.34369
		Méd	264.36054	938.32213	561.90874	**21.20584**
		DP	90.62854	660.41857	479.72002	15.03463
levy	4	Mín	29.41171	20.06219	3.30504	1.90911
		Máx	172.70506	218.62149	10.86444	4.09165
		Méd	80.73727	84.85658	5.09484	**2.43427**
		DP	35.16846	42.72420	2.18352	0.74579
perm	4	Mín	143.52097	241.60335	32.79529	16.75714
		Máx	757.93421	4315.60985	4016.53187	2745.68583
		Méd	**336.44727**	1148.54103	1056.04022	538.00176
		DP	159.26038	890.74383	1436.59091	705.83887
perm0	4	Mín	28.92909	88.57738	3.97201	2.26688
		Máx	536.97904	5152.27293	7.04329	3.67659
		Méd	264.81200	1389.55830	5.15576	**2.41805**
		DP	129.88000	1260.60285	0.93249	0.24078
powell	4	Mín	23.82893	62.74084	2.85207	1.97591
		Máx	765.07230	1720.98194	5.92655	3.66212
		Méd	273.46422	616.35136	3.81305	**2.39334**
		DP	155.91781	426.70742	0.84813	0.46100
rastrigin	4	Mín	101.89646	3.30003	23.19169	1.17458
		Máx	412.14483	31.73345	267.61125	4.76506
		Méd	238.79294	12.79914	80.91849	**3.31695**
		DP	67.69784	5.90309	46.06130	1.07603
rosenbrock	4	Mín	146.67976	311.08144	14.90831	7.72363
		Máx	632.62109	9364.50028	135.08673	22.92326
		Méd	302.74415	1767.84192	61.64876	**14.19089**
		DP	139.76413	1886.28321	30.11297	3.73094
sphere	4	Mín	8.91593	1.32975	2.52742	1.33041
		Máx	80.03912	4.57483	4.67320	1.55985
		Méd	43.99671	2.28144	3.26678	**1.45112**
		DP	18.89508	0.87717	0.64071	0.05761
sumsquares	4	Mín	28.29111	51.34294	2.69877	1.66323

Continua na próxima página

Funções	**n**	**Resultados**	**Métodos - tempo(s)**			
			BRKGA	BRKGA-Levy	BRKGA-LS	BRKGA-Levy-LS
		Máx	343.34416	267.17601	5.69925	1.86733
		Méd	147.49668	135.33296	3.64569	**1.78282**
		DP	80.24241	60.18667	0.79744	0.05347
trid	4	Mín	191.06404	44.90358	1647.67433	2.94436
		Máx	595.98756	627.69091	7133.80680	3.32748
		Méd	271.97191	201.72987	3160.08860	**3.12683**
		DP	89.27260	125.54897	1132.28665	0.09017
zakharov	4	Mín	178.25188	6.50151	689.62116	1.58427
		Máx	603.88285	101.02651	3184.17509	1.80814
		Méd	346.74276	34.51547	1605.52032	**1.70111**
		DP	121.25796	21.30099	665.28492	0.05457
colville	4	Mín	727.81450	305.21077	24.73755	5.35644
		Máx	6129.14850	19458.30752	293.35922	15.58136
		Méd	1896.77323	4842.21022	137.75434	**9.92715**
		DP	995.85855	4309.51549	66.39333	2.34772
powersum	4	Mín	273.83659	356.11114	10.95370	2.77139
		Máx	2413.60564	10573.95963	63.16532	10.99521
		Méd	1039.78609	1910.48030	32.01762	**5.95296**
		DP	556.61624	1872.58863	16.26309	2.14188
shekel5	4	Mín	115.90481	53.58954	38.45812	2.58737
		Máx	1647.12743	1021.10081	127.23930	2.89260
		Méd	464.07707	280.65735	80.94453	**2.67937**
		DP	387.16405	189.21367	28.41476	0.07139
shekel7	4	Mín	86.62057	113.53062	58.29457	2.95218
		Máx	1535.66995	1040.03855	164.41992	3.33770
		Méd	401.09464	349.95216	101.91657	**3.11019**
		DP	283.31764	231.39341	33.06394	0.09082
shekel10	4	Mín	106.07346	49.07773	67.70606	3.62144
		Máx	1129.62908	667.69598	196.76389	4.39502
		Méd	472.05451	314.26985	138.65897	**3.84699**
		DP	245.26031	166.66412	44.21640	0.13602

Tabela 19 – Tempo computacional das execuções para n=10

Funções	**n**	**Resultados**	**Métodos - tempo(s)**			
			BRKGA	BRKGA-Levy	BRKGA-LS	BRKGA-Levy-LS
ackley	10	Mín	110.86546	11.60525	4520.07149	174.30546
		Máx	1008.55423	214.91555	22311.97388	1028.44873
		Méd	501.41006	**63.75780**	8082.36913	620.77465
		DP	209.74857	52.32270	5113.32434	235.28628
alpine2	10	Mín	143.91354	84.69354	52.59264	20.97396
		Máx	584.14384	1116.98104	132.43042	36.38796
		Méd	348.48840	355.87731	84.19879	**29.15093**
		DP	100.66149	309.12956	23.73616	3.54198
griewank	10	Mín	176.14365	664.25045	6765.56882	4607.81616
		Máx	1038.73116	9081.70229	35218.33115	15015.46764
		Méd	**447.33429**	3033.33565	19913.78213	6163.98694
		DP	192.58848	2590.92807	9558.83989	1981.54308
levy	10	Mín	207.80329	192.32315	94.93022	22.12955
		Máx	590.23354	5400.56550	448.09309	53.41700
		Méd	410.88576	1148.63034	207.22881	**30.98919**
		DP	107.42164	1077.16289	92.16939	6.64241
perm	10	Mín	219.72806	416.29457	96553.45070	2799.88713
		Máx	1069.70826	9776.88063	454372.81846	28429.41129
		Méd	**504.79175**	3115.16808	262800.30273	12476.45633
		DP	230.56621	2462.70115	102981.41775	7255.52088
perm0	10	Mín	249.31615	719.49290	40.87118	54.31875
		Máx	1005.89684	10031.51007	175.35476	130.50197
		Méd	532.84892	3408.92869	87.87146	**79.25920**
		DP	195.36017	2569.50101	43.50152	16.76789
powell	10	Mín	150.87452	492.40915	7.69003	18.98682
		Máx	1243.66898	13357.93971	29.62684	43.42598
		Méd	532.82812	3960.13854	**16.91192**	28.65014
		DP	259.41277	3725.14394	6.31626	5.05407
rastrigin	10	Mín	162.07097	84.56468	744.62964	26.79342
		Máx	849.61810	863.26268	2820.98124	172.68612
		Méd	449.29175	280.73809	1631.51326	**61.52943**
		DP	164.93841	217.41607	741.20033	43.13717
rosenbrock	10	Mín	241.99618	519.91903	3067.56294	1416.86524
		Máx	1228.58970	12975.33444	9596.90461	5438.94227
		Méd	**553.11140**	3872.09519	5129.67182	3047.80187
		DP	239.39173	3097.08457	2394.03690	1143.64079
sphere	10	Mín	170.00307	2.38845	7.47971	5.50057
		Máx	515.24712	28.51516	7.73084	10.68718
		Méd	360.32051	9.09818	**7.55922**	9.42848
		DP	102.48401	7.37977	0.07458	1.12262
sumsquares	10	Mín	221.37298	386.06034	18.46998	18.79310
		Máx	960.83111	6373.62432	41.24448	31.20520
		Méd	461.81736	2286.81757	26.21487	**23.18300**
		DP	178.40800	1798.40143	7.91719	2.35388
trid	10	Mín	358.50115	1687.09563	33759.99503	34.05607

Continua na próxima página

Funções	**n**	**Resultados**	**Métodos - tempo(s)**			
			BRKGA	BRKGA-Levy	BRKGA-LS	BRKGA-Levy-LS
		Máx	1220.39959	38527.74509	85536.05796	291.05245
		Méd	650.01181	10712.88856	49539.29221	**219.95113**
		DP	183.31959	9925.88880	15110.76199	67.01426
zakharov	10	Mín	227.86686	1013.93290	1110.01838	15.25940
		Máx	1793.15615	19908.05031	3049.67945	29.78065
		Méd	600.97234	5043.61830	2084.00515	**23.97346**
		DP	349.41355	4312.94948	633.93000	3.14999

Tabela 20 – Tempos de execução do C-GRASP.

Funções	**n**	**Resultados**	**Métodos**
			C-GRASP
bohachevsky	2	Mín	0.12639
		Máx	10.68801
		Méd	3.17061
		DP	2.72262
booth	2	Mín	0.02751
		Máx	0.07540
		Méd	0.05268
		DP	0.01454
branin	2	Mín	0.03217
		Máx	0.06407
		Méd	0.04206
		DP	0.00722
colville	4	Mín	2.71418
		Máx	313.17354
		Méd	114.14425
		DP	69.01474
perm0	4	Mín	0.08435
		Máx	2.72014
		Méd	0.69841
		DP	0.70465
powersum	4	Mín	0.04365
		Máx	17.65892
		Méd	5.36999
		DP	4.62639
rastrigin	10	Mín	178.19893
		Máx	618.62320
		Méd	303.21206
		DP	117.31629
sumsquares	10	Mín	202.91515
		Máx	673.54314
		Méd	345.62101
		DP	118.62115
trid	10	Mín	156.21528
		Máx	485.81827
		Méd	268.38812
		DP	80.26918
zakharov	10	Mín	184.59799
		Máx	961.51512
		Méd	471.92715
		DP	161.35867