

BRKGA para o Problema de Corte de Estoque Unidimensional com *setups*

Eduardo Machado Silva, Antônio Augusto Chaves

Universidade Federal de São Paulo - ICT

Av. Cesare Monsueto Giulio Lattes, 1201 - Eugênio de Melo, São José dos Campos - SP,
12247-014

`machado.silva@unesp.br`, `antonio.chaves@unifesp.br`

Silvio Alexandre de Araujo

Universidade Estadual Paulista- IBILCE

R. Cristóvão Colombo, 2265 - Jardim Nazareth, São José do Rio Preto - SP, 15054-000

`silvio.araujo@unesp.br`

RESUMO

O problema de corte de estoque unidimensional consiste em cortar objetos disponíveis em estoque para atender a uma demanda de itens menores. Quando há necessidade de reposicionamento das facas durante o processo, surgem custos adicionais de *setup*, tornando desejáveis soluções com menor número de *setups*. O *Biased Random-Key Genetic Algorithm* (BRKGA) é uma metaheurística que realiza buscas em um hipercubo, onde cada ponto é mapeado para uma solução viável do problema por meio de um procedimento determinístico chamado *decoder*. Este trabalho propõe um novo *decoder* voltado para o problema de corte com *setups*, cuja eficácia é avaliada por meio de testes em dois grupos de instâncias da literatura: um baseado em dados reais e outro em instâncias aleatórias. Os resultados demonstram que o método proposto é eficiente na redução do número de *setups*, especialmente em cenários com altos custos de *setup*, superando, nesses casos, os resultados do estado da arte.

PALAVRAS CHAVE. *Biased Random-Key Genetic Algorithm*, Problemas de Corte, *setups*.

Metaheurísticas. Otimização Discreta.

ABSTRACT

The one-dimensional cutting stock problem consists of cutting stock objects into smaller items to meet a given demand. When knife repositioning is required during the process, additional setup costs are incurred. Solutions with fewer setups are desirable. The Biased Random-Key Genetic Algorithm (BRKGA) is a metaheuristic that performs searches in a hypercube, where each point is mapped to a feasible solution of the problem through a deterministic procedure called a decoder. This work proposes a new decoder tailored to the cutting problem with setups, whose effectiveness is evaluated using two groups of benchmark instances: one based on real-world data and the other on randomly generated instances. The results indicate that the proposed method is effective in minimizing the number of *setups*, particularly in scenarios characterized by high *setup* costs, where it outperforms existing state-of-the-art approaches.

KEYWORDS. *Biased Random-Key Genetic Algorithm*, Cutting Stock Problem, *setups*.

Metaheuristics. Discrete Optimization.

1. Introdução

O problema clássico de corte de estoque unidimensional (PCE) consiste em cortar barras de comprimento fixo L para produzir n tipos de itens, cada um com comprimento l_i e demanda d_i ($i = 1, \dots, n$), de modo a atender completamente essa demanda utilizando a menor quantidade possível de material. A forma como um objeto é cortado, gerando um subconjunto de itens, é denominada padrão de corte. A solução do problema consiste em um conjunto de padrões de corte e suas respectivas frequências de uso, denominado plano de corte. Problemas de corte são comuns em industriais de papel, barra metálica, roupas e vidro [Suliman, 2012; Henn e Wäscher, 2013]. Em geral, o objetivo principal do CSP é minimizar os custos decorrentes das perdas de material no processo de corte. No entanto, em aplicações práticas, podem surgir custos adicionais [Wuttke e Heese, 2018; Cherri, A. C. et al., 2014]. Um desses custos está relacionado à interrupção da produção e/ou à troca de facas para a execução de novos padrões, caracterizando um *setup*. Dessa forma, planos de corte com poucos padrões são preferíveis, especialmente quando os custos de *setup* são elevados.

O problema de corte de estoque (PCE) que tem como objetivo exclusivo a minimização do número de padrões de corte é conhecido como o Problema da Minimização de Padrões (PMP) [Vanderbeck, 2000]. Segundo McDiarmid [McDiarmid, 1999], esse problema pertence à classe dos problemas NP-difíceis. A partir deste ponto, o PCE com minimização de padrões será denotado por PCEs. Uma revisão abrangente sobre classificações e extensões do PCE pode ser encontrada em [Henn e Wäscher, 2013].

Diversas técnicas para resolver o PCEs tem sido propostas na literatura, variando desde modelos exatos [Vanderbeck, 2000; Martin et al., 2022], passando por heurísticas e metaheurísticas [Haessler, 1975; de Araujo et al., 2014], até métodos híbridos que combinam modelos e heurísticas [Yanasse e Limeira, 2006; Cui et al., 2013, 2015; Mobasher e Ekici, 2013]. Na implementação de metaheurísticas, um dos principais desafios é a calibração adequada dos parâmetros, visto que isso impacta diretamente a eficácia da busca. Em [Chaves e Lorena, 2021], foi proposta uma estratégia de adaptação de parâmetros baseada em aprendizado por reforço, em particular o método *Q-learning*, aplicada ao *Biased Random-Key Genetic Algorithm* (BRKGA), originalmente proposto por [Gonçalves e Resende, 2011].

Este trabalho tem como objetivo desenvolver um BRKGA com *Q-learning* (BRKGA-QL) para resolver o PCEs. Para a implementação do BRKGA, é necessário um procedimento chamado *decoder*. Assim, este trabalho propõe um *decoder* para o PCEs que gera padrões de corte de forma independente de modelos matemáticos e, em seguida, calcula suas frequências para compor o plano de corte. Os resultados obtidos foram comparados com os métodos propostos em [Cui et al., 2015, 2013] e [Yanasse e Limeira, 2006], utilizando dois grupos de instâncias: um conjunto gerado aleatoriamente pelo método CUTGEN1 [Gau e Wäscher, 1995] e outro composto por instâncias práticas apresentadas em [Umetani et al., 2003].

O conteúdo deste artigo está organizado da seguinte forma: a Seção 2 apresenta uma breve revisão da literatura sobre métodos de solução para o PCEs. A Seção 3 descreve o modelo matemático do problema. A metodologia proposta é detalhada na Seção 4, enquanto os resultados computacionais são discutidos na Seção 5. Por fim, as conclusões e propostas para trabalhos futuros são apresentadas na Seção 6.

2. Revisão de literatura

A maioria dos trabalhos envolvendo o PCEs ou o PMP concentra-se em heurísticas híbridas que combinam algum tipo de geração de colunas. O primeiro trabalho a propor um método exato para o PMP foi apresentado por [Vanderbeck, 2000], no qual o problema foi modelado por meio de uma formulação quadrática. O método foi testado em 16 instâncias práticas, das quais apenas 12 foram resolvidas até a otimalidade dentro do tempo limite proposto pelo autor. Em [Martin

et al., 2022], foi apresentada uma abordagem exata biobjetivo para o PCEs, baseada em uma formulação matemática inspirada no modelo de [Vanderbeck, 2000]. Um modelo alternativo foi proposto para fins de comparação, e os resultados foram comparados com os de [Umetani et al., 2006].

A primeira heurística proposta para o PCEs foi apresentada por [Haessler, 1975]. Os autores propuseram o método SHP, do inglês *Sequential Heuristic Procedure*, que consiste na seleção de padrões de corte com base nos requisitos de produção previamente definidos para cada item. O SHP é motivado pela observação de que aumentar a frequência de determinados padrões de corte pode ser eficaz na redução do número de *setups*. Em [Foerster e Wäscher, 2000], planos de corte reduzidos para o PCEs são obtidos em dois estágios: no primeiro, gera-se um plano inicial de corte; no segundo, aplica-se um procedimento iterativo que combina dois padrões de corte em um, três padrões em dois, e quatro padrões em três, repetindo-se o processo até que nenhuma combinação adicional consiga reduzir o número total de padrões.

Uma metaheurística foi proposta para o PCEs em [Umetani et al., 2003], na qual os autores desenvolveram uma ILS, do inglês *Iterated Local Search*, para resolver o PCEs com um número fixo de padrões de corte. Em [Umetani et al., 2006], a ILS foi aprimorada com a introdução de dois tipos distintos de vizinhança. Além disso, algoritmos genéticos foram propostos por [Golfeto et al., 2009] e [de Araujo et al., 2014] como alternativas para resolver o PCEs como um problema multi-objetivo.

Em [Yanasse e Limeira, 2006] o PCEs foi resolvido em três estágios. No primeiro estágio uma SHP é utilizada com o critério de que cada padrão de corte gerado deve satisfazer a demanda de pelo menos dois itens. No segundo estágio, o problema residual formado pelos padrões de cortes gerados é resolvido pelo SHP sem o critério anterior. Por fim, o método de [Foerster e Wäscher, 2000] é aplicado para reduzir os *setups* no plano de corte obtido. Os trabalhos de [Cui e Liu, 2010; Mobasher e Ekici, 2013; Cui et al., 2015] propuseram técnicas híbridas combinando modelos matemáticos e o SHP para resolver o CSPs.

3. Modelo matemático

Para a modelagem do problema, utilizam-se as seguintes notações:

Parâmetros: J : conjunto de padrões de corte (índice j e cardinalidade $|J|$); u_o : custo por objeto. u_s : custo por *setup* (custo para cortar um novo padrão de corte); a_{ij} : número de itens do tipo i no padrão j ; M : limite superior da frequência de padrões, por exemplo, $M = \sum_{i=1}^n d_i$.

Variáveis: x_j : frequência do padrão de corte j ; y_j : variável binária que indica se o padrão de corte j foi cortado ou não.

O modelo de programação linear inteira (PI) para o PCEs é dado por:

$$Z_{ILP} = \min \left\{ u_o \sum_{j=1}^{|J|} x_j + u_s \sum_{j=1}^{|J|} y_j \right\} \quad (1)$$

$$\sum_{j=1}^{|J|} a_{ij} x_j \geq d_i, \quad i \in \{1, \dots, n\} \quad (2)$$

$$x_j \leq M y_j, \quad j \in \{1, \dots, |J|\} \quad (3)$$

$$x_j \in \mathbb{Z}^+, \quad y_j \in \{0, 1\}, \quad j \in \{1, \dots, |J|\} \quad (4)$$

O modelo (1)-(4) aparece em diversos trabalhos na literatura, como por exemplo [Cui et al., 2015; Mobasher e Ekici, 2013; Yanasse e Limeira, 2006]. A função objetivo, representada por (1), consiste na minimização da soma dos custos dos objetos e dos *setups*. As restrições (2) asseguram que as demandas dos itens sejam atendidas. As restrições (3) modelam o *setup* do padrão

de corte j , garantindo que ele seja utilizado se, e somente se, $y_j = 1$. As restrições (4) definem o domínio das variáveis do problema, em que o conjunto $\mathbb{Z}^+$ representa os números inteiros não negativos.

4. BRKGA com Q -learning

O BRKGA é uma metaheurística desenvolvida por [Gonçalves e Resende, 2011], baseada no *Random Key Genetic Algorithm* (RKGA) proposto por [Bean, 1994]. No RKGA, os indivíduos são codificados por meio de números aleatórios (também chamados de chaves aleatórias) pertencentes ao intervalo $[0, 1)$. Assim, um indivíduo χ é representado como um vetor em $[0, 1)^r$, onde r denota o tamanho do cromossomo. Um algoritmo determinístico, denominado *decoder*, é responsável por mapear esses vetores de chaves aleatórias em soluções do problema de otimização considerado.

O processo evolutivo do RKGA inicia-se com a geração aleatória de k indivíduos. A reprodução ocorre mediante a cópia dos melhores indivíduos de uma geração para a seguinte (estratégia elitista). Seja p_e , com $p_e \leq k$, o número de soluções da elite (aquelas com os melhores valores de *fitness*) preservadas entre gerações. A população é, então, dividida entre indivíduos de elite e não-elite. Uma vantagem da estratégia elitista é a melhoria monotônica dos melhores indivíduos ao longo das gerações. Contudo, essa abordagem pode levar à convergência prematura para ótimos locais. Para mitigar esse risco, adota-se uma taxa elevada de mutação: p_m novos indivíduos são gerados aleatoriamente, substituindo a mutação tradicional de baixa probabilidade aplicada gene a gene. Esse mecanismo contribui para manter a diversidade populacional e evita a convergência prematura.

Os $k - p_e - p_m$ indivíduos restantes da nova geração são produzidos por meio do operador de *crossover*. O RKGA emprega o *Parametric Uniform Crossover* (PUX) [Spears e De Jong, 1991], em que dois pais são selecionados para gerar um descendente, e um parâmetro ρ_e define a probabilidade de os genes do filho serem herdados do primeiro ou do segundo pai. O critério de parada do algoritmo baseia-se no número máximo de gerações, max_{gen} , ou no tempo total de execução.

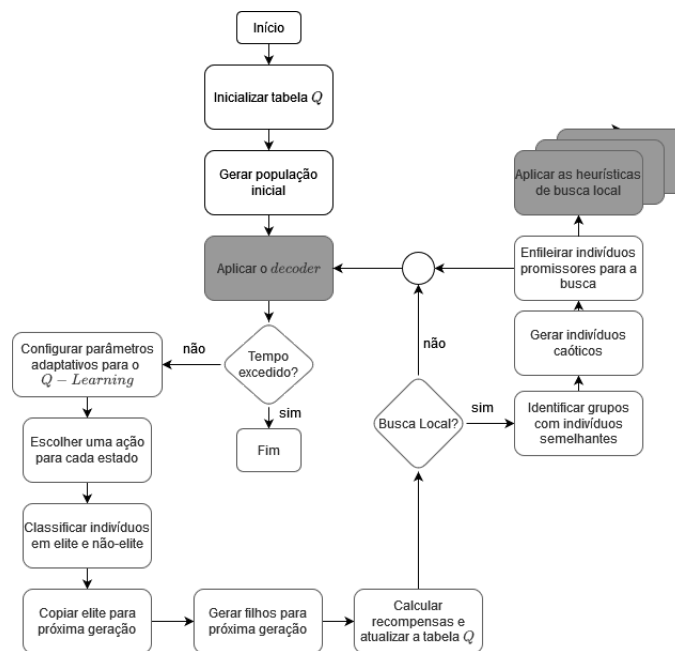
O BRKGA se diferencia do RKGA na forma de seleção dos pais para o operador de cruzamento. Especificamente, no BRKGA, cada descendente é gerado pela combinação de um indivíduo escolhido aleatoriamente da partição de elite e outro da partição não-elite.

No BRKGA-QL proposto por [Chaves e Lorena, 2021], o algoritmo *Q-Learning* [Watkins, 1989] foi empregado para controlar dinamicamente os parâmetros do BRKGA ao longo do processo evolutivo. O *Q-Learning* é um algoritmo de aprendizado por reforço, independente de modelos, que utiliza um *framework* baseado em processos de decisão de Markov para modelar a interação entre um agente de aprendizado e seu ambiente, em termos de estados, ações e recompensas. Por meio da exploração iterativa e da avaliação de todas as ações possíveis em diferentes estados, o agente adquire conhecimento sobre as escolhas ótimas para um dado problema. Essas avaliações são armazenadas em uma estrutura conhecida como tabela Q , que é inicialmente preenchida com valores nulos.

Um estado no BRKGA-QL é representado por uma 4-upla composta pelos valores dos parâmetros k , μ_e , μ_m e ρ_e , em que $p_e = \mu_e k$ e $p_m = \mu_m k$. Cada ação corresponde à alteração de apenas um desses parâmetros. O valor $Q(s_e, h_l)$ indica a utilidade de realizar a ação h_l no estado s_e , refletindo a expectativa de recompensa acumulada. A seleção dos parâmetros a cada geração é guiada pela tabela Q , utilizando uma política ϵ -gulosa. Nessa política, o agente escolhe, com probabilidade $1 - \epsilon$, o par estado-ação com a maior recompensa estimada na tabela Q , e com probabilidade ϵ , seleciona uma ação aleatória, promovendo a exploração. O objetivo do agente é maximizar a recompensa total ao longo do processo evolutivo.

A Figura 1 apresenta o fluxograma genérico do algoritmo BRKGA-QL, proposto por [Schenekemberg et al., 2024]. Dentro desse *framework*, o *decoder* e eventuais algoritmos de busca local são os únicos componentes dependentes do problema. Todos os demais elementos podem ser encapsulados em uma estrutura geral e independente do domínio específico. Em comparação com outros métodos baseados no BRKGA, diversas inovações foram introduzidas por Schenekemberg et al. [2024]. Dentre elas, destaca-se o ajuste adaptativo dos parâmetros por meio do *Q-Learning*, com o objetivo de balancear exploração e intensificação durante o processo evolutivo conduzido pelo agente. Ademais, foram incorporados indivíduos caóticos, gerados a partir de comunidades, bem como uma modificação no operador de mutação, que passou a utilizar o PUX em substituição ao uso tradicional de indivíduos mutantes. Neste trabalho, nenhum algoritmo de busca local foi implementado.

Figura 1: *Framework* do BRKGA-QL. Baseado em Schenekemberg et al. [2024].



4.1. Decoder

O *decoder* proposto baseia-se no método SHP de [Haessler, 1975]. A construção dos padrões de corte considera critérios relacionados à ordenação dos itens e às suas demandas residuais. Para as soluções geradas pelo *decoder*, assume-se que serão utilizados, no máximo, $n + 2$ padrões de corte distintos, ou seja, $n + 2$ setups, em que n representa a quantidade de itens da instância em análise. A ideia central é representar as primeiras $n + 2$ chaves aleatórias como candidatos a padrões de corte.

Cada padrão pode ser construído com base em uma entre três regras de ordenação dos itens:

- **R1:** ordena os itens em ordem decrescente de comprimento;
- **R2:** ordena os itens em ordem decrescente de demanda residual;
- **R3:** ordena os itens de forma aleatória.

Para implementar a ordenação aleatória (R3), são utilizadas n chaves aleatórias adicionais. Por fim, uma chave aleatória extra é usada para determinar qual regra de ordenação será aplicada na construção da solução. Assim, cada indivíduo χ é representado por um vetor de $2n + 3$ chaves aleatórias.

Para inicializar o *decoder*, definimos rd_i como a demanda residual do item do tipo i . Inicialmente, temos $rd_i = d_i, \forall i \in \{1, \dots, n\}$. Define-se o valor $M' = 0,7 \sum_{i=1}^n rd_i$.

Para cada chave aleatória $\chi(j)$, com $j \in \{1, \dots, n + 2\}$, prossegue-se conforme os seguintes passos:

1. Calcula-se M' e $\phi_j = \lceil \chi(j) \cdot M' \rceil$;
2. Verifica-se o valor da chave aleatória $\chi(2n + 3)$ para determinar a regra de ordenação a ser utilizada:
 - se $\chi(2n + 3) \leq 1/3$, utiliza-se a regra R3 (ordem aleatória);
 - se $\chi(2n + 3) \in (1/3, 2/3]$, utiliza-se a regra R2 (ordem decrescente das demandas residuais);
 - se $\chi(2n + 3) \in (2/3, 1)$, utiliza-se a regra R1 (ordem decrescente dos comprimentos).
3. Define-se $L' = 0$. Para $i = 1$ até n , considerando os itens na ordem estabelecida:
 - se $rd_i > 0$ e $L' + a_{ij}l_i \leq L$, incrementa-se a_{ij} enquanto $L' + a_{ij}l_i \leq L$ e $\phi_j a_{ij} \leq rd_i$.
4. Calcula-se a frequência do padrão de corte j como:

$$x_j = \left\lceil \min_i \left\{ \frac{rd_i}{a_{ij}} \mid a_{ij} > 0 \right\} \right\rceil ;$$

5. Atualiza-se a demanda residual de cada item:

$$rd_i = rd_i - x_j \cdot a_{ij}, \quad \text{e se } rd_i < 0, \text{ então } rd_i = 0.$$

6. Se $rd_i = 0$ para todo $i \in \{1, \dots, n\}$, o plano de corte factível foi gerado e o processo é encerrado.

A *fitness* do indivíduo χ é avaliada por meio da função objetivo (1).

Para fins ilustrativos, considere a seguinte instância do PCE: objeto de tamanho $L = 5180$, itens com comprimentos $l_1 = 1020$, $l_2 = 1000$, $l_3 = 660$ e $l_4 = 650$, e respectivas demandas $d_1 = 18$, $d_2 = 105$, $d_3 = 47$ e $d_4 = 13$. Como $n = 4$, cada indivíduo é representado por um vetor de 11 chaves aleatórias. Suponha, por exemplo, o seguinte vetor χ :

$$\chi = (,05 ,04 ,17 ,15 ,32 ,69 ,50 ,14 ,80 ,38 ,29).$$

Nota-se que $\chi(11) = 0,29 < 1/3$, logo, usaremos a regra R3 para ordenar os itens. Para obter a ordenação, consideramos as chaves aleatórias das posições $n + 3$ até $2n + 2$ de χ . O subvetor formado tem tamanho n , em que cada índice de coordenada está associado a um item do tipo i . Para χ , temos o vetor $(\chi(7), \chi(8), \chi(9), \chi(10)) = (0,50 \ 0,14 \ 0,80 \ 0,38)$, em que $\chi(7)$ está associado ao item l_1 , $\chi(8)$ ao item l_2 , $\chi(9)$ ao item l_3 e $\chi(10)$ ao item l_4 . Ordenamos o subvetor de acordo com a ordem decrescente dos valores das chaves aleatórias. O vetor resultante é $(\chi(9), \chi(7), \chi(10), \chi(8))$, o que significa que a ordem a ser considerada é (l_3, l_1, l_4, l_2) .

1. Inicialmente, temos $M' = 128,1$ e $\chi(1) = 0,05$, logo, $\phi_1 = 7$. Iniciando por l_3 , temos que para $a_{31} = 6$, $6 \cdot \phi_1 = 42$, portanto, para $a_{31} = 7$, o produto $7 \cdot \phi_1$ ultrapassa a demanda $rd_3 = 47$. Passamos para o item l_1 com $L' = 6l_3 = 3960$. Nota-se que o item l_1 cabe apenas uma vez, resultando no padrão de corte $a_1^T = (1 \ 0 \ 6 \ 0)$ com perda de 200. Pelo passo 4, temos que

$$x_1 = \left\lceil \min_i \left\{ \frac{18}{1}, \frac{47}{6} \right\} \right\rceil = 8.$$

Atualizamos as demandas residuais da seguinte maneira: $rd_1 = 18 - 8 = 10$ e $rd_3 = 47 - 48 = -1$, portanto, fixamos $rd_3 = 0$. O novo valor de M' é 89,6.

2. Para $\chi(2) = 0,04$, temos $\phi_2 = 4$. Iniciamos a construção do padrão de corte via passo 2 com o item l_1 , visto que $rd_3 = 0$. Neste caso, temos que $a_{12} = 2$ é o limite para inserir o item l_1 de forma que a produção $\phi_2 \cdot a_{12} \leq rd_1$. Ao considerar o próximo item, estamos com $L' = 2040$, daí inserimos o item l_4 três vezes no padrão de corte, resultando em $L' = 3990$ e $a_{42} = 3$. Por fim, inserimos o item l_2 uma vez, gerando o padrão de corte $a_2^T = (2 \ 1 \ 0 \ 3)$ com perda de 190. A frequência do padrão de corte é dada por:

$$x_2 = \left\lceil \min_i \left\{ \frac{10}{2}, \frac{105}{1}, \frac{13}{3} \right\} \right\rceil = 5.$$

Atualizando as demandas residuais, temos $rd_1 = 0$, $rd_2 = 105 - 5 = 100$ e $rd_4 = 13 - 15 = -2$, portanto, fixamos $rd_4 = 0$. O valor de M' torna-se 70.

3. Para $\chi(3) = 0,17$, temos $\phi_3 = 12$. Nota-se que, neste caso, o item l_2 é inserido cinco vezes, criando o padrão de corte $a_3^T = (0 \ 5 \ 0 \ 0)$ com perda de 180. A frequência desse padrão é dada por $rd_2/5 = 20$, ou seja, $x_3 = 20$, consequentemente, $rd_2 = 0$.
4. Não analisamos $\chi(4)$, pois o *decoder* já obteve um plano de corte factível. O plano de corte gerado para esta instância consome 33 objetos e utiliza 3 *setups*.

5. Resultados computacionais

Os testes do BRKGA-QL foram realizados em um Intel i7 com 16 GB de RAM, usando C++. O código-fonte está disponível no *GitHub*¹. Foi adotado um limite de 5 segundos e nove execuções por instância. A análise considerou dois conjuntos de instâncias, cujos resultados são apresentados a seguir.

O primeiro conjunto de dados considera 18 classes de instâncias geradas pelo gerador aleatório CUTGEN1 [Gau e Wäscher, 1995]. Cada classe contém 100 instâncias, conforme as características apresentadas na Tabela 1. A coluna l_i representa o intervalo de comprimento dos itens e a coluna d representa a média das demandas.

Tabela 1: Características das instâncias do primeiro grupo ($L = 1000$).

Classe	n	l_i	d	Classe	n	l_i	d	Classe	n	l_i	d
1	10	[10, 200]	10	7	10	[10, 800]	10	13	10	[200, 800]	10
2	10	[10, 200]	100	8	10	[10, 800]	100	14	10	[200, 800]	100
3	20	[10, 200]	10	9	20	[10, 800]	10	15	20	[200, 800]	10
4	20	[10, 200]	100	10	20	[10, 800]	100	16	20	[200, 800]	100
5	40	[10, 200]	10	11	40	[10, 800]	10	17	40	[200, 800]	10
6	40	[10, 200]	100	12	40	[10, 800]	100	18	40	[200, 800]	100

¹ <https://github.com/antoniochaves19/BRKGA-QL>

A Tabela 2 apresenta uma comparação entre o BRKGA-QL e as melhores heurísticas discutidas em [Cui et al., 2015], incluindo o SGPIP, do inglês, *Sequential Grouping Procedure for Integer Programming*, proposto pelos próprios autores, e o método descrito em [Yanasse e Limeira, 2006], denotado por YL. A coluna LB exibe a média dos limitantes inferiores para o número de objetos na solução. As colunas indexadas por B mostram as médias de objetos utilizados por cada método, enquanto a coluna P apresenta a média de *setups*. Foram considerados o valor de *setup* $u_s = 100$ e o custo de objetos $u_o = L$. Os resultados do BRKGA-QL referem-se ao melhor resultado obtido em cada execução. Observa-se que o método SGPIP apresenta, em geral, os melhores desempenhos médios tanto no consumo de objetos quanto no número de *setups* para a maioria das classes. As únicas exceções com relação ao número médio de *setups* ocorrem na classe 2, onde o BRKGA-QL obteve a menor média e nas classes 3 e 5, em que o YL obteve as menores médias. De forma agregada, o consumo médio de objetos pelo BRKGA-QL é apenas 0,14% superior ao valor médio dos limitantes inferiores. Embora o BRKGA-QL tenha registrado uma média de *setups* superior à do YL nas classes de 1 a 6 e de 7 a 12, seu desempenho global em relação a *setups* ainda é melhor. Isso evidencia que o BRKGA-QL apresenta uma vantagem sobre o YL, especialmente nos grupos de classes de 7 a 18.

Tabela 2: Resultados para cada classe do primeiro conjunto de instâncias ($u_s = 100$, $u_o = L$).

Classe	LB	SGPIP		YL		BRKGA-QL	
		B	P	B	P	B	P
1	11,48	11,49	3,73	11,56	3,31	11,54	3,48
2	110,25	110,25	6,42	110,4	6,95	110,98	5,99
3	22,13	22,13	5,42	22,17	4,96	22,23	5,87
4	215,93	215,93	8,84	215,98	10,32	216,95	9,39
5	42,95	42,95	8,01	42,99	7,63	43,12	9,31
6	424,68	424,68	12,98	424,89	13,31	425,97	14,5
Média 1-6	137,90	137,90	7,57	138,00	7,75	138,46	8,09
7	50,24	50,24	6,40	51,69	7,66	50,32	7,7
8	499,62	499,62	7,03	502,23	9,62	500,82	8,88
9	93,65	93,65	12,26	99,49	13,64	93,76	13,83
10	932,26	932,26	14,03	948,41	18,21	932,81	17,15
11	176,9	176,91	24,21	195,67	24,60	177,28	26,9
12	1763,45	1763,46	28,06	1847,42	33,23	1765,32	33,79
Média 7-12	586,02	586,02	15,33	607,48	17,83	586,72	18,04
13	63,47	63,47	7,52	64,20	8,93	63,94	8,81
14	632,36	632,36	8,10	633,26	10,51	635,38	9,63
15	119,59	119,59	14,15	123,90	16,28	119,64	15,69
16	1192	1192	15,66	1197,66	19,89	1192,64	18,07
17	224,85	224,85	27,12	244,02	29,76	224,92	29,21
18	2242,59	2242,59	30,26	2268,3	37,90	2242,98	34,16
Média 13-18	745,81	745,81	17,13	755,22	20,54	746,58	19,26
Média	489,91	489,91	13,34	500,23	15,37	490,59	15,13

O segundo grupo é composto por 40 instâncias práticas disponibilizadas em [Umetani et al., 2003]. Para esse conjunto, o BRKGA-QL foi comparado com o SGPIP proposto em [Cui et al., 2015]. Neste grupo, os objetos possuem apenas dois comprimentos, $L = 5180$ e $L = 9080$. A instância Fiber07_5180 deste grupo foi utilizada na Subseção 4.1 para ilustrar, de forma prática, o funcionamento do *decoder*.

A Tabela 3 apresenta os resultados para o segundo grupo de instâncias considerando $u_s = 100$ e $u_o = L$. A coluna LB exibe o limitante inferior para o número de objetos na solução. As colunas indexadas por B mostram o número de objetos utilizados por cada método, enquanto a coluna P apresenta o número de *setups*. Nota-se que o BRKGA-QL obtém os menores valores de *setups* em todas as instâncias. Entretanto, em apenas 22 instâncias ele utilizou uma quantidade de objetos menor ou igual à do SGPIP. Além disso, tanto para $L = 5180$ quanto para $L = 9080$, a soma

total de *setups* foi menor no BRKGA-QL, porém com maior consumo de objetos em comparação ao SGPIP.

Tabela 3: Resultados para as instâncias do segundo conjunto ($u_s = 100$, $u_o = L$)

Nome	LB	SGPIP		BRKGA-QL		Nome	LB	SGPIP		BRKGA-QL	
		B	P	B	P			B	P	B	P
Fiber06_5180	33	33	5	34	4	Fiber06_9080	19	19	3	19	3
Fiber07_5180	33	33	4	33	3	Fiber07_9080	19	19	4	19	3
Fiber08_5180	86	86	6	86	4	Fiber08_9080	48	48	4	48	3
Fiber09_5180	53	53	7	54	5	Fiber09_9080	29	29	5	29	4
Fiber10_5180	69	69	7	69	5	Fiber10_9080	39	39	5	39	3
Fiber11_5180	67	67	7	67	6	Fiber11_9080	38	38	6	38	4
Fiber13a_5180	56	56	7	56	6	Fiber13a_9080	32	32	5	32	4
Fiber13b_5180	28	28	5	28	4	Fiber13b_9080	16	16	4	16	3
Fiber14_5180	47	47	10	48	6	Fiber14_9080	27	27	5	27	4
Fiber15_5180	57	57	8	57	6	Fiber15_9080	32	32	5	32	4
Fiber16_5180	82	82	11	84	7	Fiber16_9080	47	47	6	47	6
Fiber17_5180	83	83	8	84	6	Fiber17_9080	47	47	6	47	6
Fiber18_5180	96	96	10	97	8	Fiber18_9080	54	54	7	55	5
Fiber19_5180	133	133	9	133	7	Fiber19_9080	73	73	11	74	5
Fiber20_5180	32	32	8	33	7	Fiber20_9080	19	19	6	19	5
Fiber23_5180	141	141	18	144	14	Fiber23_9080	80	80	9	81	7
Fiber26_5180	190	190	13	192	8	Fiber26_9080	107	107	9	108	6
Fiber28a_5180	83	83	27	85	8	Fiber28a_9080	48	48	7	48	7
Fiber28b_5180	117	118	13	120	10	Fiber28b_9080	67	67	8	68	7
Fiber29_5180	62	62	11	63	9	Fiber29_9080	35	35	7	36	6
Total	1548	1549	194	1567	133	Total	876	876	122	882	95

A Tabela 4 apresenta os resultados para o segundo grupo de instâncias considerando $u_s = L$ e $u_o = L$. O significado das colunas é similar ao da Tabela 3. Nesse cenário de custos, a solução obtida pelo BRKGA-QL obteve um valor menor que o valor da solução de SGPIP apenas para as instâncias Fiber16_5180, Fiber16_9080 e Fiber17_9080. Com exceção dessas três instâncias, o BRKGA-QL obteve, de modo geral, os menores valores de *setups* em todas as demais. Embora em alguns casos tenha ocorrido aumento no consumo de objetos, a redução nos *setups* foi suficiente para manter a função objetivo, na pior das hipóteses, equivalente à do SGPIP. Observa-se uma melhora na função objetivo em 31 instâncias.

Tabela 4: Resultados para as instâncias do segundo conjunto ($u_s = u_o = L$)

Nome	LB	SGPIP		BRKGA-QL		Nome	LB	SGPIP		BRKGA-QL	
		B	P	B	P			B	P	B	P
Fiber06_5180	33	34	4	34	4	Fiber06_9080	19	19	4	20	3
Fiber07_5180	33	34	3	33	3	Fiber07_9080	19	19	4	19	3
Fiber08_5180	86	86	6	87	3	Fiber08_9080	48	48	4	48	3
Fiber09_5180	53	55	5	54	5	Fiber09_9080	29	30	4	29	4
Fiber10_5180	69	70	6	70	4	Fiber10_9080	39	39	5	39	3
Fiber11_5180	67	67	7	68	5	Fiber11_9080	38	38	5	38	4
Fiber13a_5180	56	56	7	56	6	Fiber13a_9080	32	32	5	32	4
Fiber13b_5180	28	28	5	28	4	Fiber13b_9080	16	16	4	16	3
Fiber14_5180	47	48	6	49	5	Fiber14_9080	27	27	5	28	3
Fiber15_5180	57	57	7	58	4	Fiber15_9080	32	32	5	32	4
Fiber16_5180	82	83	8	84	8	Fiber16_9080	47	47	6	48	6
Fiber17_5180	83	83	8	85	5	Fiber17_9080	47	47	5	48	5
Fiber18_5180	96	96	11	98	6	Fiber18_9080	54	54	7	55	5
Fiber19_5180	133	133	9	133	7	Fiber19_9080	73	74	7	74	5
Fiber20_5180	32	32	8	33	7	Fiber20_9080	19	19	6	19	5
Fiber23_5180	141	142	13	146	8	Fiber23_9080	80	81	8	81	6
Fiber26_5180	190	191	12	193	8	Fiber26_9080	107	107	9	108	6
Fiber28a_5180	83	84	11	85	8	Fiber28a_9080	48	48	8	48	6
Fiber28b_5180	117	119	12	120	10	Fiber28b_9080	67	67	8	68	7
Fiber29_5180	62	62	11	64	8	Fiber29_9080	35	35	7	36	6
Total	1548	1560	159	1578	118	Total	876	879	116	886	91

A Figura 2 apresenta o *performance profile* (PP) de [Dolan e Moré, 2002], comparando a função objetivo (1) dos métodos nos dois grupos de instâncias. Para a plotagem do gráfico, foi considerada a escala logarítmica $\log_2$, e consideraram-se apenas as soluções cujo *gap* percentual foi, no máximo, 1% da melhor solução encontrada entre os métodos apresentados. A Figura 2a apresenta o PP para a função objetivo dos resultados da Tabela 2. Para $\tau = 1$ ($\log_2(\tau) = 0$), SGPIP resolve aproximadamente 93% das instâncias com os melhores *gaps*, seguido por YL, que obtém os melhores *gaps* para 8% das instâncias, enquanto BRKGA-QL não resolve nenhuma instância. Para $\tau = 1024$ ($\log_2(\tau) = 10$), BRKGA-QL consegue resolver todas as instâncias com *gaps* menores que 1%, enquanto YL resolve apenas 50% das instâncias. A Figura 2b apresenta os resultados referentes à Tabela 3. Para $\tau = 1$, tanto SGPIP quanto BRKGA-QL conseguem os melhores *gaps* para 55% das instâncias; porém, BRKGA-QL consegue resolver no máximo 60% das instâncias com *gaps* menores que 1% considerando valores maiores de τ . O cenário muda quando os custos de *setup* e objetos são iguais, cujos resultados estão apresentados na Tabela 4 e PP apresentado na Figura 2c. Neste caso, quando $\tau = 1$, BRKGA-QL consegue resolver 91% das instâncias com *gaps* menores que 1% da melhor solução, enquanto SGPIP resolve apenas 21%. Quando maiores valores de τ são considerados, SGPIP consegue resolver apenas 30%.

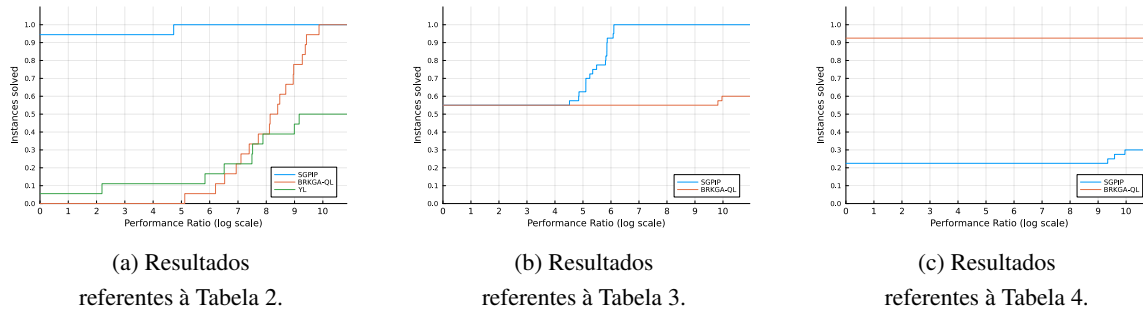


Figura 2: *Performance profile* para as instâncias dos grupos 1 e 2.

6. Conclusões

Este trabalho apresentou uma aplicação do *Biased Random-key Genetic Algorithm* com *Q-learning* (BRKGA-QL), visando resolver o Problema de Corte de Estoque unidimensional com *setups* (PCEs). Foi proposto um *decoder* baseado na *Sequential Heuristic Procedure* (SHP) de [Hassler, 1975]. Os resultados foram comparados com os métodos de [Cui et al., 2015] e [Yanasse e Limeira, 2006]. Dois conjuntos de instâncias foram considerados: um gerado aleatoriamente pelo CUTGEN1 de [Gau e Wäscher, 1995] e outro composto por instâncias reais apresentadas em [Umetani et al., 2003]. O BRKGA-QL mostrou-se competitivo no primeiro grupo de instâncias, obtendo médias de consumo de objetos próximas ao limitante inferior. No segundo grupo, obteve as soluções com os menores números de *setup*, embora com valores maiores de função objetivo, devido ao custo do objeto ser superior ao do *setup*. Por outro lado, em um cenário com custos iguais para objeto e *setup*, o BRKGA-QL foi superior em 31 das 40 instâncias do segundo grupo, demonstrando eficiência do método em contextos com alto custo de *setup*.

Como proposta para trabalhos futuros, pode-se desenvolver buscas locais aplicadas ao *decoder*. Além disso, o *decoder* pode ser estendido para tratar *setups* dependentes da sequência, assim como o *framework* de busca baseado em chaves aleatórias pode ser adaptado para outras metaheurísticas, como em [Chaves et al., 2024].

Agradecimentos

Eduardo M. Silva agradece a Fundação de Amparo à Pesquisa do Estado de São Paulo, (FAPESP) (processo 23/04588-4). Antonio A. Chaves agradece a FAPESP (processos 2018/15417-

8, 2022/05803-3 e 2024/08848-3), e também ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) (processos 423694/2018-9 e 303736/2018-6). Silvio A. de Araujo agradece a FAPESP (processos 2013/07375-0, 2022/05803-3 e 2024/01409-4), e também ao CNPq (processos 302998/2022-5 and 402240/2023- 5).

Referências

- Cherri, A. C., Arenales, M. N., Yanasse, H. H., Poldi, K. C., e Vianna, A. C. G. (2014). The one-dimensional cutting stock problem with usable leftovers - a survey. *European Journal of Operational Research*, 236(2):395 – 402.
- Bean, J. C. (1994). Genetic algorithms and random keys for sequencing and optimization. *ORSA Journal on Computing*, 6(2):154–160.
- Chaves, A. A., Resende, M. G. C., Schuetz, M. J. A., Brubaker, J. K., Katzgraber, H. G., Arruda, E. F., e Silva, R. M. A. (2024). A random-key optimizer for combinatorial optimization. *arxiv*. URL <https://doi.org/10.48550/arXiv.2411.04293>.
- Chaves, A. A. e Lorena, L. H. N. (2021). An adaptive and near parameter-free brkga using q-learning method. In *2021 IEEE Congress on Evolutionary Computation (CEC)*, p. 2331–2338.
- Cui, Y., Yang, L., Zhao, Z., Tang, T., e Yin, M. (2013). Sequential grouping heuristic for the two-dimensional cutting stock problem with pattern reduction. *International Journal of Production Economics*, 144(2):432–439.
- Cui, Y., Zhong, C., e Yao, Y. (2015). Pattern-set generation algorithm for the one-dimensional cutting stock problem with setup cost. *European Journal of Operational Research*, 243(2):540 – 546.
- Cui, Y. e Liu, Z. (2010). C-sets-based sequential heuristic procedure for the one-dimensional cutting stock problem with pattern reduction. *Optimization Methods and Software*, 26(1):155–167.
- de Araujo, S. A., Poldi, K. C., e Smith, J. (2014). A genetic algorithm for the one-dimensional cutting stock problem with setups. *Pesquisa Operacional*, 34:165 – 187.
- Dolan, E. D. e Moré, J. J. (2002). Benchmarking optimization software with performance profiles. *Mathematical Programming*, 91(2):201 – 213.
- Foerster, H. e Wäscher, G. (2000). Pattern reduction in one-dimensional cutting stock problems. *International Journal of Production Research*, 38(7):1657 – 1676.
- Gau, T. e Wäscher, G. (1995). Cutgen1: A problem generator for the standard one-dimensional cutting stock problem. *European Journal of Operational Research*, 84(3):572 – 579.
- Golfeto, R. R., Moretti, A. C., e de Salles Neto, L. L. (2009). A genetic symbiotic algorithm applied to the cutting stock problem with multiple objectives. *Advanced Modeling and Optimization*, 11 (4):473 – 501.
- Gonçalves, J. F. e Resende, M. (2011). Biased random-key genetic algorithms for combinatorial optimization. *J Heuristics*, 17:487–525.

- Haessler, R. W. (1975). Controlling cutting pattern changes in one-dimensional trim problems. *Operations Research*, 23(3):483–493.
- Henn, S. e Wäscher, G. (2013). Extensions of cutting problems: setups. *Pesquisa Operacional*, 33: 133 – 162.
- Martin, M., Yanasse, H. H., Salles-Neto, e Leduino, L. (2022). Pattern-based ilp models for the one-dimensional cutting stock problem with setup cost. *Journal of Combinatorial Optimization*, 1.
- McDiarmid, C. (1999). Pattern minimisation in cutting stock problems. *Discrete Applied Mathematics*, 98(1-2):121 – 130.
- Mobasher, A. e Ekici, A. (2013). Solution approaches for the cutting stock problem with setup cost. *Computers and Operations Research*, 40(1):225–235.
- Schenekemberg, C. M., Chaves, A. A., Guimarães, T. A., e Coelho, L. C. (2024). Hybrid metaheuristic for the dial-a-ride problem with private fleet and common carrier integrated with public transportation. *Annals of Operations Research*, p. 1–39.
- Spears, W. M. e De Jong, K. A. (1991). On the virtues of parametrized uniform crossover. *Proc. 4th Int. Conf. Genetic Algorithms*, R. Belew and L. Booker, Eds. San Mateo, CA: Morgan Kaufmann, p. 230–236.
- Suliman, S. M. A. (2012). An algorithm for solving lot sizing and cutting stock problem within aluminum fabrication industry. In *International Conference on Industrial Engineering and Operations Management*, 2012.
- Umetani, S., Yagiura, M., e Ibaraki, T. (2003). One-dimensional cutting stock problem to minimize the number of different patterns. *European Journal of Operational Research*, 146(2):388 – 402.
- Umetani, S., Yagiura, M., e Ibaraki, T. (2006). One-dimensional cutting stock problem with a given number of setups: A hybrid approach of metaheuristics and linear programming. *Journal of Mathematical Modelling and Algorithms*, 5(1):43–64.
- Vanderbeck, F. (2000). Exact algorithm for minimising the number of setups in the one-dimensional cutting stock problem. *Operations Research*, 48(6):915–926.
- Watkins, C. J. C. H. (1989). *Learning from Delayed Rewards*. PhD thesis, King's College, Oxford.
- Wuttke, D. A. e Heese, S. H. (2018). Two-dimensional cutting stock problem with sequence dependent setup times. *European Journal of Operational Research*, 265(1):303 – 315.
- Yanasse, H. H. e Limeira, M. S. (2006). A hybrid heuristic to reduce the number of different patterns in cutting stock problems. *Computers & Operations Research*, 33(9):2744 – 2756.