BRKGA com configuração *on-line* de parâmetros aplicado a problemas de otimização combinatória

Antonio Augusto Chaves
Instituto de Ciência e Tecnologia, Universidade Federal de São Paulo – UNIFESP
São José dos Campos, Brasil, 12247-014
antonio.chaves@unifesp.br

José Fernando Gonçalves
Faculdade de Economia, Universidade do Porto – UP
Porto, Portugal, 4200-464
jfgoncal@fep.up.pt

RESUMO

Metaheurísticas são ferramentas robustas para resolver problemas de otimização combinatória. Entretanto, o desenvolvimento de uma metaheurística eficiente requer uma configuração adequada dos seus parâmetros. Esta é uma tarefa difícil e custosa para os desenvolvedores, tendo em vista que as metaheurísticas possuem diferentes parâmetros e que há uma correlação entre eles. Neste artigo desenvolvemos uma variação do *Biased Random-Key Genetic Algorithm* (BRKGA) com configuração *on-line* de parâmetros, que chamamos A-BRKGA. Os parâmetros são controlados por meio de regras determinísticas e auto-adaptáveis. Esta característica associada ao uso de representação de soluções por chaves aleatórias fornece um método com código ágil para ser reutilizado. Para validar o A-BRKGA resolvemos três problemas de otimização combinatória: Problema de Agrupamento Centrado Capacitado, Problema de Alocação Tática de Berços e Problema de Empacotamento. Analisando os resultados obtidos é possível afirmar que o A-BRKGA apresenta resultados competitivos em termos de qualidade de solução e tempo de execução.

PALAVRAS CHAVE. Algoritmo Genético; Chaves Aleatórias; Configuração de Parâmetros.

Área Principal: Metaheurísticas

ABSTRACT

Metaheuristics are robust tools for solving combinatorial optimization problems. However, the development of an efficient metaheuristic requires adequate configuration of its parameters. This is a difficult and costly task for developers, given that metaheuristics have different parameters and there is a correlation between them. In this paper we have developed a variation of the Biased Random-Key Genetic Algorithm (BRKGA) with on-line parameters control, which we call A-BRKGA. Parameters are controlled by deterministic and self-adaptive rules. This feature associated with the use of solution representation by random-keys provides a method with agile code to be reused. To validate A-BRKGA, we solve three combinatorial optimization problems: Capacitated Centered Clustering Problem, Tactical Berth Allocation Problem, and Bin Packaging Problem. Analyzing the results obtained, it is possible to state that A-BRKGA presents competitive results in terms of solution quality and execution time.

KEYWORDS. Genetic Algorithm; Random-keys; Parameter control.

Main Area: Metaheuristics

https://proceedings.science/p/85286?lang=pt-br

1. Introdução

As metaheurísticas são boas ferramentas para obter soluções de qualidade para problemas de otimização combinatória. Entretanto, os desenvolvedores precisam tomar várias decisões ao projetar metaheurísticas, por exemplo: representação de solução; cálculo da função objetivo; tratamento das restrições; estruturas de vizinhança; solução inicial; e configuração dos parâmetros. Essas decisões têm muitas alternativas possíveis e as escolhas apropriadas são muito importantes para encontrar boas soluções para um problema de otimização específico.

O *Biased Random-Keys Genetic Algorithm* (BRKGA), proposto por Goncalves e Resende [2011], é um algoritmo evolutivo que simplifica algumas destas decisões. Ele se baseia no conceito de chaves aleatórias para representar uma solução e em uma função chamada *decoder* para converter a solução de chaves aleatórias em uma solução do problema de otimização. Assim, os desenvolvedores precisam implementar apenas o *decoder* específico para o problema estudado e configurar os parâmetros do método.

Uma importante propriedade das metaheurísticas, no entanto, é a influência dos parâmetros sobre a eficiência e eficácia da busca pelo espaço de soluções. Esses parâmetros devem ser configurados em qualquer metaheurística. No entanto, os valores não são ótimos para todos os problemas ou instâncias, ou não são adequados em termos de tempo computacional disponível para as execuções. Assim, deve ser realizada uma configuração eficiente dos valores dos parâmetros utilizados em cada aplicação.

Eiben et al. [2007] apresentam uma classificação de técnicas de configuração de parâmetros para algoritmos evolutivos. Os valores dos parâmetros podem ser definidos antes da execução e permanecem fixos ao longo da busca (configuração *off-line*), ou os valores dos parâmetros podem ser modificados dinamicamente durante a execução (configuração *on-line*).

A execução de uma metaheurística é um processo dinâmico e adaptativo em que diferentes valores de parâmetros podem ser ótimos em diferentes estágios da busca. Portanto, o controle de parâmetros *on-line* pode ser usado para balancear diversificação e intensificação. Os valores dos parâmetros são modificados por regras determinísticas (por exemplo, variando ao longo do número de iterações) ou adaptados por informações retornadas pela busca (por exemplo, a qualidade da solução encontrada). Os parâmetros também podem ser incluídos na representação da solução e devem ser evoluídos durante o processo de busca (controle auto-adaptativo). Essas abordagens não diminuem necessariamente o número de parâmetros, no entanto, tornam a configuração de parâmetros mais simples.

Alguns exemplos de algoritmos evolutivos adaptativos são *Genetic Algorithm with Varying Population Size* (GAVaPS) [Arabas et al., 1994], *Self-Adaptive Genetic Algorithm* (SAGA) [Hinterding et al., 1996], *Parameter-less GA* [Harik e Lobo, 1999], *Population Resizing on Fitness Improvement GA* (PRoFIGA) [Eiben et al., 2004] e *Population Training Algorithm* (PTA) [Oliveira e Lorena, 2005], entre outros.

Neste artigo foi proposto um BRKGA adaptativo (A-BRKGA) para produzir uma metaheurística eficiente e geral na resolução de problemas sem que haja parâmetros visíveis externamente. Para analisar o comportamento do A-BRKGA foram resolvidos três problemas de otimização combinatória: Problema de Agrupamento Centrado Capacitado (CCCP, do inglês *Capacitated Centered Clustering Problem*) [Negreiros e Palhano, 2006], Problema de Empacotamento (BPP, do inglês *bin packing problem*) [Scholl et al., 1997], e Problema de Alocação Tática de Berços (TBAP, do inglês *Tactical Berth Allocation Problem*) [Giallombardo et al., 2010]. Experimentos computacionais realizados em instâncias disponíveis na literatura comprovam que a abordagem proposta é competitiva em termos de qualidade de solução e tempo de execução.

O restante do artigo está organizado da seguinte forma. A seção 2 apresenta o BRKGA e a seção 3 descreve as ideias básicas do A-BRKGA e os conceitos de controle de parâmetros. Na seção 4 descreve-se o A-BRKGA aplicado aos problemas estudados. A seção 5 apresenta as conclusões e trabalhos futuros.

https://proceedings.science/p/85286?lang=pt-br

2. BRKGA

O BRKGA é uma metaheurística proposta por Goncalves e Resende [2011], na qual as soluções são codificadas como vetores com n elementos de números reais no intervalo $[0, 1]$. Esses números são chamados chaves aleatórias.

Os usuários precisam implementar apenas uma função chamada *decoder*, a qual tem como entrada um vetor de chaves aleatórias que é convertido em uma solução do problema de otimização com um valor de função objetivo (*fitness*).

O BRKGA evolui uma população de p vetores de chaves aleatórias com tamanho n até um número máximo de gerações (max_{gen}). Cada indivíduo da população inicial é gerado de forma aleatória e independente com genes do cromossomo no intervalo $[0, 1]$. A função *decoder* calcula a aptidão de cada indivíduo. Assim, em cada geração a população é dividida em dois grupos de indivíduos: um grupo com os p_e melhores indivíduos (elite) e o conjunto restante de $p - p_e$ indivíduos (não elite).

Uma nova geração de indivíduos é produzida para evoluir a população. Todos os indivíduos da elite são copiados sem modificação para a próxima população. O BRKGA não usa um operador de mutação, em vez disso, cria-se p_m novos indivíduos gerados da mesma maneira que os indivíduos na população inicial. Com p_e indivíduos da elite e p_m mutantes contabilizados na nova população, $p - p_e - p_m$ indivíduos são necessários para produzir os p indivíduos que compõem a população de uma nova geração. O operador de *crossover* é responsável pela produção dos filhos, combinando pares de indivíduos da elite e da não elite.

O *parameterized uniform crossover* [Spears e Jong, 1991] é usado no BRKGA. Seja ρ_e a probabilidade de um filho herdar um gene (elemento do vetor) de seu pai elite. Cada elemento i do filho assume o valor do i-ésimo elemento do pai elite com probabilidade ρ_e, caso contrário recebe o valor do i-ésimo elemento do pai não-elite.

Gonçalves e Resende [2016] recomendam valores para os parâmetros do BRKGA e Prasetyo et al. [2015] realizam uma pesquisa acerca de aplicações do BRKGA na literatura, mostrando que esses parâmetros recomendados são válidos e utilizados nos estudos existentes. A Tabela 1 apresenta os parâmetros do BRKGA e os intervalos recomendados.

Tabela 1: Parâmetros e intervalos de valores recomendados para o BRKGA.

Parâmetro	Intervalo
p: tamanho da população	$\kappa_p * n \approx [100, 1000]$
p_e: tamanho da partição elite	$p_e = max\{3, \kappa_e * p\}$, sendo $\kappa_e \in [0, 10, 0, 25]$
p_m: tamanho da partição mutante	$p_m = max\{1, \kappa_m * p\}$, sendo $\kappa_m \in [0, 05, 0, 20]$
ρ_e: probabilidade de herdar o i-ésimo gene do pai elite	$0, 65 \leq \rho_e \leq 0, 80$
max_{gen}: número máximo de gerações	$[50, 500]$

O BRKGA, como outras metaheurísticas, não possuem valores ótimos de parâmetros. Assim, algumas questões precisam ser consideradas, como a população elite, que precisa ter um número grande de soluções com qualidade e diversidade. Esta questão e o valor do parâmetro ρ_e são características importantes para evitar uma convergência prematura. O tamanho da partição de mutantes é usado para evitar que o método fique preso em ótimos locais. Apesar destes intervalos recomendados para os parâmetros, definir a melhor configuração para uma aplicação específica não é uma tarefa simples.

3. A-BRKGA

A motivação do BRKGA adaptativo (A-BRKGA) é realizar um ajuste de parâmetros para que haja um balanceamento entre diversificação e intensificação ao longo do processo evolutivo, de tal forma a realizar uma exploração adequada do espaço de soluções.

https://proceedings.science/p/85286?lang=pt-br

A Figura 1 ilustra o processo evolutivo do A-BRKGA proposto. O componente destacado em cinza representa a função *decoder*, que converte os vetores de chaves aleatórias em soluções do problema e calcula a função objetivo (*fitness*). Este componente é o único relacionado com o problema e os usuários do A-BRKGA também precisam implementar para cada aplicação. Os outros componentes são independentes e podem ser reutilizados.

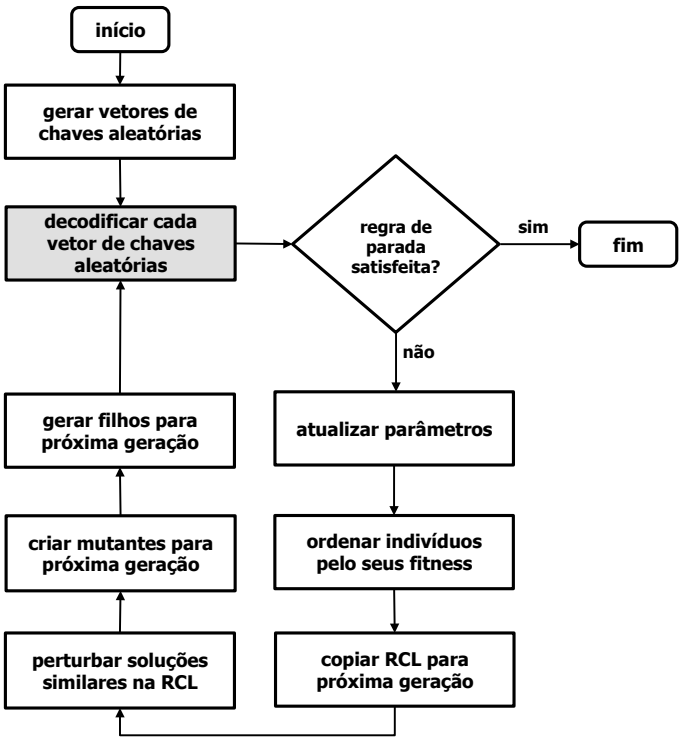

Figura 1: Fluxograma do A-BRKGA.

A população inicial do A-BRKGA é gerada com p vetores de chaves aleatórias (indivíduos). Cada cromossomo de um indivíduo tem $n + 2$ genes gerados aleatoriamente. O valor de n é definido de acordo com o problema que está sendo resolvido, por exemplo, número de itens no problema da mochila. Os dois elementos extras inseridos no cromossomo, posições $n+1$ e $n+2$ do vetor, são usados para calcular os parâmetros auto-adaptativos ρ_e e β, respectivamente. Este último foi proposto neste trabalho para determina a intensidade da perturbação em indivíduos similares.

Cada geração do A-BRKGA consiste em seis etapas:

(1) Os parâmetros são atualizados usando regras determinísticas que levam em consideração o progresso do processo evolutivo. Há dois parâmetros auto-adaptativos que são atualizados pelo próprio processo evolutivo, uma vez que estão incluídos nos cromossomos. A configuração específica de cada parâmetro será discutida nas próximas subseções.

(2) Os cromossomos da população são classificados de modo que os melhores indivíduos estão no topo e os piores na parte inferior.

(3) A população é dividida em dois grupos, sendo que os melhores indivíduos são mantidos em uma lista restrita de cromossomos (RCL). Inspirado no *Reactive* GRASP [Prais e Ribeiro, 2000], o RCL é formado por indivíduos cujo *fitness* é menor do que um valor limitante (f_e) definido pelo parâmetro $\alpha \in [0, 1]$:

$$f_e = f_{min} + \alpha(f_{max} - f_{min}) \quad (1)$$

https://proceedings.science/p/85286?lang=pt-br

sendo f_{min} e f_{max}, respectivamente, os valores de *fitness* do melhor e pior indivíduos na população atual. O caso $\alpha = 0$ corresponde à adição de indivíduos somente com o melhor *fitness*, enquanto que $\alpha = 1$ é equivalente a adicionar todos os indivíduos na RCL. O valor do parâmetro α é modificado periodicamente de acordo com o progresso da busca. O número máximo de indivíduos na RCL, no entanto, é limitado por p_e. Finalmente, os indivíduos na RCL são copiados para a próxima população.

(4) Realizar uma perturbação de indivíduos semelhantes para controlar a diversificação na RCL. Por simplicidade, a semelhança entre indivíduos é definida apenas por valores de *fitness* iguais. As perturbações permitem que o A-BRKGA escape de ótimos locais e evita a desvantagem de adicionar apenas indivíduos aleatórios. O A-BRKGA utiliza uma série de trocas aleatórias no cromossomo e a intensidade da perturbação é definida pelo parâmetro auto-adaptativo $\beta \in [0,001, 0,01]$ que é a probabilidade de trocar dois valores de chaves aleatórias. Para cada indivíduo similar, uma número aleatório é gerado para cada gene do cromossomo para determinar se a chave aleatória será perturbada. Essa probabilidade deve ser baixa, caso contrário, a perturbação se transformará em um gerador de soluções aleatórias. O parâmetro β foi incluído na posição $n + 2$ do cromossomo e esta evolui junto com os outros genes associados ao problema.

(5) Gerar aleatoriamente p_m indivíduos mutantes com o mesmo procedimento usado para criar a população inicial.

(6) Combinar $p - |RCL| - p_m$ pares de pais, sendo um escolhido aleatoriamente da RCL e outro escolhido aleatoriamente do restante da população. O *parameterized uniform crossover* é realizado gerando números aleatórios r_j no intervalo $[0, 1]$ para cada gene do cromossomo. Para cada j, se $r_j \leq \rho_e$, então o j-ésimo gene do cromossomo herda o j-ésimo alelo do pai RCL. Caso contrário, ele herda o alelo do outro pai. O valor do parâmetro ρ_e é auto-adaptativo e está incluído na posição $n + 1$ do cromossomo. O valor ρ_e é sempre selecionado do pai não-RCL.

O critério de parada do A-BRKGA é o número máximo de gerações que é calculado com base no parâmetro γ. Este parâmetro também é usado para atualizar o tamanho da população em cada geração. O valor de γ é uma escolha do usuário que leva em consideração o tempo de execução disponível para o algoritmo.

Nas próximas subseções será descrito como os parâmetros são controlados *on-line* durante o processo de busca para aumentar a diversidade populacional e evitar uma convergência prematura.

3.1. Tamanho da população e número máximo de gerações

Lobo e Lima [2007] argumentam que o tamanho da população deve ser configurado proporcionalmente ao tamanho e dificuldade do problema. Mas não é fácil determinar estas questões em problemas reais. Além disso, diferentes tamanhos podem ser ótimos em diferentes fases de um processo de busca. Por exemplo, pode ser interessante ter uma população maior no início do processo, a fim de fazer uma exploração do espaço de busca e encontrar boas regiões. O tamanho da população pode diminuir para refinamento das soluções durante as gerações restantes.

Em Prasetyo et al. [2015] o tamanho da população usado no BRKGA é geralmente entre 100 e 1000 indivíduos (ou $k_p * n$). No entanto, para uma população com até 1000 indivíduos é comum encontrar soluções de qualidade.

Portanto, propomos utilizar um tamanho de população máximo e mínimo ($p_{max} = 1000$ e $p_{min} = 100$) e o controle é baseado no processo de resfriamento de temperatura do *Simulated Annealing* [Kirkpatrick et al., 1983]. O A-BRKGA começa com o tamanho máximo da população e em cada geração o tamanho da população é diminuído pelo parâmetro $\gamma < 1$ ($p^{k+1} = p^k * \gamma$). Os piores indivíduos são eliminados neste processo.

https://proceedings.science/p/85286?lang=pt-br

Estes parâmetros também são usados para calcular o critério de parada do A-BRKGA. Inspirado em uma fórmula para calcular o número máximo de iterações necessárias para uma dada temperatura do *Simulated Annealing* [Ingber, 1993], o número máximo de gerações (max_{gen}, critério de parada do A-BRKGA) é definido pela Equação 2.

$$max_{gen} = \gamma^{\left(log_{\gamma}^{p_{min}} - p_{max}\right)} \tag{2}$$

O parâmetro γ precisa ser configurado pelo usuário, no entanto, é mais simples de calibrar. Nós definimos apenas três valores para o parâmetro $\gamma \in \{0,999, 0,998, 0,997\}$ que é escolhido pelos usuários de acordo com o tempo computacional disponível e a qualidade da solução desejada. Todos os valores resultam em uma diminuição lenta do tamanho da população. A principal diferença é o número máximo de gerações e, consequentemente, é possível ter diferentes tempos de execução e espaços de soluções.

Um procedimento de reinicialização é realizado quando o tamanho da população é inferior a p_{min} indivíduos. Apenas indivíduos da RCL são mantidos na população atual e $p_{max} - |RCL|$ novos indivíduos são gerados aleatoriamente.

3.2. Partições elite e mutantes

Os parâmetros $\kappa_e \in [0,10, 0,25]$ e $\kappa_m \in [0,05, 0,20]$ são usados para definir as partições elite (p_e) e mutantes (p_m) da população. Novamente, objetiva-se que o processo de busca seja diversificado na fase inicial e realize mais intensificação à medida que a população evolua.

As Equações 3 e 4 definem, respectivamente, o controle dos parâmetros κ_e e κ_m, sendo que g_k é o número da geração atual. Assim, a partição elite máxima tem poucos indivíduos nas gerações iniciais quando a qualidade média dos indivíduos é baixa e esse número aumenta nas gerações seguintes. Por outro lado, o número de mutantes diminui ao longo do processo de busca.

$$\kappa_e = 0,10 + \frac{g_k}{max_{gen}} * (0,25 - 0,10) \tag{3}$$

$$\kappa_m = 0,05 + \left(1 - \frac{g_k}{max_{gen}}\right) * (0,20 - 0,05) \tag{4}$$

O parâmetro α define o número de indivíduos elite na RCL (Equação 1). O valor de α é iniciado com $0,3$ e é diminuído a cada geração pela Equação 5. O valor mínimo de α é $0,1$. Este intervalo é definido devido às soluções de baixa qualidade nas gerações iniciais e à tendência de achatamento da população.

$$\alpha = 0,10 + \left(1 - \frac{g_k}{max_{gen}}\right) * (0,30 - 0,10) \tag{5}$$

A ideia é incluir soluções com qualidade na RCL. No entanto, a diversidade neste grupo também é importante. Com esse objetivo, os indivíduos do A-BRKGA com valores iguais de *fitness* são perturbados com uma intensidade definida por β que é um parâmetro auto-adaptativo. Esse valor é calculado pela Equação 6 e evolui dentro do cromossomo x na posição $n + 2$.

$$\beta = 0,001 + x_{n+2} * (0,10 - 0,001) \tag{6}$$

3.3. Crossover

O *parameterized uniform crossover* deve privilegiar a herança de alelos de cromossomos da RCL. O parâmetro ρ_e define a probabilidade de um filho herdar um elemento do vetor de chaves aleatórias do pai RCL e também é auto-adaptativo no A-BRKGA. O parâmetro ρ_e é calculado usando a Equação 7 e a chave aleatória $n + 1$ do cromossomo não-RCL y.

$$\rho_e = 0,65 + y_{n+1} * (0,80 - 0,65) \tag{7}$$

https://proceedings.science/p/85286?lang=pt-br

4. Experimentos Computacionais

Com objetivo de avaliar a eficiência e robustez do A-BRKGA, foram realizadas três aplicações do método à problemas de otimização combinatória pertencentes a classe NP-difícil: Problema de Agrupamento Centrado Capacitado (CCCP), Problema de Empacotamento (BPP), e Problema de Alocação Tática de Berços (TBAP). A seguir são apresentadas as funções *decoder* específicas para cada problema, bem como um resumo dos resultados computacionais obtidos pelo A-BRKGA.

Além do A-BRKGA foi testada uma versão clássica do BRKGA que teve os parâmetros configurados por meio do *software* de configuração *off-line* CALIBRA [Adenso-Diaz e Laguna, 2006]. Todos os métodos foram implementados na linguagem C++ e os testes computacionais foram realizados em um PC com processador Intel Core i7 3.4 GHz e 16GB de RAM. Para os problemas CCCP e TBAP cada instância foi resolvida 30 vezes por cada método usando sementes diferentes. No problema BPP cada instância foi resolvida 100 vezes com sementes diferentes.

4.1. A-BRKGA aplicado ao CCCP

O problema de agrupamento centrado capacitado (CCCP, do inglês *Capacitated Centered Clustering Problem*), proposto por Negreiros e Palhano [2006], é uma generalização do p-medianas [Hakimi, 1964]. O CCCP consiste em definir um conjunto de m agrupamentos com capacidade limitada e mínima dissimilaridade entre os agrupamentos formados. Cada agrupamento possui um centróide localizado no centro geométrico de seus pontos cobrindo as demandas dos pontos alocados neste agrupamento. A principal diferença para o p-medianas é que os agrupamentos possuem capacidades e estão centrados na média das coordenadas de seus pontos, enquanto que no p-medianas os agrupamentos estão centrados em pontos definidos como medianas.

Uma solução do CCCP é codificada como um vetor de n chaves aleatórias, sendo que cada chave aleatória está associada a um ponto. A função *decoder* baseia-se na heurística construtiva *Best-Fit* [Muritiba et al., 2012].

A decodificação das n chaves aleatórias de cada solução em uma sequência de pontos é realizada ordenando os pontos em ordem crescente de seus valores de chaves aleatórias correspondentes. Os primeiros m pontos são atribuídos a m agrupamentos como centróides. Então, os $n - m$ pontos restantes são atribuídos ao agrupamento mais próximo com capacidade disponível. O centróide sempre é atualizado após um novo ponto ser adicionado no agrupamento.

O *fitness* de uma solução com todos os pontos alocados é calculada como a soma da distância euclidiana entre cada centróide dos agrupamentos e seus pontos alocados. Observe que a posição geométrica do centróide depende dos pontos que compõem o agrupamento. Portanto, a posição dos centroídes deve ser calculada antes que as distâncias sejam computadas. O objetivo é minimizar a distância total.

Os experimentos computacionais foram realizados com 25 instâncias disponíveis na literatura (http://people.brunel.ac.uk/ mastjjb/jeb/info.html). A Tabela 2 apresenta uma comparação do A-BRKGA com os melhores métodos encontrados na literatura para resolver o CCCP: *Variable Neighboorhod Search* (VNS) [Negreiros e Palhano, 2006], *Clustering Search* (CS) [Chaves e Lorena, 2011] e *Tabu Search* (TS) [Muritiba et al., 2012].

Tabela 2: CCCP: Comparação entre A-BRKGA e os melhores métodos da literatura considerando número de instâncias para as quais os métodos obtêm a melhor solução conhecida.

Grupo	VNS [Negreiros e Palhano, 2006]	CS [Chaves e Lorena, 2011]	TS [Muritiba et al., 2012]	A-BRKGA
ta	3 / 7	7 / 7	7 / 7	7 / 7
sjc	0 / 6	3 / 6	2 / 6	6 / 6
doni	1 / 7	0 / 7	4 / 7	2 / 7
p3038	0 / 5	0 / 5	3 / 5	2 / 5
	4 / 25	10 / 25	16 / 25	17 / 25

https://proceedings.science/p/85286?lang=pt-br

O A-BRKGA encontra as melhores solução conhecidas em todas as instâncias dos grupos *ta* e *sjc*. Em sete instâncias o A-BRKGA obteve soluções melhores que a literatura: *sjc3a*, *sjc3b*, *sjc4b*, *doni1*, *doni6*, *p3038-600* e *p3038-700*. O método também se mostrou robusto, com um desvio padrão máximo de 0,98%. Os melhores resultados conhecidos foram igualados ou melhorados em 17 das 25 instâncias do CCCP.

O A-BRKGA também foi comparado com o BRKGA configurado com CALIBRA em termos de qualidade da solução, robustez e tempo computacional. Os desvios padrão médios do A-BRKGA e BRKGA foram de 1,07 e 1,11, respectivamente. Em relação ao tempo computacional, o A-BRKGA é mais rápido que o BRKGA devido ao mecanismo de resfriamento do tamanho da população, que diminui o número de indivíduos ao longo do processo de busca.

O teste estatístico Wilcoxon (WSR) foi aplicado para comparar os conjuntos de soluções e investigar se existe uma diferença significativa entre as soluções do A-BRKGA e BRKGA. O WSR indicou que os resultados do A-BRKGA foram estatisticamente significativos em relação ao BRKGA ($Z = 11165$, *p-value*$= 1.358e-06$). Este resultado sugere que o uso do A-BRKGA neste conjunto de instâncias melhora os resultados encontrados em comparação com o BRKGA.

4.1.1. A-BRKGA aplicado ao TBAP

Muitos problemas estudados no setor portuário são classificados como *NP-difícil*, tais como, problema de alocação de berços (BAP, do inglês *Berth Allocation Problem*) e problema de alocação de guindastes (QCAP, do inglês *Quay Crane Allocation Problem*). O BAP consiste em alocar navios em berços, de forma que seja minimizado o tempo de serviço dos navios. As decisões a serem tomadas dizem respeito a posição e ao instante em que o navio deverá atracar [Imai et al., 2001]. O QCAP recebeu pouca atenção na literatura, ganhando mais visibilidade quando proposto juntamente com o BAP, criando assim o problema tático de alocação de berços (TBAP, do inglês *Tactical Berth Allocation Problem*).

Assim, o TBAP busca determinar o berço, o tempo de operação e a alocação de guindastes através de perfis, para cada navio que chega ao porto com horário previsto [Giallombardo et al., 2010]. O tempo de operação de cada navio nos berços é definido através do perfil utilizado para o atendimento. Cada perfil define a quantidade de guindastes associada a cada 2 horas de operação. Quanto mais guindastes por 2 horas se utiliza, maior o custo da operação, e com isso, mais rápido o navio é atendido. Cada navio tem um conjunto de perfis associados a ele. Todos os perfis têm um custo associado ao número de guindastes utilizado e navio atendido. Portanto, um mesmo perfil pode ter custo diferente devido a estar relacionado à diferentes navios.

Quando um navio chega no porto para ser atendido, alguns dos contêineres que ele carregará podem ainda não ter chegado ou estar em outro berço e isto deve ser levado em consideração. Cada navio traz a informação do fluxo de carga que os outros navios irão transportar. Assim, o custo de armazenamento é proporcional à carga armazenada, e dependendo de qual navio receberá a carga, um custo relacionado ao transporte entre navios é contabilizado.

O *decoder* para o TBAP utiliza um vetor de chaves aleatórias de tamanho $n = 2 * N$ (sendo N o número de navios do TBAP). A primeira metade do vetor (posições 1 a N) representa os navios, e a segunda metade (posições $N + 1$ a $2N$) representa o perfil utilizado pelos navios. A primeira parte da subsequência é ordenada de forma crescente, definindo a sequência de navios que terá prioridade no atendimento. A segunda subsequência divide o intervalo $[0, 1]$ com o número de possibilidade de perfis para cada navio, sendo cada intervalo representando um perfil de guindaste.

O *decoder* inicializa definindo a sequência de navios que serão atendidos e alocando os perfis de guindastes para cada navio. Em seguida utiliza-se uma heurística para associar os navios aos berços baseado na *berthing positioning heuristic* [Lalla-Ruiz et al., 2014]. Para cada navio é calculado o custo de atribuir o navio em cada berço considerando o custo de atendimento do navio e os custos de transferências para os navios já alocados no berço. A alocação de um navio no berço deve satisfazer a janela de tempo do navio e o número de guindastes disponíveis em cada unidade de tempo.

https://proceedings.science/p/85286?lang=pt-br

O *fitness* de uma solução é o somatório dos custos dos perfis alocados aos navios subtraído dos custos de transferência de cargas.

Para verificar a eficiência do A-BRKGA foram utilizadas as instâncias de Giallombardo et al. [2010], Vacca et al. [2013] e Lalla-Ruiz et al. [2014]. Os resultados obtidos foram comparados com os resultados do BRKGA [Lalla-Ruiz et al., 2014], *Tabu Search*[Giallombardo et al., 2010] e com os limitantes superiores do CPLEX e *Branch&Bound* (B&P) [Vacca et al., 2013].

A Tabela 3 apresenta o número de instâncias para as quais os métodos encontraram a melhor solução conhecida na literatura. O método A-BRKGA encontrou a melhor solução conhecida em 108 das 125 instâncias testadas, sendo que em 72 instâncias (58%) houve melhora em relação à melhor solução obtida pelos métodos da literatura.

Tabela 3: TBAP: Comparação entre A-BRKGA e os melhores métodos da literatura considerando o número de instâncias para as quais os métodos obtêm a melhor solução conhecida.

Grupo	CPLEX	B&P [Vacca et al., 2013]	TS-BP [Giallombardo et al., 2010]	BRKGA [Lalla-Ruiz et al., 2014]	A-BRKGA
Vacca	2 / 20	16 / 20	-	19 / 20	20 / 20
Giallombardo	0 / 60	-	0 / 60	1 / 60	57 / 60
Lalla-Ruiz	12 / 45	-	-	28 / 45	31 / 45
	13 / 124	-	-	48 / 125	108 / 125

A Tabela 4 apresenta uma comparação entre A-BRKGA, BRKGA configurado com o *software* CALIBRA e o BRKGA proposto por Lalla-Ruiz et al. [2014]. O tempo computacional do A-BRKGA é superior ao apresentado pelo BRKGA [Lalla-Ruiz et al., 2014], principalmente devido ao tamanho da população utilizado no BRKGA da literatura (100 indivíduos). Entretanto, o aumento da qualidade da solução justifica o tempo computacional mais elevado. Em relação ao BRKGA configurado com CALIBRA, o tempo computacional é equivalente. Apesar disso, a qualidade das soluções do A-BRKGA também é superior à qualidade apresentada pelo BRKGA com CALIBRA.

Tabela 4: TBAP: Comparação entre A-BRKGA, BRKGA CALIBRA e BRKGA [Lalla-Ruiz et al., 2014]

Grupo	#Instâncias	Métrica	BRKGA Lalla-Ruiz et al. [2014]	BRKGA Calibra	A-BRKGA
Vacca	20	Desvio	-	0,01	0,01
		Melhor	0,95	1,00	1,00
		Tempo	1,01	7,88	8,34
Giallombardo	60	Desvio	-	0,13	0,10
		Melhor	0,02	0,27	0,95
		Tempo	19,71	69,11	67,46
Lalla-Ruiz	45	Desvio	-	1,34	1,03
		Melhor	0,62	0,51	0,69
		Tempo	2,96	7,45	7,60

4.1.2. A-BRKGA aplicado ao BPP

O problema de empacotamento (BPP, do inglês *Bin Packing Problem*) é um problema de otimização combinatória da classe *NP-difícil*, no qual itens de diferentes tamanhos precisam ser empacotados em um número finito de *bins* (caixas) com capacidade limitada, de tal forma que o número de *bins* usados seja minimizado.

Considerando um conjunto de itens $I = \{1, 2, ..., n\}$ com pesos $w_i (i = 1, ...n)$ associados, o BPP consiste em encontrar o número mínimo de *bins*, de capacidade C, necessários para empacotar todos os itens sem violar nenhuma restrição de capacidade.

Uma solução do BPP é codificada como um vetor de n chaves aleatórias, sendo que cada chave aleatória está associada a um item. A função *decoder* baseia-se na heurística *First Fit Decreasing* [Quiroz-Castellanos et al., 2015]. As chaves aleatórias são ordenadas em ordem

https://proceedings.science/p/85286?lang=pt-br

crescente gerando uma ordem de preferências dos itens a serem inseridos nos *bins*. Em seguida, cada item é alocado no primeiro *bin* com capacidade disponível. Caso não haja *bin* disponível, um novo *bin* é criado.

O *fitness* de uma solução com todos os itens alocados é calculada pela Equação 8, sendo que m é o número de *bins* usados na solução, S_i é a soma dos pesos dos itens alocados no *bin* i, e C é a capacidade dos *bins*. Desta forma, há uma probabilidade menor de haver várias soluções com mesmo *fitness* no espaço de soluções. Entretanto, para comparar com outros métodos da literatura utilizamos apenas o resultado com menor número de *bins* obtidos para alocar todos os itens.

$$F_{BPP} = \frac{\sum_{i \in m}(S_i/C)^2}{m} \tag{8}$$

O A-BRKGA foi testado com 1615 instâncias disponíveis na literatura: dois conjuntos com 80 instâncias propostas por Falkenauer [1996], três conjuntos com 720, 480 e 10 instâncias, respectivamente, propostas por Scholl et al. [1997], dois conjuntos com 100 instâncias propostas por Schwerin e Wäscher [1998], um conjunto com 17 instâncias propostas por Wäscher e Gau [1996], e um conjunto com 28 instâncias difíceis propostas por Belov e Scheithauer [2006].

Os resultados computacionais do A-BRKGA são comparados com os resultados dos três melhores métodos heurísticos encontrados na literatura: Perturbation-SAWMBS (P-SAWMBS) [Fleszar e Charalambous, 2011], GGA-CGT [Quiroz-Castellanos et al., 2015] e CNS-BP [Buljubašić e Vasquez, 2016]. A Tabela 5 apresenta o número de instâncias para as quais os métodos encontraram a solução ótima. As soluções ótimas foram provadas por Brandao e Pedroso [2016].

Observa-se que o A-BRKGA foi capaz de encontrar mais soluções ótimas que os métodos P-SAWMBS e GGA-CGT, mas foi inferior ao CNS-BP que utiliza a metaheurística *Tabu Search* combinada com métodos de intensificação da busca. O tempo computacional do A-BRKGA foi equivalente ao tempo computacional do CNS-BP que utiliza um computador com configuração semelhante nos testes computacionais.

O conjunto hard28 [Belov e Scheithauer, 2006] possui as instâncias em que os métodos encontram maiores dificuldades para resolver. O A-BRKGA consegue encontrar o ótimo para 18 das 28 instâncias disponíveis, para as demais instâncias a solução do A-BRKGA é sempre um *bin* a mais que a solução ótima.

Tabela 5: BPP: Comparação entre A-BRKGA e os melhores métodos da literatura considerando o número de instâncias para as quais os métodos obtêm a melhor solução conhecida.

Grupo	P-SAWMBS	GGA-CGT	CNS-BP	A-BRKGA
Falkenauer	159 / 160	160 / 160	160 / 160	160 / 160
Scholl	1210 / 1210	1210 / 1210	1210 / 1210	1210 / 1210
Schwerin	200 / 200	200 / 200	200 / 200	200 / 200
Wascher	16 / 17	16 / 17	17 / 17	16 / 17
Hard28	5 / 28	16 / 25	25 / 28	18 / 28
	1590 / 1615	1602 / 1615	1612 / 1615	1604 / 1615

5. Conclusões e trabalhos futuros

Este artigo apresenta um método adaptativo baseado no BRKGA para resolver problemas de otimização combinatória. O BRKGA tem sido utilizado para resolver diversos problemas práticos e teóricos. No entanto, ainda não havia sido estudado o controle de parâmetros *on-line* no contexto do BRKGA.

A ideia principal do BRKGA adaptativo (A-BRKGA) é simplificar a fase de desenvolvimento de uma metaheurística para resolver qualquer problema com base nos conceitos de chaves aleatórias e controle de parâmetros durante a execução. O A-BRKGA oferece suporte aos usuários para reduzir o tempo de desenvolvimento. As regras determinísticas e auto-adaptativas são usadas para definir os parâmetros e permitir uma busca diversificada do espaço da solução.

https://proceedings.science/p/85286?lang=pt-br

Os experimentos computacionais consideraram três problemas de otimização classificados como NP-difícil e 1765 instâncias disponíveis na literatura. O A-BRKGA encontra a melhor solução conhecida em 1729 instâncias. Os resultados computacionais mostram que o A-BRKGA é tão eficiente e robusto como os métodos estado-de-arte para resolver os problemas estudados. A grande vantagem da abordagem A-BRKGA é ser simples de implementar, já que o processo evolutivo é reutilizado e os usuários não precisam se preocupar com o ajuste de parâmetros.

Para trabalhos futuros pretende-se desenvolver uma estrutura genérica para o A-BRKGA e aplicar este método a outros problemas de otimização combinatória.

Agradecimentos

Este trabalho é financiado pela Fundação de Amparo à Pesquisa do Estado de São Paulo - FAPESP (Processo nº 2016/07135-7) e pelo Conselho Nacional de Desenvolvimento Científico e Tecnológico - CNPq (Processo nº 307330/2015-0, Processo nº 304979/2012-0).

Referências

Adenso-Diaz, B. e Laguna, M. (2006). Fine-tuning of algorithms using fractional experimental designs and local search. *Operations research*, 54(1):99–114.

Arabas, J., Michalewicz, Z., e Mulawka, J. (1994). Gavaps-a genetic algorithm with varying population size. In *IEEE World Congress on Computational Intelligence.,*, p. 73–78.

Belov, G. e Scheithauer, G. (2006). A branch-and-cut-and-price algorithm for one-dimensional stock cutting and two-dimensional two-stage cutting. *EJOR*, 171(1):85–106.

Brandao, F. e Pedroso, J. P. (2016). Bin packing and related problems: general arc-flow formulation with graph compression. *Computers & Operations Research*, 69:56–67.

Buljubašić, M. e Vasquez, M. (2016). Consistent neighborhood search for one-dimensional bin packing and two-dimensional vector packing. *Computers & Operations Research*, 76:12–21.

Chaves, A. A. e Lorena, L. A. N. (2011). Hybrid evolutionary algorithm for the capacitated centered clustering problem. *Expert Systems with Applications*, 38(5):5013–5018.

Eiben, A. E., Michalewicz, Z., Schoenauer, M., e Smith, J. E. (2007). Parameter control in evolutionary algorithms. In *Parameter setting in evolutionary algorithms*, p. 19–46. Springer.

Eiben, A. E., Marchiori, E., e Valko, V. (2004). Evolutionary algorithms with on-the-fly population size adjustment. In *Inter. Conf. on Parallel Problem Solving from Nature*, p. 41–50. Springer.

Falkenauer, E. (1996). A hybrid grouping genetic algorithm for bin packing. *J. of Heur*, 2(1):5–30.

Fleszar, K. e Charalambous, C. (2011). Average-weight-controlled bin-oriented heuristics for the one-dimensional bin-packing problem. *European J. of Operational Research*, 210(2):176–184.

Giallombardo, G., Moccia, L., Salani, M., e Vacca, I. (2010). Modeling and solving the tactical berth allocation problem. *Transportation Research Part B*, 44(2):232–245.

Goncalves, J. F. e Resende, M. G. C. (2011). Biased random-key genetic algorithms for combinatorial optimization. *Journal of Heuristics*, 17(5):487–525.

Gonçalves, J. F. e Resende, M. G. C. (2016). *Random-Key Genetic Algorithms*, p. 1–13. Springer International Publishing, Cham. ISBN 978-3-319-07153-4.

Hakimi, S. (1964). Optimum location of switching centers and the absolute centers and the medians of a graph. *Operations Research*, 12:450–459.

https://proceedings.science/p/85286?lang=pt-br

Harik, G. R. e Lobo, F. G. (1999). A parameter-less genetic algorithm. In *Proceedings of the 1st Annual Conference on Genetic and Evolutionary Computation-Volume 1*, p. 258–265.

Hinterding, R., Michalewicz, Z., e Peachey, T. (1996). Self-adaptive genetic algorithm for numeric functions. *Parallel Problem Solving from Nature—PPSN IV*, p. 420–429.

Imai, A., Nishimura, E., e Papadimitrious, S. (2001). The dynamic berth allocation problem for a container port. *Transportation Research*, Part B,(35):401–417.

Ingber, L. (1993). Simulated annealing: Practice versus theory. *Mathematical and computer modelling*, 18(11):29–57.

Kirkpatrick, S., Gellat, D. C., e Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, 220:671–680.

Lalla-Ruiz, E., Velarde, J. L. G., Batista, B. M., e Moreno-Vega, J. M. (2014). Biased random key genetic algorithm for the tactical berth allocation problem. *Applied Soft Computing*, 22:60 – 76.

Lobo, F. G. e Lima, C. F. (2007). Adaptive population sizing schemes in genetic algorithms. *Parameter Setting in Evolutionary Algorithms*, 54:185–204.

Muritiba, A. E. F., Negreiros, M., Oriá, H. L. G., e de Souza, M. F. (2012). A tabu search algorithm for the capacitated centred clustering problem. *Simpósio Brasileiro de Pesquisa Operacional, Proceedings of SBPO/CLAIO*.

Negreiros, M. J. e Palhano, A. W. (2006). The capacitated centred clustering problem. *Computers and Operations Research*, 33(6):1639–1663.

Oliveira, A. e Lorena, L. A. (2005). Population training heuristics. In *Proceedings of the 5th European conference on Evolutionary Computation in Combinatorial Optimization*, p. 166–176. Springer-Verlag.

Prais, M. e Ribeiro, C. (2000). Reactive grasp an application to a matrix decomposition problem in tdma traffic assignment. *INFORMS Journal on Computing*, 12(3):164–176.

Prasetyo, H., Fauza, G., Amer, Y., e Lee, S. (2015). Survey on applications of biased-random key genetic algorithms for solving optimization problems. In *Industrial Engineering and Engineering Management (IEEM), 2015 IEEE International Conference on*, p. 863–870. IEEE.

Quiroz-Castellanos, M., Cruz-Reyes, L., Torres-Jimenez, J., Gómez, C., Huacuja, H. J. F., e Alvim, A. C. (2015). A grouping genetic algorithm with controlled gene transmission for the bin packing problem. *Computers & Operations Research*, 55:52–64.

Scholl, A., Klein, R., e Jürgens, C. (1997). Bison: A fast hybrid procedure for exactly solving the one-dimensional bin packing problem. *Computers & Operations Research*, 24(7):627 – 645.

Schwerin, P. e Wäscher, G. (1998). *A new lower bound for the bin-packing problem and its integration into MTP*. Martin-Luther-Univ. Halle-Wittenberg, Wirtschaftswiss. Fak.

Spears, W. M. e Jong, K. A. D. (1991). On the virtues of parameterized uniform crossover. *Proc. of the Fourth International Conference on Genetic Algorithms*, p. 230–236.

Vacca, I., Salani, M., e Bierlaire, M. (2013). An exact algorithm for the integrated planning of berth allocation and quay crane assignment. *Transportation Science*, 47(2):148–161. ISSN 1526-5447.

Wäscher, G. e Gau, T. (1996). Heuristics for the integer one-dimensional cutting stock problem: A computational study. *Operations-Research-Spektrum*, 18(3):131–144.

https://proceedings.science/p/85286?lang=pt-br