

Um Estudo Empírico com o BRKGA Usando Chaves Aleatórias Inteiras

Anderson Zudio

Instituto de Matemática e Estatística, Universidade do Estado do Rio de Janeiro (IME/UERJ)
Rua São Francisco Xavier, 524, Maracanã, Rio de Janeiro – RJ – Brasil, 20550-900
azudio@ime.uerj.br

Igor Machado Coelho

Instituto de Computação, Universidade Federal Fluminense (IC/UFF)
Av. Gal. Milton Tavares de Souza, s/n - São Domingos, Niterói - RJ - Brasil, 24210-310
imcoelho@ic.uff.br

Cristiane Oliveira Faria

Instituto de Matemática e Estatística, Universidade do Estado do Rio de Janeiro (IME/UERJ)
Rua São Francisco Xavier, 524, Maracanã, Rio de Janeiro – RJ – Brasil, 20550-900
cofaria@ime.uerj.br

RESUMO

O algoritmo genético de chaves aleatórias viciadas, BRKGA, é uma meta-heurística proposta na literatura [Gonçalves e Resende, 2011]. O BRKGA possui um único componente dependente do problema chamado decodificador, que é responsável por converter um conjunto de chaves em uma representação de solução. Originalmente, as chaves são reais variando de zero a um. Assim, os pesquisadores frequentemente desconsideram a precisão das chaves durante sua implementação. Este trabalho estuda empiricamente o formato das chaves com duas implementações: uma com variáveis de precisão dupla e outra com variáveis inteiras, objetivando analisar o impacto na qualidade da solução. O problema alvo é o caixeiro viajante, abreviado como TSP. As implementações foram submetidas a um teste empírico com instâncias da literatura [Reinelt, 1991]. Os dados obtidos mostram que a versão com chaves inteiras, quando configurada, obtém resultados melhores ou equivalentes em comparação com os resultados da versão de precisão dupla, sugerindo que a precisão das chaves é um importante parâmetro a ser calibrado em aplicações que usam o BRKGA.

PALAVRAS CHAVE. BRKGA, TSP, Estudo.

Meta-heurísticas, Inteligência Computacional, Otimização Combinatória

ABSTRACT

The Biased Random-Key Genetic Algorithm (BRKGA) is a metaheuristic proposed in the literature [Gonçalves e Resende, 2011]. The BRKGA has a single problem-dependent component called the decoder, which is responsible for converting a set of keys into a solution. Originally, the keys are real numbers ranging from zero to one. Thus, researchers often disregard the precision of the keys. This work empirically studies the format of the keys with two implementations: one with double-precision variables and another with integer variables, aiming to analyze the impact on solution quality. The target problem is the Traveling Salesman Problem (TSP). The implementations were subjected to an empirical test with instances from the literature [Reinelt, 1991]. The results show that the integer-key version, when properly configured, achieves results that are better or equivalent compared to the double-precision version, suggesting that key precision is an important parameter to be tuned in applications using the BRKGA.

KEYWORDS. BRKGA. TSP. Study.

Metaheuristics, Computational Intelligence, Combinatorial Optimization

1. Introdução

Diversos trabalhos encontrados na literatura utilizam meta-heurísticas como o principal método para resolver problemas de otimização combinatória que modelam aplicações industriais do mundo real [Carrabs, 2021; Simões et al., 2021; Marzo et al., 2022]. Quando o pesquisador implementa uma meta-heurística para um problema específico, o resultado é uma heurística capaz de buscar soluções evitando persistir em ótimos locais [Gendreau et al., 2010]. Os algoritmos evolutivos (EA) são meta-heurísticas que trabalham com um conjunto de soluções, aplicando estratégias inspiradas nos princípios da evolução biológica e da seleção natural, como triagem, reprodução, transformação e recombinação [Mirjalili, 2019]. Nesse contexto, o algoritmo genético de chaves aleatórias ciciadas, abreviado do inglês como BRKGA, é um EA que foi proposto relativamente recentemente no trabalho [Gonçalves e Resende, 2011], sendo uma variação do algoritmo genético proposto anteriormente em [Bean, 1994]. Desde então, diversos trabalhos no estado da arte utilizam o BRKGA como o método principal para resolver problemas de otimização combinatória com aplicações em ambientes industriais [Gonçalves e Resende, 2013; Andrade et al., 2019; Pinto et al., 2021].

O BRKGA original trabalha com um único componente dependente do problema chamado decodificador, que o usuário precisa implementar para transformar um conjunto de números reais no intervalo $[0, 1]$, chamados *random-keys*, para uma representação de solução que a heurística pode avaliar em termos de qualidade. Todos os outros componentes são independentes do problema e somente trabalham com as próprias *random-keys*. Assim como outras meta-heurísticas, o BRKGA possui alguns parâmetros que o usuário precisa calibrar dependendo da aplicação prática. No entanto, grande parte dos pesquisadores que publicam trabalhos usando o BRKGA na literatura não fornecem qualquer implementação da sua respectiva heurística e não mencionam qualquer informação sobre a precisão utilizada para as *random-keys*, desconsiderando um aspecto essencial deste algoritmo evolutivo [Gonçalves e Resende, 2013; Andrade et al., 2019; Pinto et al., 2021,?; Andrade et al., 2021].

Neste contexto, este trabalho apresenta um estudo empírico com duas implementações do BRKGA para o clássico problema do caixeiro Viajante (TSP), que foi escolhido devido ao seu status de ampla divulgação na literatura e sua importância para o campo da pesquisa operacional [Lin e Kernighan, 1973; Lenstra e Kan, 1975]. A primeira implementação é uma abordagem padrão que utiliza variáveis de precisão dupla que assumem um valor no intervalo $[0, 1]$ para as *random-keys*, enquanto a segunda implementa as *random-keys* como inteiros de 32 bits, que operam em um intervalo determinado pelo usuário da aplicação. O teste empírico tem como objetivo comparar ambas as abordagens em termos da qualidade encontrada para a solução final, ao mesmo tempo em que verifica como a versão com *random-keys* como números inteiros afeta o desempenho computacional do BRKGA. Ambas as implementações foram submetidas a uma bateria de testes em um mesmo ambiente computacional dedicado com 24 instâncias do TSP amplamente usadas na literatura [Reinelt, 1991]. Porém, antes desta bateria, a versão com precisão dupla foi calibrada através da ferramenta iRace [López-Ibáñez et al., 2016], a fim de estabelecer um conjunto adequado de valores para cada parâmetro calibrável. Assim, todas as execuções com ambas as implementações usaram os mesmos parâmetros encontrados na fase de preliminar de calibração, objetivando a manter uma comparação justa. A versão com *random-keys* composta por números inteiros foi executada com um intervalo crescente para as chaves aleatórias de maneira a gerar resultados que enfatizem o impacto da diferença entre a possível menor chave e a maior na qualidade da solução final. Portanto, o teste realizado com cada aplicação difere apenas no formato das *random-keys*. O código-fonte utilizado neste trabalho, todas as instâncias do TSP e a os dados obtidos estão disponíveis publicamente ¹.

¹O código-fonte, as instâncias utilizadas e a melhor solução serão incluídos aqui na submissão final.

Os resultados deste estudo empírico mostram que a versão do BRKGA com inteira *random-keys* implementadas com variáveis inteiras obtém soluções melhores ou equivalentes do que a de precisão dupla, sem nenhum impacto negativo no desempenho para o conjunto de instâncias da literatura utilizado. Eles indicam que o formato das *random-keys* tem um impacto significativo em termos da qualidade da solução encontrada pela heurística quando o usuário calibra corretamente o intervalo entre a menor e a maior chave aleatória. Deste modo, os dados sugerem que o intervalo das *random-keys*, muitas vezes negligenciado pela comunidade científica em seus artigos na literatura, é um parâmetro essencial a ser detalhado em termos de precisão e formato para qualquer pessoa interessada em replicar qualquer trabalho realizado com o BRKGA.

Para sumarizar, a contribuição deste estudo empírico é:

- Uma comparação entre duas versões do BRKGA usando *random-keys* implementadas com variáveis de precisão dupla conforme o trabalho original e outra implementada usando números inteiros para um problema NP-Difícil amplamente conhecido na literatura [Garey e Johnson, 1979], enfatizando o impacto que o formato das *random-keys* tem na qualidade da solução final quando o usuário calibra corretamente o formato delas.
- Sugestão baseada em um extenso teste empírico de que o formato e o intervalo das *random-keys* são parâmetros essenciais que devem ser calibrados e detalhados em qualquer trabalho científico replicável que usa o BRKGA.

O restante deste artigo está estruturado da seguinte forma: A Seção 2 descreve brevemente o BRKGA; A Seção 3 detalha o decodificador do BRKGA para o TSP; A Seção 4 apresenta o experimento computacional e os resultados deste estudo; e a Seção 5 conclui este trabalho.

2. Sumário do BRKGA

O BRKGA é uma meta-heurística proposta recentemente no trabalho [Gonçalves e Resende, 2011]. Trata-se de uma variação do algoritmo evolutivo proposto anteriormente em Bean [1994], que os autores do primeiro trabalho citado rotulam como algoritmo genético de chaves aleatórias não viciadas, ou simplesmente algoritmo genético de chaves aleatórias (RKGA, do inglês Random-key Genetic Algorithm). O BRKGA opera com um conjunto de números reais no intervalo $[0, 1]$ para representar os indivíduos (soluções) denominados de *random-keys*. Cada componente que o usuário normalmente precisa modelar para um algoritmo evolutivo, como reprodução e mutação, é definido para trabalhar com as *random-keys*. Assim, o usuário só precisa implementar um decodificador, que transforma o indivíduo formado por chaves aleatórias em uma representação de solução que pode ser avaliada qualitativamente. A meta-heurística BRKGA utiliza apenas quatro parâmetros calibráveis: o tamanho máximo da população P_{max} , um fator de elitismo para a operação de cruzamento ρ , uma taxa de mutação ω e um valor de seleção elitista ϵ . Os dois últimos parâmetros são chamados BOT e TOP, respectivamente.

O BRKGA começa com uma população aleatória, chamada geração zero, composta por indivíduos representados por *random-keys*. O algoritmo classifica a primeira população por aptidão (qualidade da solução) e inicia seu processo iterativo. Cada iteração do BRKGA introduz uma nova geração, começando com um processo seletivo que copia os indivíduos mais aptos, os elites correspondendo a $100\epsilon\%$ da população, para uma nova. Em seguida, a iteração prossegue com um procedimento de reprodução para introduzir $100(1 - \epsilon - \omega)\%$ de descendentes, que possuem dois pais escolhidos aleatoriamente. Um dos pais é um indivíduo elite, enquanto o outro é um não-elite que não está presente na nova geração. O procedimento de cruzamento forma cada descendente copiando as chaves aleatórias de seus pais, uma a uma, onde a chave do pai elite é escolhida com

100% de chance em um processo de sorteio. O restante da nova população é composto por mutantes, que possuem um novo conjunto de *random-keys*, finalizando assim a geração. O BRKGA para de acordo com um determinado critério de parada. Geralmente, o critério de parada é um valor limite para desenvolver gerações. A heurística retorna o indivíduo mais apto (melhor solução) como resultado. Uma implementação do BRKGA usa o decodificador definido pelo usuário durante o processo de classificação para obter a porção elite da geração.

3. Decodificador do BRKGA para o TSP

O TSP clássico possui inúmeras aplicações na pesquisa operacional e é classificado como NP-Difícil, sendo um dos primeiros problemas a receber essa classificação Garey e Johnson [1979]. Seleccionamos esse problema para o estudo porque ele é um dos mais abordados na literatura. O TSP consiste em uma lista de N cidades conectadas e tem como objetivo encontrar a menor distância de viagem para visitar todas elas, começando e terminando em uma cidade específica. Frequentemente, ele é modelado como um grafo completo, com um vértice para cada cidade. Consequentemente, a solução para o TSP é um ciclo que contém todos os vértices, minimizando a distância percorrida.

Conforme mencionado brevemente na Seção 2, o único componente dependente do problema no BRKGA é o decodificador. Uma das principais vantagens do BRKGA é a capacidade de desenvolver uma heurística para um problema utilizando um algoritmo evolutivo composto por componentes reutilizáveis para implementar uma heurística para outro problema. Portanto, as abordagens que utilizam essa meta-heurística concentram-se no decodificador, que é responsável por converter um conjunto de chaves aleatórias de um indivíduo fornecido como entrada em uma representação de solução real para medir sua aptidão.

Podemos representar uma solução do TSP com uma permutação P_N de $\{1, \dots, N\}$ cidades, concedendo uma ordem de visitação. Utilizando essa representação de solução, cada indivíduo no BRKGA para o TSP possui N *random-keys*: uma para cada cidade na instância. O decodificador ordena as chaves aleatórias do indivíduo de entrada para obter uma permutação que representa a ordem de visitação das cidades. Assim, $\forall i, j \in P_N$, a cidade i é visitada antes da cidade j se $r(i) > r(j)$, onde $r(i), r(j)$ são as *random-keys* respectivas do indivíduo sendo decodificado. A aptidão de um indivíduo é a distância total percorrida pelo ciclo formado pela sua permutação. Em particular, ambas as implementações do BRKGA utilizadas neste trabalho ordenam as chaves aleatórias em ordem crescente, sendo a única diferença entre as duas o formato dos variáveis ordenadas: uma utilizando precisão dupla e a outra números inteiros.

4. Resultados Computacionais

Esta seção apresenta os resultados obtidos com duas implementações do BRKGA escritas na linguagem de programação C++17 para o estudo proposto. A primeira implementação utiliza variáveis de ponto flutuante de dupla precisão sem restrições sobre o número de casas decimais para representar as *random-keys*, enquanto a segunda utiliza variáveis inteiras de 32 bits. Na segunda versão, o usuário pode limitar o valor mínimo e máximo das chaves aleatórias geradas definindo um parâmetro de intervalo. Ambas as versões utilizam uma implementação idêntica do gerador de números pseudoaleatórios Mersenne Twister, semeado adequadamente no início de cada execução para manter um ambiente de execução controlado Matsumoto e Nishimura [1998]. O ambiente computacional utilizado para cada teste neste estudo possui um processador Intel Core i7-4790 de 3,6GHz e 16GB de RAM, executando o sistema operacional Ubuntu 20.04 LTS. Apenas um único núcleo do processador foi utilizado em cada teste.

Seleccionamos vinte e quatro instâncias disponíveis na literatura para este trabalho: todas elas disponíveis na TSPLIB Reinelt [1991]. O código-fonte, as instâncias e a melhor solução encontrada durante os testes deste estudo estão disponíveis em um repositório relacionado na Seção 1.

Antes de iniciar os testes, submetemos a primeira versão do BRKGA com números reais a um processo preliminar de calibração com a ferramenta iRace [López-Ibáñez et al., 2016]. Foi disponibilizado um total de 3240 execuções para o iRace e nove das vinte e quatro instâncias da TSPLIB. O critério de parada para todas as execuções foi atingir 10 segundos de processamento, pois o tamanho máximo da população afeta o tempo de execução de uma geração. O tipo de teste utilizado foi o *F-test*, com elitismo e o parâmetro de confiança foi mantido como 95% para esta calibração. A Tabela 1 relaciona os parâmetros calibráveis com o intervalo configurado para o iRace considerar e a melhor configuração estabelecida.

Tabela 1: Relação entre os parâmetros calibráveis com os intervalos usados no iRace e seu resultado final.

Parâmetro	Intervalo considerado	Valor final
Tamanho máximo da população	[50, 800]	$P_{max} = 102$
Taxa de seleção elitista (TOP)	[0.05, 0.45]	$\epsilon = 0.946$
Taxa de mutação (BOT)	[0.0, 0.4]	$\omega = 0.253$
Taxa de reprodução elitista	[0.55, 0.95]	$\rho = 0.946$

A ideia da calibração é fornecer um conjunto de valores de boa qualidade para todos os parâmetros calibráveis do BRKGA em todas as execuções deste estudo, a fim de manter uma comparação justa entre os formatos das *random-keys*. A calibração potencializa a heurística e fornece soluções finais de qualidade superior quando comparado com parâmetros selecionados manualmente arbitrariamente.

A bateria principal de testes com ambas as implementações BRKGA para o TSP utilizou uma qualidade de solução alvo como critério de parada, com base na instância. A ideia é retornar a melhor solução assim que a aplicação encontrar uma que supere este determinado limite em termos de qualidade. Para atingir essas metas, realizamos uma bateria de execuções preliminares com a implementação de precisão dupla do BRKGA, consistindo de trinta execuções para cada instância do estudo, utilizando sementes distintas no intervalo [1, 30] para alimentar o gerador pseudoaleatório. O tempo limite de todas as execuções para esta bateria de testes preliminar foi 10 segundos. A Tabela 2 resume o resultado dessa bateria, em que a primeira coluna relaciona o nome da instância e a segunda coluna detalha a distância Z_{avg} usada como alvo para o critério de parada em todas as execuções da bateria principal.

Após o teste preliminar, a implementação do BRKGA usando chaves aleatórias de precisão dupla foi executada trinta vezes para cada instância, utilizando o limite de solução alvo ou até que 60 segundos se passassem como critério de parada. Da mesma forma, para a versão inteira, utilizamos o intervalo $[0, 2^i]$, $i \in [10, 31]$ para as chaves aleatórias, executando assim cada configuração 30 vezes com o mesmo critério de parada para cada instância individualmente. Para cada instância separadamente, utilizamos sementes distintas em cada execução, com o intervalo [1, 30], sequencialmente, no gerador de números pseudoaleatórios, visando controlar o ambiente de execução.

As Tabelas 3-4 mostram os resultados da bateria onde: a primeira coluna apresenta o formato das chaves aleatórias e as colunas seguintes mostram os seguintes dados obtidos para cada instância: T_{avg} em todas as execuções para atingir o alvo; o tempo de execução mínimo e máximo T_{min}/T_{max} ; o desvio padrão T_{σ} do tempo de execução e a taxa de gerações por segundo Gen/T .

Os resultados em T_{avg} mostra que os melhores resultados são obtidos com a versão do BRKGA que usa variáveis inteiras para as *random-keys*. Em todas as instâncias, a solução alvo foi encontrada com um tempo médio menor ou equivalente em uma configuração que usa chaves aleatórias inteiras quando comparado com a versão de precisão dupla. As tabelas enfatizam em

Tabela 2: Solução alvo usada como critério de parada nas execuções para ambas as versões do BRKGA relacionadas com a instância.

Instância	Z_{avg}
berlin52	8794
kroD100	30369
pr226	256870
lin318	127220
pcb442	195668
d657	287816
rat783	64889
dsj1000	229894612
d198	31859
a280	6793
ali535	10489
ch130	9218
ch150	10428
rl1304	4934322
tsp225	7988
u159	79209
u574	199439
u724	299708
u1060	2865857
u1432	2157202
u1817	1280210
u2152	1644813
u2319	4130192
vm1084	3796124

Tabela 3: Resultados para a bateria principal de testes com as doze primeiras instâncias.

	Berlin52				KrodD100				Pr226				Lin318			
Format	T_{avg}	T_{min}/T_{max}	T_{σ}	Gen/T	T_{avg}	T_{min}/T_{max}	T_{σ}	Gen/T	T_{avg}	T_{min}/T_{max}	T_{σ}	Gen/T	T_{avg}	T_{min}/T_{max}	T_{σ}	Gen/T
Double	23.2	0.1/60.0	26.3	6449	21.0	1.5/60.0	26.3	3173	10.5	5.6/17.3	2.8	1286	10.3	6.5/13.5	1.8	855
²¹⁰	23.2	0.4/60.0	26.5	6092	50.8	3.7/60.0	17.4	3129	60.0	60.0/60.0	0.0	1286	60.0	60.0/60.0	0.0	1286
²¹¹	20.5	0.2/60.0	26.0	6211	41.7	5.4/60.0	21.5	3137	58.9	43.4/60.0	3.8	1288	60.0	60.0/60.0	0.0	1288
²¹²	30.9	0.4/60.0	26.5	6090	33.9	2.8/60.0	22.3	3125	43.8	9.9/60.0	17.6	1296	56.9	36.4/60.0	5.9	1296
²¹³	18.1	0.1/60.0	24.1	6203	25.7	2.3/60.0	23.8	3131	25.4	10.8/60.0	13.4	1297	36.0	12.9/60.0	6.1	1297
²¹⁴	15.2	0.2/60.0	22.0	6171	28.2	1.9/60.0	25.5	3134	16.4	7.5/36.5	6.5	1299	19.4	9.5/31.8	7.0	1299
²¹⁵	13.7	0.2/60.0	21.5	6152	21.5	1.7/60.0	21.6	3123	13.7	5.9/31.7	6.2	1293	12.9	8.5/24.5	3.3	1293
²¹⁶	21.3	0.3/60.0	25.4	6185	30.0	3.2/60.0	25.3	3129	12.9	5.9/21.0	4.1	1296	12.5	8.1/19.7	3.0	1296
²¹⁷	18.5	0.1/60.0	25.0	6283	20.6	2.0/60.0	20.4	3122	11.8	5.4/21.3	4.3	1292	10.7	6.0/16.5	3.0	1292
²¹⁸	24.5	0.2/60.0	25.3	6185	22.6	2.5/60.0	23.4	3120	12.2	5.2/30.9	6.0	1295	9.7	7.2/12.8	1.4	1295
²¹⁹	12.7	0.2/60.0	20.2	6028	21.3	2.0/60.0	20.4	3123	10.6	4.0/22.1	3.9	1293	10.1	7.0/19.9	2.6	1293
²²⁰	17.7	0.2/60.0	24.3	6065	17.5	2.0/60.0	17.9	3119	12.5	7.8/22.8	3.9	1294	9.8	7.0/13.7	1.6	1294
²²¹	24.1	0.2/60.0	28.3	6116	19.3	3.0/60.0	19.7	3118	10.9	5.8/20.4	3.7	1293	9.6	7.4/13.0	1.6	1293
²²²	26.0	0.2/60.0	25.0	6089	26.4	2.5/60.0	23.1	3131	10.1	5.7/16.4	2.8	1292	9.8	7.2/14.6	1.5	1292
²²³	19.6	0.2/60.0	26.2	6129	19.4	1.8/60.0	20.1	3133	10.9	5.1/16.3	2.7	1291	9.0	6.1/14.8	1.8	1291
²²⁴	17.4	0.2/60.0	21.0	6143	26.5	1.4/60.0	23.0	3193	10.8	5.6/18.8	3.5	1292	9.5	6.2/15.7	2.0	1292
²²⁵	12.9	0.2/60.0	18.7	6102	26.4	3.1/60.0	24.0	3197	10.6	5.4/19.8	3.8	1295	9.0	6.0/11.4	1.6	1295
²²⁶	21.2	0.2/60.0	24.4	6128	24.8	2.0/60.0	24.6	3186	11.4	4.2/21.0	4.0	1292	9.8	6.2/14.6	1.8	1292
²²⁷	12.6	0.3/60.0	22.1	6141	28.8	3.0/60.0	22.4	3190	10.5	4.8/18.6	3.4	1288	9.4	6.9/12.3	1.4	1288
²²⁸	18.6	0.3/60.0	23.9	6079	29.2	3.3/60.0	24.0	3174	11.4	6.0/31.0	4.9	1284	10.4	7.2/17.2	2.6	1284
²²⁹	24.1	0.3/60.0	27.1	6121	23.3	1.5/60.0	22.2	3157	10.6	6.1/20.9	3.3	1271	9.9	7.3/15.2	1.8	1271
²³⁰	25.6	0.2/60.0	26.8	6056	22.7	2.4/60.0	21.4	3079	11.8	6.3/20.3	4.2	1245	10.0	7.4/14.9	1.6	1245
²³¹	26.3	0.3/60.0	25.9	5585	17.3	1.7/60.0	20.8	2858	12.2	6.5/23.2	3.9	1164	10.5	7.1/18.5	2.4	1164
	Pcb442				d657				Rat783				Dsj1000			
Double	10.3	8.2/12.3	1.2	570	10.3	7.7/12.0	0.9	341	10.3	8.6/11.9	0.8	269	10.3	8.9/12.4	0.8	193
²¹⁰	60.0	60.0/60.0	0.0	581	60.0	60.0/60.0	0.0	347	60.0	60.0/60.0	0.0	274	60.0	60.0/60.0	0.0	196
²¹¹	60.0	60.0/60.0	0.0	580	60.0	60.0/60.0	0.0	347	60.0	60.0/60.0	0.0	274	60.0	60.0/60.0	0.0	197
²¹²	60.0	60.0/60.0	0.0	582	60.0	60.0/60.0	0.0	346	60.0	60.0/60.0	0.0	274	60.0	60.0/60.0	0.0	197
²¹³	51.4	27.4/60.0	9.8	584	60.0	60.0/60.0	0.0	347	60.0	60.0/60.0	0.0	274	60.0	60.0/60.0	0.0	197
²¹⁴	29.0	12.4/57.4	11.5	581	52.1	29.5/60.0	9.3	347	58.5	47.8/60.0	3.3	274	59.4	50.9/60.0	2.1	197
²¹⁵	18.9	8.4/37.4	6.6	579	25.9	11.8/60.0	9.6	345	40.1	20.0/60.0	13.5	275	44.4	21.3/60.0	12.3	198
²¹⁶	12.6	10.1/17.0	1.6	578	17.9	10.7/32.1	5.8	344	21.0	12.7/44.8	7.9	273	25.3	16.0/47.2	8.1	197
²¹⁷	11.8	7.7/18.8	2.2	577	13.5	9.9/21.3	2.7	343	13.8	10.0/24.8	3.4	272	15.9	9.7/24.4	4.2	196
²¹⁸	10.9	9.0/13.6	1.2	576	12.7	9.4/23.1	3.3	342	11.9	8.8/15.6	1.9	270	13.3	10.2/19.2	2.4	196
²¹⁹	10.9	8.5/14.3	1.3	576	10.8	7.8/15.0	1.5	342	11.2	8.4/16.7	2.0	271	12.0	8.6/19.0	2.6	196
²²⁰	10.3	8.3/13.7	1.4	575	10.4	7.9/14.7	1.3	342	10.6	8.4/14.4	1.3	271	11.5	9.2/19.6	2.1	195
²²¹	10.5	7.7/14.2	1.6	575	10.4	9.1/14.1	1.1	341	10.8	9.0/15.5	1.5	271	10.3	9.0/12.6	0.8	195
²²²	10.8	8.3/15.0	1.5	576	10.3	8.8/13.7	1.0	341	10.3	8.4/12.6	1.1	270	10.3	9.0/12.8	0.9	196
²²³	10.4	7.8/14.1	1.5	575	10.3	8.8/14.2	1.1	341	10.1	9.0/12.0	0.7	271	10.2	8.3/12.1	0.8	196
²²⁴	10.4	7.9/12.3	1.1	575	10.1	8.4/12.0	0.9	340	9.6	8.0/11.5	0.9	271	10.3	8.5/13.2	1.1	196
²²⁵	9.8	7.5/12.4	1.2	576	9.9	8.4/11.9	0.9	341	10.1	8.5/12.1	0.8	271	10.2	9.0/11.6	0.7	195
²²⁶	10.1	7.9/12.2	1.0	576	10.2	8.5/11.9	0.8	341	10.5	9.1/12.0	0.7	271	10.3	9.0/12.1	0.7	195
²²⁷	10.7	8.7/12.7	1.2	575	10.2	8.5/12.6	0.9	341	10.4	8.1/12.8	0.8	269	10.4	9.2/12.3	0.8	195
²²⁸	10.1	7.9/13.3	1.3	583	10.1	8.6/12.2	0.8	339	10.3	9.0/12.3	0.8	268	10.2	8.8/11.5	0.7	197
²²⁹	10.4	8.5/15.2	1.5	583	10.2	8.3/12.6	0.9	337	10.2	8.8/12.0	0.9	267	10.0	8.7/11.2	0.7	199
²³⁰	10.1	7.5/13.3	1.4	572	10.5	8.7/13.8	1.0	331	10.1	9.1/12.2	0.8	270	10.4	8.9/12.1	0.8	199
²³¹	10.4	8.3/12.2	1.0	539	10.9	9.0/13.3	1.0	313	10.7	8.9/12.7	0.8	258	10.4	8.9/12.2	0.8	190
	d198				a280				Ali535				Ch130			
Double	12.5	4.0/60.0	10.4	1486	10.3	6.5/15.6	2.5	996	10.4	7.6/14.3	1.8	444	14.2	3.7/60.0	15.1	2355
²¹⁰	60.0	60.0/60.0	0.0	1492	60.0	60.0/60.0	0.0	997	60.0	60.0/60.0	0.0	459	58.7	22.7/60.0	6.8	2356
²¹¹	56.8	16.9/60.0	10.3	1490	60.0	60.0/60.0	0.0	997	60.0	60.0/60.0	0.0	458	45.4	9.1/60.0	18.6	2359
²¹²	45.0	17.8/60.0	16.2	1494	55.6	26.9/60.0	9.4	1000	60.0	60.0/60.0	0.0	459	29.4	6.4/60.0	19.2	2358
²¹³	26.1	5.1/60.0	18.0	1496	30.9	14.3/60.0	13.5	1001	56.8	31.7/60.0	6.7	459	26.5	4.2/60.0	21.9	2352
²¹⁴	23.2	5.2/60.0	14.2	1494	18.6	10.0/58.4	9.2	1000	40.1	19.4/60.0	12.4	459	20.8	2.8/60.0	17.4	2356
²¹⁵	12.7	2.9/23.7	5.8	1489	12.1	8.0/19.2	3.0	998	18.8	10.5/40.3	6.8	457	16.8	3.4/60.0	16.2	2350
²¹⁶	14.6	6.4/60.0	10.3	1492	11.6	7.0/17.2	3.0	998	13.7	8.1/20.0	3.0	453	12.0	3.8/40.5	9.2	2355
²¹⁷	13.5	3.7/60.0	11.1	1491	12.2	7.5/25.4	3.5	995	11.6	6.8/20.1	3.0	451	10.8	3.7/39.8	8.8	2354
²¹⁸	12.2	4.7/24.3	5.6	1491	11.1	7.2/22.8	3.1	998	11.2	7.8/23.5	2.8	452	13.9	4.8/44.6	9.7	2355
²¹⁹	13.7	4.3/34.7	8.0	1495	10.8	6.3/16.2	2.4	998	10.7	7.2/13.0	1.6	452	16.6	4.7/60.0	14.1	2356
²²⁰	12.7	5.0/32.2	6.7	1493	10.3	6.6/14.4	2.0	999	10.4	7.1/15.0	1.8	451	14.2	3.8/60.0	15.2	2354
²²¹	16.9	6.0/60.0	13.8	1495	10.8	6.9/17.4	2.1	997	11.0	8.5/14.1	1.3	451	12.6	4.1/60.0	13.5	2350
²²²	11.3	4.6/50.9	9.0	1490	10.4	6.3/16.7	2.5	999	10.2	7.3/12.9	1.5	451	16.8	3.6/60.0	17.4	2347
²²³	14.8	4.5/45.9	8.7	1500	10.4	7.6/15.2	2.0	998								

Tabela 4: Resultados para a bateria principal de testes com as doze últimas instâncias.

	Ch150				R11304				Tsp225				u159			
Format	Tavg	Tmin/Tmax	Tσ	Gen/T	Tavg	Tmin/Tmax	Tσ	Gen/T	Tavg	Tmin/Tmax	Tσ	Gen/T	Tavg	Tmin/Tmax	Tσ	Gen/T
Double	14.0	3.2/60.0	11.8	2012	10.4	9.7/11.7	0.5	136	10.7	6.6/18.0	3.1	1284	11.3	3.7/24.5	6.4	1911
²¹⁰	60.0	60.0/60.0	0.0	2021	60.0	60.0/60.0	0.0	140	60.0	60.0/60.0	0.0	1287	60.0	60.0/60.0	0.0	1912
²¹¹	56.5	11.6/60.0	11.5	2020	60.0	60.0/60.0	0.0	141	60.0	60.0/60.0	0.0	1289	49.9	12.5/60.0	16.7	1919
²¹²	37.0	6.9/60.0	21.6	2016	60.0	60.0/60.0	0.0	141	49.5	17.6/60.0	15.8	1292	28.8	7.4/60.0	18.6	1919
²¹³	22.5	6.7/60.0	15.5	2013	60.0	60.0/60.0	0.0	141	21.6	7.9/60.0	14.1	1295	18.0	4.9/60.0	13.7	1918
²¹⁴	18.6	6.0/60.0	13.7	2018	60.0	60.0/60.0	0.0	141	17.5	7.2/55.1	9.9	1291	10.3	3.5/24.8	4.3	1917
²¹⁵	17.1	3.6/60.0	14.0	2022	59.9	58.0/60.0	0.4	141	13.3	5.8/40.5	6.5	1290	8.8	2.5/22.8	4.1	1917
²¹⁶	13.6	4.1/60.0	11.4	2020	40.4	17.3/60.0	12.5	142	12.2	5.8/29.3	5.8	1289	9.6	4.0/35.7	5.9	1915
²¹⁷	11.0	4.7/30.3	6.2	2018	20.1	12.7/46.1	8.2	141	9.8	4.4/16.1	2.4	1290	7.7	3.8/15.7	3.0	1918
²¹⁸	12.2	4.9/40.6	7.7	2020	14.4	9.7/34.4	5.6	144	10.2	4.4/21.7	3.6	1286	9.0	4.2/39.4	6.4	1916
²¹⁹	11.7	4.1/26.9	5.4	2016	11.8	9.2/18.3	2.1	145	9.7	5.5/16.2	2.4	1289	8.5	4.2/34.4	5.7	1911
²²⁰	11.7	4.7/32.9	6.2	2019	10.6	8.6/15.0	1.5	145	10.0	5.1/14.8	2.2	1293	8.3	4.1/20.8	3.6	1916
²²¹	10.2	3.1/32.9	6.2	2013	9.9	8.7/12.3	1.0	145	10.3	6.2/15.2	2.3	1291	7.9	4.0/18.4	3.3	1918
²²²	11.6	5.1/28.6	5.2	2011	9.6	8.3/10.9	0.6	144	9.2	5.6/14.4	2.6	1290	8.7	4.5/15.2	3.2	1917
²²³	11.0	5.4/30.8	5.3	2005	9.9	8.1/13.8	1.2	145	9.2	4.8/15.8	2.6	1290	7.8	3.6/13.0	2.5	1917
²²⁴	13.5	4.6/43.9	7.9	2011	9.8	8.6/10.9	0.6	144	9.3	6.4/17.4	2.1	1292	7.8	4.0/20.3	3.3	1915
²²⁵	13.4	4.1/60.0	9.8	2010	9.8	8.4/11.1	0.6	144	9.6	5.2/16.5	2.8	1302	7.4	3.4/12.8	2.2	1910
²²⁶	12.7	4.8/60.0	10.6	2013	9.9	8.7/11.0	0.5	144	9.4	6.8/15.5	2.2	1318	9.5	5.5/20.3	4.0	1916
²²⁷	12.4	3.5/24.8	5.5	2008	9.6	8.1/11.3	0.8	144	9.5	6.3/16.8	2.4	1315	8.0	3.7/15.5	3.0	1906
²²⁸	11.8	5.5/30.1	5.6	2005	9.7	8.6/11.5	0.6	144	9.5	6.2/14.2	2.3	1311	8.1	2.8/19.8	3.4	1906
²²⁹	11.2	5.1/23.7	4.2	1985	9.6	8.4/10.6	0.5	143	9.6	6.2/16.7	2.0	1299	8.0	2.7/14.6	3.5	1883
²³⁰	15.2	5.5/60.0	12.7	1943	9.7	8.6/11.0	0.6	141	9.6	5.4/15.4	2.7	1269	9.2	4.3/18.8	3.5	1847
²³¹	12.0	5.5/42.7	6.8	1805	10.3	9.2/11.7	0.7	135	10.4	6.4/19.0	2.9	1186	9.1	4.1/19.4	3.9	1713
	u724				u574				u1060				u1432			
Double	10.3	9.4/11.8	0.7	299	10.3	8.3/12.1	1.0	408	10.3	9.1/11.7	0.7	178	10.3	9.1/11.7	0.7	178
²¹⁰	60.0	60.0/60.0	0.0	302	60.0	60.0/60.0	0.0	421	60.0	60.0/60.0	0.0	181	60.0	60.0/60.0	0.0	123
²¹¹	60.0	60.0/60.0	0.0	304	60.0	60.0/60.0	0.0	421	60.0	60.0/60.0	0.0	181	60.0	60.0/60.0	0.0	123
²¹²	60.0	60.0/60.0	0.0	304	60.0	60.0/60.0	0.0	421	60.0	60.0/60.0	0.0	183	60.0	60.0/60.0	0.0	124
²¹³	60.0	60.0/60.0	0.0	304	59.7	51.8/60.0	1.5	422	60.0	60.0/60.0	0.0	183	60.0	60.0/60.0	0.0	124
²¹⁴	54.4	40.5/60.0	6.8	304	36.6	18.1/60.0	12.5	422	60.0	60.0/60.0	0.0	183	60.0	60.0/60.0	0.0	124
²¹⁵	29.5	15.4/47.9	8.9	304	22.1	11.0/38.3	6.7	420	54.9	26.2/60.0	7.5	183	60.0	60.0/60.0	0.0	124
²¹⁶	18.3	11.1/27.8	4.6	302	15.8	8.3/31.7	4.5	420	28.0	13.9/55.0	9.9	183	43.5	21.5/60.0	13.4	124
²¹⁷	13.5	9.1/20.5	2.8	302	12.6	9.5/21.3	2.5	418	17.9	9.8/41.2	7.2	182	23.7	11.6/44.7	8.3	124
²¹⁸	11.4	8.0/17.1	1.9	299	11.0	8.7/15.4	1.7	426	13.6	9.7/24.3	3.4	182	15.2	9.8/27.4	4.6	124
²¹⁹	10.6	9.0/14.7	1.5	301	9.9	8.1/11.6	0.8	426	12.1	9.1/18.4	2.4	182	12.3	9.3/18.5	2.3	124
²²⁰	10.8	8.8/16.7	1.5	299	10.2	7.6/15.9	1.4	426	10.8	9.0/13.3	1.1	182	11.0	9.2/15.0	1.4	124
²²¹	11.2	8.7/23.9	2.8	300	9.9	8.0/12.3	1.0	426	10.6	8.7/13.2	1.1	181	10.4	8.8/11.6	0.6	124
²²²	10.1	8.4/12.3	0.9	301	9.8	8.0/13.0	1.1	425	10.3	8.7/12.0	0.8	182	10.3	9.1/12.1	0.7	124
²²³	10.0	8.6/11.8	0.8	300	9.9	8.0/12.0	1.1	425	10.4	9.4/12.6	0.8	181	10.1	9.1/12.0	0.7	123
²²⁴	10.3	9.1/11.6	0.7	301	9.5	7.4/11.4	1.0	426	10.2	8.9/11.9	0.7	182	10.1	9.0/11.1	0.5	124
²²⁵	10.1	8.6/12.3	0.8	300	10.1	8.5/12.3	0.9	427	10.5	9.2/12.2	0.8	181	9.9	8.8/11.8	0.7	127
²²⁶	9.9	8.5/12.0	0.8	300	9.9	7.5/12.5	1.1	426	10.4	9.3/12.6	0.8	182	9.7	8.5/10.5	0.6	127
²²⁷	10.2	8.5/11.6	0.7	300	9.8	8.1/12.7	1.1	423	10.2	8.9/11.4	0.6	184	9.9	8.7/14.0	0.9	127
²²⁸	10.3	8.3/12.9	1.0	299	9.8	7.8/12.2	1.0	424	10.1	8.6/11.2	0.7	186	9.8	8.8/10.7	0.5	127
²²⁹	10.1	8.7/12.4	0.8	297	9.9	8.1/12.8	1.0	420	10.2	9.0/11.5	0.7	185	9.9	9.0/11.1	0.6	126
²³⁰	10.4	8.9/12.1	0.9	292	10.1	8.0/12.1	1.2	413	10.2	8.9/11.9	0.8	183	9.9	8.6/11.5	0.6	126
²³¹	10.8	9.2/12.8	0.9	277	11.0	7.6/13.5	1.3	391	10.8	9.3/12.0	0.6	174	10.3	9.0/11.5	0.6	120
	u1817				u2152				u2319				Vm1084			
Double	10.4	9.4/12.5	0.7	91	10.3	9.4/11.4	0.5	74	10.3	9.0/11.7	0.7	68	10.3	8.7/11.4	0.7	173
²¹⁰	60.0	60.0/60.0	0.0	93	60.0	60.0/60.0	0.0	76	60.0	60.0/60.0	0.0	70	60.0	60.0/60.0	0.0	177
²¹¹	60.0	60.0/60.0	0.0	93	60.0	60.0/60.0	0.0	76	60.0	60.0/60.0	0.0	69	60.0	60.0/60.0	0.0	178
²¹²	60.0	60.0/60.0	0.0	93	60.0	60.0/60.0	0.0	76	60.0	60.0/60.0	0.0	70	60.0	60.0/60.0	0.0	178
²¹³	60.0	60.0/60.0	0.0	93	60.0	60.0/60.0	0.0	76	60.0	60.0/60.0	0.0	70	60.0	60.0/60.0	0.0	178
²¹⁴	60.0	60.0/60.0	0.0	93	60.0	60.0/60.0	0.0	76	60.0	60.0/60.0	0.0	70	60.0	60.0/60.0	0.0	178
²¹⁵	60.0	60.0/60.0	0.0	93	60.0	60.0/60.0	0.0	76	60.0	60.0/60.0	0.0	70	53.5	37.3/60.0	7.7	178
²¹⁶	57.7	45.0/60.0	4.5	93	60.0	60.0/60.0	0.0	76	58.7	31.0/60.0	5.6	70	27.3	13.2/50.2	7.5	178
²¹⁷	31.4	11.6/60.0	13.6	93	38.6	17.2/60.0	15.1	76	44.5	22.1/60.0	13.5	70	18.6	10.1/31.0	6.4	178
²¹⁸	18.5	11.2/46.2	6.5	93	21.1	9.4/43.0	7.6	76	22.2	10.7/57.0	9.9	70	14.3	9.6/28.1	4.3	177
²¹⁹	12.9	9.1/25.7	4.1	93	13.5	9.3/19.5	2.6	76	14.5	9.5/27.4	4.6	70	11.5	9.0/16.3	1.9	177
²²⁰	11.6	9.6/19.2	1.8	93	11.1	9.3/15.5	1.6	76	12.2	9.4/18.1	2.2	70	10.3	8.4/14.2	1.3	177
²²¹	11.0	9.1/16.3	1.4	93	11.2	9.2/16.4	1.6	76	11.1	9.6/16.7	1.6	70	10.3	8.7/12.9	0.9	177
²²²	10.2	9.2/11.4	0.6	93	10.2	9.0/11.3	0.6	76	10.3	9.0/14.3	1.0	70	10.0	9.2/13.0	0.7	177
²²³	10.2	8.6/12.6	0.7	93	10.2	8.9/11.8	0.7	76	10.4	9.0/15.7	1.3	69	10.4	9.0/14.3	1.0	177
²²⁴	10.4	9.4/13.4	0.7	93	10.0	8.8/11.4	0.6	76	10.0	8.9/12.1	0.6	70				

negrito o melhor resultado em termos do tempo de execução para atingir o alvo, porém há vários casos em que a versão com chaves aleatórias inteiras consegue obter médias menores para o tempo de execução quando comparado ao de precisão dupla. Pode-se observar também pelos valores de T_{max} e T_{min} que as melhores configurações envolvem *random-keys* com formato de variáveis inteiras. Esses dados sugerem que o formato das chaves aleatórias influencia positivamente a qualidade final obtida para este conjunto de instâncias, implicando que a versão com números inteiros devidamente calibrada obtém consistentemente resultados melhores ou equivalentes do que a abordagem de precisão dupla nesse teste. Quando os melhores valores de T_{avg} são submetidos em um teste estatístico de Wilcoxon com parâmetro de confiança de $p = 0.01$ contra os valores obtidos pela versão de números reais, o teste informa que a diferença encontrada é estatisticamente significativa.

Por fim, os resultados de Gen/T indicam que as duas versões tem performance similar, pois o número de iterações por segundo de ambas é similar nas duas versões para qualquer configuração e formato de *random-key*. Este resultado indica que *random-keys* com formato de números inteiros não tem um impacto positivo e nem negativo na performance da heurística para as instâncias usadas.

5. Conclusão

Este trabalho apresentou um estudo empírico sobre o formato das chaves aleatórias do algoritmo genético de chave aleatória viciada (BRKGA) por meio do conhecido problema do caixeiro Viajante (TSP). O teste computacional utilizou vinte e quatro instâncias em uma bateria de execuções com duas implementações do BRKGA: uma utilizando precisão dupla e outra com inteiros de 32 bits para as *random-keys*. A implementação com inteiros permite que o usuário selecione o intervalo a ser usado para a geração das chaves aleatórias como entrada. Realizamos os testes em uma configuração computacional semelhante para garantir uma comparação justa entre as duas versões. Ambas as implementações foram executadas com o mesmo valor para os parâmetros calibráveis do BRKGA, diferindo apenas no formato das chaves aleatórias. Os dados resultantes deste estudo sugerem que o formato das chaves aleatórias tem pouco impacto no desempenho da heurística para o conjunto de instâncias utilizado. Os dados também indicam que, quando devidamente calibrada, a versão com inteiros pode produzir consistentemente soluções melhores do que a versão com precisão dupla para este conjunto de instâncias. Os produtos deste trabalho, como o código-fonte de ambas as implementações, os dados de resultado e as instâncias, estão disponíveis publicamente.

Em trabalhos futuros, pretendemos implementar heurísticas baseadas em BRKGA para outros problemas conhecidos, a fim de expandir os resultados. Também desejamos calibrar o algoritmo por meio de uma ferramenta estatística. Por fim, o desempenho em termos de qualidade positiva das soluções pode inspirar o desenvolvimento de heurísticas baseadas em BRKGA que utilizam o poder de processamento paralelo de ambientes computacionais com unidades de processamento gráfico (GPU) como processadores de propósito geral, juntamente com a CPU.

Agradecimentos

Os recursos computacionais usados para esse estudo foram providenciados pelo Laboratório de Inteligência Computacional (LaBIC) da Universidade Federal Fluminense (UFF). O código fonte dessa aplicação foi baseado no OptFrame framework Coelho et al. [2011]. Além disso, o presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001, do Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) processo (PQ 2021 315059/2021-4) e da Fundação Carlos Chagas Filho de Amparo à Pesquisa do Estado do Rio de Janeiro (FAPERJ) processo (JCNE 2021 E-26/201.451/2022).

Referências

- Andrade, C. E., Silva, T., e Pessoa, L. S. (2019). Minimizing flowtime in a flowshop scheduling problem with a biased random-key genetic algorithm. *Expert Systems with Applications*, 128: 67–80.
- Andrade, C. E., Toso, R. F., Gonçalves, J. F., e Resende, M. G. (2021). The multi-parent biased random-key genetic algorithm with implicit path-relinking and its real-world applications. *European Journal of Operational Research*, 289(1):17–30.
- Bean, J. C. (1994). Genetic algorithms and random keys for sequencing and optimization. *ORSA Journal on computing*, 6(2):154–160.
- Carrabs, F. (2021). A biased random-key genetic algorithm for the set orienteering problem. *European Journal of Operational Research*, 292(3):830–854.
- Coelho, I. M., Munhoz, P. L. A., Haddad, M. N., Coelho, V. N., de Melo Silva, M., Souza, M. J. F., e Ochi, L. S. (2011). Optframe: a computational framework for combinatorial optimization problems. In *Proceedings of VII ALIO/EURO*, p. 51–54, Porto. Workshop on Applied Combinatorial Optimization, VII, 2011, Porto, ALIO-EURO.
- Garey, M. R. e Johnson, D. S. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, 1 edition. ISBN 0716710447.
- Gendreau, M., Potvin, J.-Y., et al. (2010). *Handbook of metaheuristics*, volume 2. Springer.
- Gonçalves, J. F. e Resende, M. G. (2011). Biased random-key genetic algorithms for combinatorial optimization. *Journal of Heuristics*, 17(5):487–525.
- Gonçalves, J. F. e Resende, M. G. (2013). A biased random key genetic algorithm for 2d and 3d bin packing problems. *International Journal of Production Economics*, 145(2):500–510.
- Lenstra, J. K. e Kan, A. R. (1975). Some simple applications of the travelling salesman problem. *Journal of the Operational Research Society*, 26(4):717–733.
- Lin, S. e Kernighan, B. W. (1973). An effective heuristic algorithm for the traveling-salesman problem. *Operations research*, 21(2):498–516.
- López-Ibáñez, M., Dubois-Lacoste, J., Cáceres, L. P., Birattari, M., e Stützle, T. (2016). The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, 3:43–58.
- Marzo, R. G., Melo, R. A., Ribeiro, C. C., e Santos, M. C. (2022). New formulations and branch-and-cut procedures for the longest induced path problem. *Computers & Operations Research*, 139:105627.
- Matsumoto, M. e Nishimura, T. (1998). Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 8(1):3–30.
- Mirjalili, S. (2019). Evolutionary algorithms and neural networks. In *Studies in computational intelligence*, volume 780. Springer.

- Pinto, B. Q., Ribeiro, C. C., Riveaux, J. A., e Rosseti, I. (2021). A brkga-based matheuristic for the maximum quasi-clique problem with an exact local search strategy. *RAIRO-Operations Research*, 55:S741–S763.
- Reinelt, G. (1991). TspLib—a traveling salesman problem library. *ORSA journal on computing*, 3 (4):376–384.
- Simões, E. M. L., Batista, L. D. S., e Souza, M. J. F. (2021). A matheuristic algorithm for the multiple-depot vehicle and crew scheduling problem. *IEEE Access*, 9:155897–155923.