Universidade Federal do Maranhão
Centro de Ciências Exatas e Tecnologia
Curso de Ciência da Computação

GILVAN TAVARES RABELO JUNIOR

BUSCA EVOLUTIVA GUIADA POR AGRUPAMENTOS DE CHAVES ALEATÓRIAS VICIADAS APLICADA AO PROBLEMA DE SEQUENCIAMENTO DE PADRÕES

São Luís
2019

GILVAN TAVARES RABELO JUNIOR

BUSCA EVOLUTIVA GUIADA POR AGRUPAMENTOS DE CHAVES ALEATÓRIAS VICIADAS APLICADA AO PROBLEMA DE SEQUENCIAMENTO DE PADRÕES

Monografia apresentada ao curso de Ciência da Computação da Universidade Federal do Maranhão, como parte dos requisitos necessários para obtenção do grau de Bacharel em Ciência da Computação.

Orientador: Profº Dr. Alexandre César Muniz de Oliveira

São Luís

2019

Ficha gerada por meio do SIGAA/Biblioteca com dados fornecidos pelo(a) autor(a).
Núcleo Integrado de Bibliotecas/UFMA

Rabelo Junior, Gilvan Tavares.
Busca Evolutiva Guiada por Agrupamentos de chaves Aleatórias Viciadas Aplicada ao Problema de Sequenciamento de Padrões / Gilvan Tavares Rabelo Junior. – 2019.
39 f.

Orientador(a): Alexandre César Muniz de Oliveira.
Monografia (Graduação) – Curso de Ciência da Computação, Universidade Federal do Maranhão, Laboratório INOVTEC / DEINF, 2019.

1. BRKeCS. 2. BRKGA. 3. MOSP. 4. Sequenciamento de Padrões. I. Oliveira, Alexandre César Muniz de. II. Título.

Dedico este trabalho à minha mãe, Adelaide, por nunca deixar de me apoiar e nunca deixar faltar amor durante toda essa caminhada.

Agradecimentos

Agradeço primeiramente à dona Adelaide, que mesmo não tendo o mesmo sangue que o meu, me tratou como filho e sem ela, eu não chegaria até aqui. Obrigado mãe por toda palavra de incentivo que tu me deste. Obrigado pela paciência e amor durante essa jornada.

Agradeço também aos meus irmãos, Janielle, por todos os ensinamentos durante meu crescimento, sejam eles na vida de estudante, sejam eles no caráter pessoal e Ricardo por sempre ser uma pessoa com o coração enorme e positiva ao meu redor.

Agradeço ao meu Orientador, Prof. Dr. Alexandre César Muniz de Oliveira por todos seus ensinamentos desde o princípio dos tempos de PET, até os dias de hoje, onde sou membro do LACMOR. Sou muito grato por todas as oportunidades que o senhor me proporcionou durante essa jornada.

Agradeço também aos amigos Marvin, Arthur, Amanda, Matheus, Eduardo, Moisés, Chrystian, Felipe, Fernando e Italo, por todas as conversas, risadas, histórias e conselhos até aqui.

Agradeço em especial à Daniela, por toda a força que eu não sei de onde tiraria mesmo nos dias mais difíceis. Nunca esquecerei isso. Muito obrigado por todo esse aprendizado que tive ao seu lado.

Por fim, agradeço à você que está lendo esta monografia, que será eterno enquanto houver alguém que te recordes.

“A única coisa tão inevitável quanto a morte é a vida.” - Charles Chaplin

RESUMO

Este trabalho propõe a resolução do problema de minimização de pilhas abertas *MOSP* (do inglês, *minimization of open stacks problem*) por meio da metaheurística de busca evolutiva guiada por agrupamentos de chaves aleatórias ou *BRKeCS*(do inglês, *biased random-keys genetic evolutionary clustering search*), onde se utiliza premissas do algoritmo genético de chaves aleatórias viciadas, o *BRKGA*(do inglês, *biased random-keys genetic algoritm*) junto a busca evolutiva guiada por agrupamentos ou *ECS*(do inglês, *evolutionary clustering search*). O *MOSP* é oriundo do cenário de sistemas de produção de indústrias, e consiste em determinar uma sequência de padrões de corte, onde se minimiza o número máximo de pilhas abertas durante o processo de corte. As estruturas de codificação e decodificação, além da função de aptidão ou *fitness* são pontos importantes para resolução de instâncias *MOSP*. Testes computacionais foram realizados utilizando instâncias presentes na literatura.

Palavras-chaves: BRKGA, BRKeCS, MOSP, Sequenciamento de padrões, CS, ECS.

ABSTRACT

This work proposes the resolution of the problem of the minimization of open stacks problem (*MOSP*) by a metaheuristic called *biased random-keys evolutionary clustering search*(*BRKeCS*) , where it uses assumptions of the genetic algorithm of *biased random-keys genetic algorithm* (*BRKGA*) along to *evolutionary clustering search*(*ECS*). *MOSP* comes from the production systems scenario of the industries, and consists of determining a sequence of cutting patterns, where the maximum number of open stacks is minimized during the cutting process. For resolution, The encoder and decoder structures, in addition to the fitness function, are important keys for resolution of the *MOSP* instances. Computational tests were performed using instances present in the literature.

Keywords: BRKGA, BRKeCS, MOSP, pattern sequencing, CS, ECS.

Lista de ilustrações

Figura 1 – M e M^1 correspondentes a permutação s = (1, 2, 3, 4) do exemplo da Tabela 1 19
Figura 2 – M e M^1 correspondente a melhor permutação s = (3, 1, 2, 4) do exemplo da Tabela 1 19
Figura 3 – Projeto conceitual do *BRKeCS* 23
Figura 4 – Matriz de pilhas abertas M^1 gerada a partir de M 26
Figura 5 – Exemplo de codificação e decodificação de chaves aleatórias . . . 28
Figura 6 – Representação gráfica entre as médias de *gaps* entre *BRKGA* e *BRKeCS* 36

Lista de tabelas

Tabela 1 – Dados do exemplo ilustrativo do *MOSP* 18
Tabela 2 – Configuração de parâmetros gerada pelo *CVR* 30
Tabela 3 – Instâncias do *MOSP* 31
Tabela 4 – Tabela com instâncias Simonis, Shaw e Harvey 33
Tabela 5 – Instâncias presentes em Faggioli Bentivoglio 34
Tabela 6 – Instâncias apresentadas por Miller e Wilson 35
Tabela 7 – Instâncias da empresa "A" 35
Tabela 8 – instâncias da empresa "B" 35
Tabela 9 – Médias entre classes de instâncias 36

Lista de abreviaturas e siglas

BRKGA	*Biased Random-key Genetic Algorithm*
CGA	*Constructive Genetic Algorithm*
CS	*Clustering Search*
CVR	*Cross Validated Racing*
ECS	*Evolutionary Clustering Search*
MDP	*Minimization of Discontinuities Problem*
MORP	*Minimization of Order Spread Problem*
MOSP	*Minimization of Open Stacks Problem*
RKGA	*Random-key Genetic Algorithm*
VND	*Variable Neighborhood Descent*

Sumário

1 INTRODUÇÃO 13
1.1 Breve revisão bibliográfica 14
1.2 Objetivos 15
1.3 Organização do Trabalho 15

2 FUNDAMENTAÇÃO TEÓRICA 17
2.1 Problema de sequenciamento de padrões 17
2.2 Algoritmo genético de chaves aleatórias viciadas 20
2.3 Busca Evolutiva guiada por agrupamento 21
2.4 Busca evolutiva guiada por agrupamentos de chaves aleatórias viciadas 22
2.4.1 Cruzamento 23
2.4.2 Assimilação 24

3 PROPOSTA 25
3.1 Modelagem do problema 25
3.2 Função de Aptidão 26
3.3 Codificador e decodificador 27
3.4 Busca local 28
3.5 Parâmetros do BRKeCS 29

4 RESULTADOS 31
4.1 Apresentação das instâncias 31
4.2 Resultados computacionais 32

5 CONCLUSÃO 38
5.1 Trabalhos futuros 38

REFERÊNCIAS 39

1 Introdução

Atualmente, empresas que possuem grande escala de produção estão em busca de uma forma inteligente de melhor utilizar os recursos disponíveis sem perder tempo e dinheiro. Um dos problemas oriundos dessa massiva necessidade de produção é o problema de corte de estoque que consiste em cortes de peças maiores(chamados de objetos) em peças menores(chamados de itens) em tamanhos e quantidades sob demanda. Na resolução deste problema busca-se minimizar a quantidade de material desperdiçado ou maximizar o lucro, escolhendo uma coleção dos melhores padrões de corte. Outro fator importante, em certos casos, é definir a sequência em que a coleção de padrões de corte devem ser realizadas, a fim de minimizar a pilha máxima de padrões parcialmente cortados. Assim que os padrões são cortados, os itens são empilhados, sendo que cada item vai para uma pilha especifica com seu tipo. A partir do momento que um tipo de item é cortado, uma pilha é considerada aberta e permanecerá aberta até que o último pedaço de um item do mesmo tipo seja cortado. Devido ao espaço disponível ou limitações de equipamentos, pode ser que algumas pilhas sejam removidas para liberar espaços para novas pilhas, logo é aconselhado manter poucas pilhas abertas durante o processo de corte. Se uma pilha for removida pelo fim de corte e existirem outras pilhas ainda em aberto, elas devem retornar para continuar o andamento do processo. Para evitar esta ineficiência causada pela ida e retorno de pilhas abertas, é importante determinar não só a melhor ordem de corte dos padrões, mas também o número máximo de pilhas abertas durante o processo de corte. Este problema é conhecido como o problema de pilhas abertas (em inglês, *minimization of open stacks problem*, MOSP).

Uma solução para este problema presente na literatura e proposto em (GONçALVES; RESENDE; COSTA, 2014) utiliza-se a meta-heuristica evolutiva para problemas de otimização *BRKGA* (*biased random-key genetic algorithm*) combinada com o procedimento de busca local para gerar uma sequência de padrões de corte.No *BRKGA* Cada Solução é representado por um vetor de n chaves aleatórias, onde cada chave aleatória é um número real, gerado aleatoriamente, no intervalo de [0,1). Um decodificador mapeia um vetor de chaves aleatórias numa solução do problema de otimização e calcula o custo desta solução. A população inicial p vetores de chaves aleatórias, que a cada iteração, são particionados em um conjunto pequeno de melhores elementos (elite) e o restante (não-elite). Assim, todos os elementos pertencentes a elite são copiados sem nenhuma modificação à população da próxima iteração(GONçALVES; RESENDE; COSTA, 2014).

Neste trabalho, propõe-se a utilização de uma nova abordagem para resolução

do problema *MOSP* a fim de comparar os resultados obtidos com os resultados obtidos em Gonçalves, Resende e Costa (2014). O algoritmo proposto é o genético híbrido *biased sandom-key evolutionary clustering search*(*BRKeCS*) que é baseado em chaves aleatórias viciadas (*BRKGA*) e na busca guiada por agrupamentos(em inglês *clustering seach*).O *ECS* tenta localizar áreas de pesquisa promissoras enquadrando-as por agrupamentos que são representados por centros *c*. O número de máximo de agrupamentos pode ser fixado ou determinado dinamicamente de acordo com a largura das áreas de busca(tamanho do problema em questão). A cobertura do *cluster* é determinada por uma métrica de distância que calcula a semelhança entre uma dada solução e o centro do *cluster*, *c*). As métricas mais populares em aplicações são as distâncias de hamming e euclidiana (OLIVEIRA; LORENA, 2004).

1.1 Breve revisão bibliográfica

Em Oliveira e Lorena (2002) é descrito uma aplicação de um algoritmo genético construtivo(*CGA*, do inglês *Constructive Genetic Algorithm*) para o problema de minimização de pihas abertas(*MOSP*). O *CGA* apresenta novas características comparado aos algoritmos genéticos tradicionais, como por exemplo, uma população de tamanho dinâmico composto por esquemas e estruturas e treinada de acordo com o problema em questão. O *CGA* para o *MOSP* utiliza a heurística *2-Opt* para definir a função de aptidão e o operador de mutação.

Fink, Yanasse e Costa (2009) demonstra em testes quatro variações do modelo apresentado por Yanasse e Pinto (2003). Utilizando grafos, o objetivo é determinar a ordem em que as pilhas serão fechadas, minimizando assim o número máximo de pilhas abertas, o que difere de outros trabalhas que tem como objetivo a forma que os itens vão ser processados.

Em Yanasse e Senne (2010) é proposta uma redução do número de padrões pela determinação de uma instância equivalente. Os autores propõem um novo algoritmo que gera uma instância com menos padrões e é equivalente ao grafo de itens da instância original.

Em Lopes (2011), foi desenvolvido um modelo de programação inteira para o *MOSP*, mas também aplicado a outros problemas, como o *MTSP*. O *MOSP* é modelado como um grafo de intervalos, onde cada pilha aberta é associada a um desses intervalos. O objetivo é ordenar estes intervalos de certa forma que a sobreposição entre eles seja mínima, assim, fazendo com que menos pilhas sejam abertas ao mesmo tempo.

Fink (2012) propõe duas heurísticas, onde a primeira é uma estratégia de solução que consiste em converter uma instância do *MOSP* em uma instância do problema do caxeiro viajante. A motivação é dada pelo fato que o *TSP* possui uma

literatura bastante rica no que se trata de formulações com métodos exatos.

Na segunda heurística proposta, a idéia é decompor o problema original em problemas menores. São definidas condições de resultados ótimos relacionando os valores da solução dos subproblemas obtidos com o problema original.

Em Gonçalves, Resende e Costa (2014) é proposta uma heurística construtiva que usa um Algoritmo Genético de Chaves Aleatórias(*BRKGA*), um procedimento de busca local e uma nova avaliação de aptidão.

No trabalho apresentado em Lima e Carvalho (2016) foram utilizados os métodos Decida em Vizinhança Variável e de Descida mais Rápida para resolução do *MOSP*. O método Descida em vizinhança variável(ou variable neighborhood descent - *VND*), define um conjunto N de K diferentes estruturas de vizinhança e determina um ótimo local em cada uma delas. Uma vizinhança K específica é explorada a cada iteração, caso seja encontrado um melhor que o ótimo atual, esté é substituido. Em caso de não haver melhoria, explora-se a vizinhança próxima até que as k vizinhanças sejam analisadas. O método de Descida Mais Rápida é um método de busca local, onde a vizinhança é explorada pelo método *2-opt*, diferentemente do k-opt utlizado pelo *VND*. Outra diferença é que a estratégia de agrupamentos usada no *VND* não é utilizada, substituída por uma estratégia de janela deslizante.

1.2 Objetivos

O Objetivo deste trabalho é a utilização da metaheuristica *BRKeCS* para resolução do problema de minimização de pilhas abertas, ou *MOSP*(do inglês, *minimization of open stacks problem*). Além disso, fazer um levantamento comparativo dos resultados obtidos neste trabalho, com os resultados presentes em Gonçalves, Resende e Costa (2014). Para isso, este trabalho deve seguir os seguintes procedimentos:

i Modificação na leitura de instâncias do *MOSP*;

ii Modificação da Função de Aptidão de acordo com o *MOSP*;

iii Adaptação das instâncias do *MOSP* para execução do *BRKeCS*;

iv Comparações e considerações finais sobre o desempenho dos algoritmos.

1.3 Organização do Trabalho

O trabalho está organizado da seguinte forma: No Capítulo 2, apresentam-se conceitos relacionado ao problema de sequenciamento de padrões, mais especificamente o *MOSP*. Além disso, metaheurísticas como o algoritmo genético de chaves

aleatórias viciadas ou *BRKGA* (do inglês. *Biased random-keys algorithm*), busca evolutiva por agrupamento ou *ECS* (do inglês, *Evolutionary clustering search*) e a busca evolutiva guiada por agrupamentos de chaves aleatórias viciadas ou *BRKeCS*(do inglês, *Biased random-keys evolutionary clustering search*) são apresentadas.

No Capítulo 3, é apresentado a proposta dos métodos e modelagens para resolução do *MOSP* via *BRKeCS*. A modelagem matemática do problema e a função objetivo são exibidos, respectivamente. Em seguida são definidos estruturas importantes para execução do *BRKeCS*. Por fim. a definição e a configuração dos parâmetros de entrada do BRKeCS são expostos. No Capítulo 4, primeiramente são descritas as instâncias do MOSP utilizadas para resolução via BRKeCS. Em seguida, apresentam-se tabelas com os resultados obtidos e comparações com outras abordagens são realizadas.

No Capítulo 5, as considerações finais sobre o trabalho e sobre os resultados são abordados.

2 Fundamentação Teórica

Neste capítulo, são apresentados conceitos referentes ao Problema de Sequenciamento de Padrões, que inclui uma série problemas de otimização permutacionais relacionados a cenários operacionais encontrados na indústria de manufatura de aço, papel e madeira, bem como os algoritmos metaheurísticos usados para resolvê-lo.

2.1 Problema de sequenciamento de padrões

No âmbito industrial, empresas que fabricam ou manuseiam materiais como aço, papel, madeira, entre outros, encontram problemas relacionados a produção que estão ligados ao custo e ao desperdício de matéria prima. Para resolução desses problemas, as companhias buscam por métodos de otimização que visam minimizar os custos ou os desperdícios gerados. Um desses problemas, conhecido como o problema de corte de produção(PINTO; YANASSE, 2019), baseia-se em cortar peças maiores, chamados de *objetos*, que estão disponíveis em estoque ou obtidas de terceiros, para obter peças menores ou *itens*, em quantidade e dimensões sob demanda. O foco do problema visa otimizar uma determinada função objetivo, por exemplo, custo de produção, desperdícios e etc. Para este problema, a solução geralmente baseia-se em *padrões de corte*(a forma que os objetos serão cortados) e o número de vezes que cada padrão será cortado para atender determinada demanda.

Associado ao problema de corte, encontra-se o chamado problema de sequenciamento de padrões, que consiste em determinar uma sequência em que os padrões deverão ser cortados dependendo do objetivo específico, como por exemplo, minimizar o tempo máximo que uma pilha se mantém aberta (*MORP*, do inglês *minimization of order spread problem*), minimizar o número de vezes em que o processamento de um determinado item é reiniciado, isto é, a quantidade de descontinuidades(*MDP*, do inglês *minimization Of discontinuities problem*) , determinar uma sequência de padrões que minimize o número máximo de pilhas abertas durante o processo de corte (*MOSP*, do inglês *minimization of open stacks problem*), etc.

Problema de minimização de pilhas abertas

Uma solução de corte de estoque define um conjunto de padrões de corte e o número de vezes que os padrões devem ser cortados para satisfazer a demanda de itens. Quando os padrões são cortados, os itens cortados são empilhados, uma pilha para cada tipo de item(CARVALHO; SOMA, 2011). A primeira vez que um tipo de item é cortado, uma pilha é considerada *aberta*. Ele permanece *aberta* até que a

última parte do tipo de item correspondente seja cortada. Uma pilha é *fechada* quando a última peça de um tipo de item é cortada. Devido a disponibilidade de espaço ou de equipamentos limitações, o que pode forçar algumas pilhas a serem removidas para liberar espaço para as novas pilhas, é desejável manter um número pequeno de pilhas abertas durante o processo de corte.

Pilhas fechadas podem ser movidas para outro local ou podem ser entregues aos clientes. Se removido, as pilhas abertas devem ser devolvidas posteriormente para que o trabalho possa ser concluído. Isso é ineficiente, pois leva tempo e usa recursos escassos. Para evitar as ineficiências causadas pela remoção e retorno posterior de pilhas abertas, é importante determinar a ordem de corte ideal dos padrões, de forma que o número máximo de pilhas abertas durante o processo de corte seja minimizado. Este problema é conhecido como o problema de minimização de pilhas abertas, ou *MOSP* . Para ilustrar o *MOSP*, temos um exemplo do problema proposto por Gonçalves, Resende e Costa (2014) com cinco tipos de itens e quatro padrões. O detalhamento deste problema pode ser visto na tabela 1.

Tabela 1 – Dados do exemplo ilustrativo do *MOSP*

Itens	**Padrões Contendo Itens**
A	1,3
B	1,2,4
C	1,2
D	3
E	2,4

O *MOSP* é definido por Yanasse (2010) do seguinte modo: Seja $M_{I\times J}$ uma matriz boleana, onde cada linha corresponde a um tipo de item e cada coluna corresponde a um padrão de corte. Cada Entrada de $M_{i,j}$(com i = 1, ..., I e j = 1, ..., J) igual a 1 se e somente se pelo menos um item do tipo i for contido no padrão j. Seja M_s^1 a matriz resultante que corresponde a permutação s das colunas de M tal que em qualquer linha de M_s^1 cada entrada 0 entre duas entradas 1 são substituidas por 1.

A Figura 1 representa as matrizes M e M^1 correspondentes a permutação s = (1, 2, 3, 4) do exemplo representado na Tabela 1

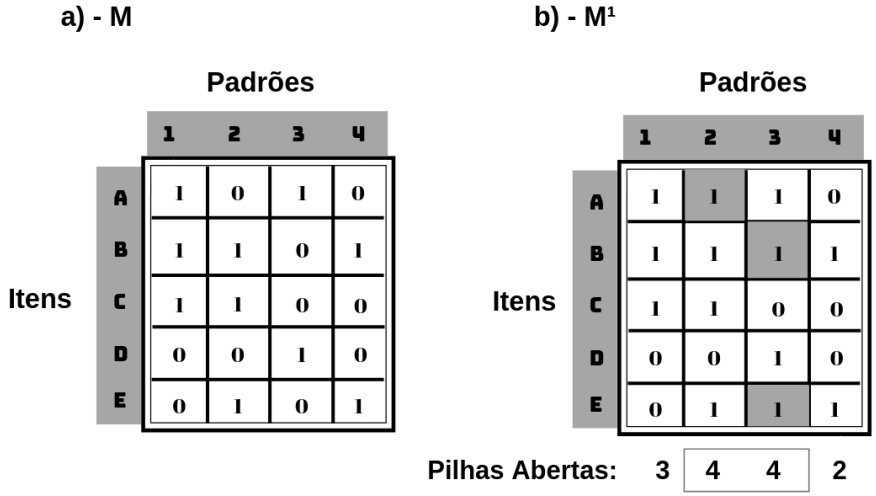

Figura 1 – M e M^1 correspondentes a permutação s = (1, 2, 3, 4) do exemplo da Tabela 1
Fonte:(Gonçalves, Resende e Costa (2014))

O objetivo do *MOSP* é encontrar a melhor permutação s^* de colunas, de certo modo que a quantidade máxima de numeros 1 em qualquer coluna na matriz $M^1_{s^*}$ esteja minimizado (YANASSE; SENNE, 2010). A Figura 2 mostra a melhor solução para o exemplo. Corresponde a permutação s = (3, 1, 2, 4) com o valor do *MOSP* de três.

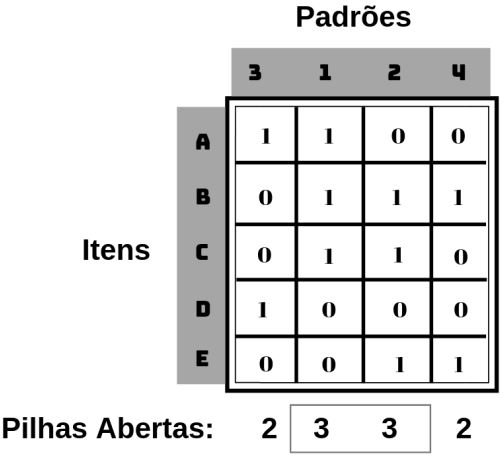

Figura 2 – M e M^1 correspondente a melhor permutação s = (3, 1, 2, 4) do exemplo da Tabela 1
Fonte:(Gonçalves, Resende e Costa (2014))

O *MOSP* é um problema NP-difícil (LINHARES; YANASSE, 2002). Sendo assim, quanto maior as instâncias deste problema, heurísticas são os métodos escolhidos para resolução.

2.2 Algoritmo genético de chaves aleatórias viciadas

Bean (1994) relata uma nova classe de algoritmos genéticos para problemas de otimização combinatória onde soluções podem ser representadas como vetores de permutação. Esses algoritmos, chamados de algoritmos genéticos de chaves aleatórias, ou *RKGA*(do inglês, *random-keys genetic algorithms*), representam uma solução do problema de otimização com um vetor de chaves aleatórias, onde cada chave é representada por um número real gerado de forma aleatória com valores entre [0,1]. Para mapear um vetor de chaves aleatórias em uma solução de problema de otimização e calcular o custo da solução, utiliza-se o *decodificador*. O decodificador é um algoritmo determinístico que pega um cromossomo como entrada e o associa com uma solução para o problema de otimização combinatória para o qual o valor objetivo ou a aptidão podem ser computados(GONçALVES; RESENDE; COSTA, 2014). O decodificador de Bean (BEAN, 1994) ordena os itens do vetor de chaves e a partir disso, gera uma permutação que corresponde aos indíces dos elementos que foram ordenados.

Dada uma população p inicial de vetores de chaves aplicado ao princípio darwinista, o *RKGA* evolui essa população onde cada indivíduo mais forte tem mais chance de encontrar um parceiro e com isso compartilhar do material genético. O algoritmo começa com p sendo a população de vetores inicial e com n chaves aleatórias, produzindo uma série de populações(RESENDE, 2011). Na k-ésima geração, os p vetores da população são particionados em um conjunto pequeno p_e< $p/2$ vetores correspondentes às *elites*, isto é, as melhores soluções encontradas. O conjunto com o restante da população é chamada de *não-elite*. Sem nenhuma alteração, os vetores elite são copiados para a população da $k+1$-ésima geração. Em seguida, p_m vetores de chaves aleatórias são introduzidos na população da $k+1$-ésima geração. Esses vetores, chamados de *mutantes*, tem como papel evitar que a população convirja para um ótimo local não global. Completando os p elementos da população da $k+1$-ésima geração, $p-p_e-p_m$ vetores são gerados com a combinação de pares de soluções da população da k-ésima geração(RESENDE, 2011), escolhidos aleatoriamente, com a combinação uniforme parametrizada conforme visto em (SPEARS; JONG, 1995). No esquema de Spears e Jong (1995), sejam a e b os vetores escolhidos como pais e c como o filho resultante. Logo, $c[i]$ o i-ésimo item do vetor filho, recebe a i-ésima chave de um dos pais. Recebe a chave $a[i]$ tendo P_a como sua probabilidade e $b[i]$ com probabilidade $P_b = 1 - P_a$.

Utilizando as premissas do *RKGA*, (RESENDE, 2011) apresenta o algoritmo de chaves aleatórias viciadas(ou *BRKGA* do inglês, *biased random-keys algorithm*). Este algoritmo vai um pouco além no quesito darwinismo. Assim como o *RKGA*, o *BRKGA* seleciona os pais de forma aleatória. Entretanto, enquanto o *RKGA* escolhe os pais a partir da população inteira, no *BRKGA* um pai é escolhido do grupo de elite e o outro é escolhido do conjunto não-elite(há casos que pode ser da população inteira). A

probabilidade de um vetor elite no *BRKGA* de ser escolhido a cada cruzamento é $1/p_e$, o que é maior que $1/(p - p_e)$, a probabilidade de um vetor não-elite ser escolhido. Pela mesma razão,a probabilidade de um vetor elite ser escolhido no *BRKGA* é maior que $1/p$, que é a probabilidade de um dado elite ser escolhido no *RKGA*.

O *BRKGA* usa o espaço de soluções do problema de otimização combinatória indiretamente, buscando no hipercubo contínuo r-dimensional, usando o decodificador para mapear o espaço do problema onde o *fitness* é avaliado(GONçALVES; RESENDE; COSTA, 2014).

2.3 Busca Evolutiva guiada por agrupamento

Clustering search (*CS*) é uma maneira genérica de combinar metaheuristícas de otimização com algoritmos de agrupamentos visando detectar áreas promissoras no espaço de busca para que essas áreas sejam exploradas por heurísticas específicas do problema (OLIVEIRA; LORENA, 2007). O *CS*, dinamicamente divide o espaço de busca em agrupamentos, ou *clusters*. Esses *clusters* são criados a partir de soluções candidatas apresentadas pela metaheurística de suporte.

Busca evolutiva guiada por agrupamento ou *Evolutionary clustering search* (*ECS*) é um algoritmo evolutivo híbrido que usa a estrutura do CS para explorar o espaço de busca. Isso possibilita que haja um controle melhor tanto da intensificação por busca local (OLIVEIRA; LORENA, 2004), quanto a velocidade de convergência(COSTA; OLIVEIRA; LORENA, 2010).

O *ECS* busca por regiões de pesquisa promissoras enquadrando-as por agrupamentos que são representados por centros c. O número máximo desses agrupamentos podem ser definidos de forma fixa ou dinamicamente de acordo com a largura das regiões de busca(tamanho do problema em questão)(OLIVEIRA; LORENA, 2004). A métrica de distância determina a cobertura do *cluster*, calculando a semelhança entre uma determinada solução e o centro do *cluster c*. Algumas da métricas mais populares para aplicações de *clustering search* são a distância de Hamming e Euclidiana(OLIVEIRA; LORENA, 2004)

O *ECS* apresenta quatro parte conceitualmente independentes. O componente *algoritmo evolutivo* age como um gerador contínuo de soluções candidatas, evoluindo uma população independente dos outros componentes presentes na estrutura do *CS*. Os indivíduos selecionados são apresentados ao *agrupador iterativo*, outro componente do *ECS*. Este componente promove o agrupamento desses indivíduos em um dos grupos já identificados, ou então se cria um novo *cluster*, de acordo com a medida de similaridade empregada. Cada *cluster* ativo recebe votos e pode ser melhor explorado posteriormente. Um *cluster* não ativo pode ser removido e a respectiva região de

pesquisa é esquecida (OLIVEIRA; LORENA, 2004).

Tendo $\varphi_j = (j = 1, 2, ...)$ como os *clusters* correntes, a regra que define quando um novo cluster deve ser criado é :

$$c_{new} = s_k \text{ se } \wp(s_k, c_j) > r_j, \forall \varphi_j, \text{ ou} \tag{2.1}$$

Quando um indivíduo tem semelhança bem próxima ao do *cluster* existente, a regra de assimilação é aplicada ao centro do *cluster* ativado, c_i e ao indivíduo semelhante, s_k, produzindo em um novo posicionamento do centro c_i' (FONSECA, 2018):

$$c_i' = c_i \oplus \beta(s_k \ominus c_i), \text{ contrário .} \tag{2.2}$$

Onde as operações abstratas $\oplus$ e $\ominus$ sobre c_i e s_k representam, respectivamente, adição e subtração de soluções. A operação $(s_k \ominus c_i)$ representa o quão distante são as soluções s_k e c_i, levando em consideração a métrica de distância. O β representa a taxa de aprendizado desta diferença e é usada para atualizar c_i para c_i'

Uma vez que os *clusters* devem enquadrar e representar uma região de busca onde existe uma sobreamostragem de soluções candidatas, identificando provavelmente uma região de busca promissora, a assimilação tem papel importantíssimo no processo de agrupamento (FONSECA, 2018).

O *Módulo Analisador* examina os recebidos por cada *cluster* em intervalos de geração regulares, indicando se são promissores.O número de votos recebidos recentemente no *cluster* é a densidade. Em Oliveira e Lorena (2004), o analisador também pode ser responsável pela remoção de *clusters* de baixa densidade. Por fim, a *Busca Local* é o componente que fornece um mecanismo de exploração em supostas áreas promissoras, enquadradas pelos *clusters* mais votados.

2.4 Busca evolutiva guiada por agrupamentos de chaves aleatórias viciadas

O algoritmo baseado em busca evolutiva guiada por agrupamentos de chaves aleatórias ou *BRKeCS*(do inglês, *Biased random-keys evolutionary clustering search*) usa as estruturas do *BRKGA*, porém com modificações nas operações de cruzamento e de busca local para melhor adequação ao problema em questão.

O *BRKGA* facilita a simplificação de alguns componentes de CS. É necessário implementar o decodificador e a heurística de busca local, apresentado na Figura 3.

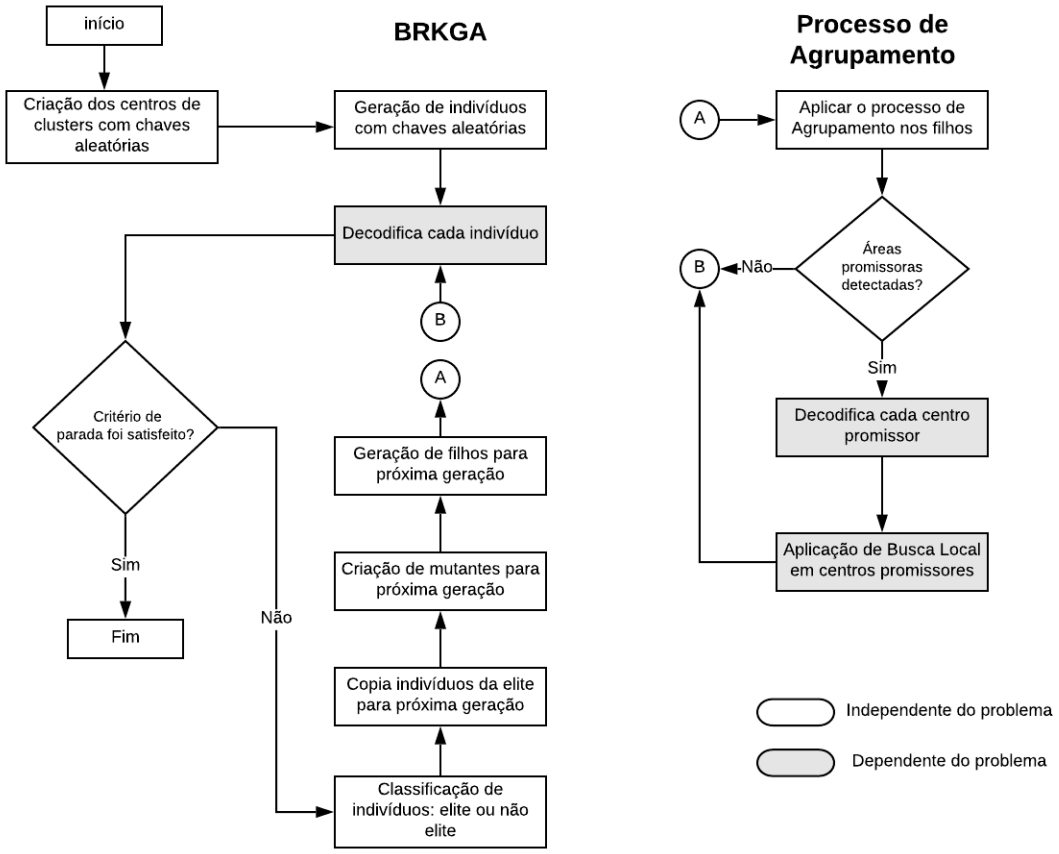

Figura 3 – Projeto conceitual do *BRKeCS*
Fonte:(FONSECA (2018))

2.4.1 Cruzamento

O principal operador evolutivo da metaheurística usa a subestrutura de dois cromossomos para produzir novas estruturas. Associado a cada cromossomo, têm-se um nível de aptidão que está correlacionado ao valor da função objetivo correspondente à solução que ele codifica (FONSECA, 2018). Contrário ao *BRKGA* regular, o *BRKeCS* utiliza o *Blend Crossover* (BLX- α) (ESHELMAN; SCHAFFER, 1993). No BLX- α, dados os cromossomos, p_1 e p_2, o cromossomo q é produzido por $q = p_1 + \alpha(p_2 - p_1)$, onde $\alpha \simeq \cup(-\alpha, 1 + \alpha)$ com $\cup$, que representa uma distribuição uniforme. Os valores de α são entre 0,5 ou 0,25, com um comportamento mais ou menos diversificado, respectivamente.

O BLX- α tende a gerar indivíduos no interior do hiperplano convexo centrado no segmento de reta traçado entre os dois individuos pais, com probabilidade uniforme, admitindo uma expansão proporcional do α usado. Se ambos os pais estão perto, vários cruzamentos sucessivos dão um comportamento mais local ao algoritmo. Entretanto, se os pais estão distantes, a busca se torna mais global (ESHELMAN; SCHAFFER, 1993).

2.4.2 Assimilação

Na medida que um *cluster* é ativado por um indivíduo gerado pela metaheurística, a operação de assimilação ocorre. Quando essa ativação ocorre, causa uma pertubação localizada no centro, proporcional à distância do indivíduo causador para o indivíduo assimilado(OLIVEIRA; LORENA, 2004).

Neste trabalho, a assimilação usada é a *Path-Relinking*(LAGUNA; MARTI, 2012). O *Path-Relinking* gera diversas soluções candidatas uniformemente espaçadas no caminho entre o centro do *cluster* e o indivíduo assimilado. O processo de assimilação corresponde a uma busca local interna aos *clusters*, usando parâmetros de início e fim de busca, retornando ao final, a melhor solução encontrada a ser assumida como novo centro(FONSECA, 2018).

A implementação do *BRKeCS*, especificamente na codificação de chaves viciadas, permite que a amostragem de soluções candidatas geradas *path-relinking* ocorra no espaço em $\mathbb{R}$, onde é mais intuitivo traçar um caminho formado por pontos a serem avaliados pela função objetivo. Entretanto, a avaliação de aptidão de cada ponto precisa passar pelo decodificador(FONSECA, 2018).

3 Proposta

Neste capítulo são apresentados modelos matemáticos e métodos computacionais propostos para resolução do *MOSP*. Primeiramente é apresentado a modelagem matemática do problema de sequenciamento de padrões *MOSP*. Além disso, a formulação da função de aptidão proposta por (OLIVEIRA; LORENA, 2002). Após as formulações matemáticas, apresentam-se as descrições das estruturas vitais do *BR-KeCS*: codificador, decodificador e a heurística de busca local. Por fim, a descrição e a configuração dos parâmetros de entrada utilizados são exibidos.

3.1 Modelagem do problema

Como apresentado no capítulo 2, o MOSP tem como objetivo minimizar a quantidade de pilhas abertas ao mesmo tempo, maximizando o tempo de trabalho, evitando assim trocas frequentes entre pilhas.

O *MOSP* pode ser modelado como uma matriz binária M de ordem I x J com linhas representando os itens e as colunas representam os padrões de corte, assim o elemento M_{ij} terá valor 1 se o item i está no padrão j (YANASSE; SENNE, 2010). Os itens e padrões variam de 1 a I e 1 a J, respectivamente. A permutação s representa a matriz na sua disposição atual, onde $s(j)$ representa a posição que o padrão j ocupa na prossecução. O conjunto de todas as permutações é representado por γ.

A partir da matriz M, uma matriz de pilhas abertas (M^1) é gerada. Onde m^1 é basicamente uma cópia de M, onde checa-se em cada linha se existe 0s entre 1s. Se existir, estes serão substituídos por 1s. Assim, M^1 apresenta uma propriedade conhecida por *1s consecutivos*. Cada elemento m^1_{ij} de M^1 será preenchido da seguinte forma:

$$m^1_{ij} = \begin{cases} 1 \text{ se } \exists u, \exists v \mid s(u) \leq j \leq s(v) \text{ e } m_{iu} = m_{iv} = 1 \\ 0 \text{ caso contrário} \end{cases} \tag{3.1}$$

Na figura 4, a matriz M^1 é gerada para sequência s = (1, 2, 3, 4) da matriz M. As posições em destaque representam os elementos m^1_{ij} intercalados por 1s na mesma linha, que apresentavam 0 e foram substituídos.

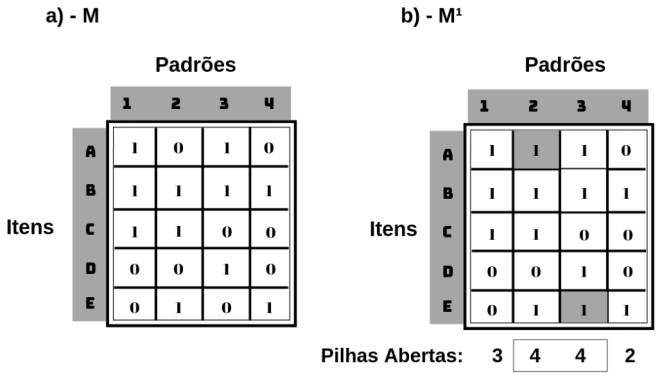

Figura 4 – Matriz de pilhas abertas M^1 gerada a partir de M
Fonte:(Gonçalves, Resende e Costa (2014))

A cada sequência s, o número máximo de pilhas abertas simultaneamente, definido como $MOS(s)$, é dado pelo maior número de 1s por coluna. Definido $MOS(s)$, teremos :

$$MOS(s) = \max_{j \in (1,...,J)} \sum_{i=1}^{I} m_{ij}^{1} \tag{3.2}$$

Assim, tendo como objetivo encontrar uma solução que minimize o número de pilhas abertas ao mesmo tempo em uma mesma permutação s, teremos o *MOSP*:

$$MOSP = \min_{s \in \gamma} MOS(s) \tag{3.3}$$

Para melhor representação, a Figura 2 mostra uma instância do MOSP modelada na matriz M, seguida da matriz M^1. Para a sequência s = (1, 2, 3, 4). Em M^1 mostra-se a permutação ótima da matriz M, onde s^* = (3,1,2,4), logo $MOS(s*)$ = 3.

3.2 Função de Aptidão

O processo evolutivo requer uma medida de aptidão da solução ou medida de qualidade. Uma função de aptidão, ou *fitness* foi proposta por Oliveira e Lorena (2002). Seja g uma função de *fitness* que reflete o custo total de uma permutação de padrões e s_k o esquema a ser avaliado. Para aumentar a diferenciação de aptidão

entre os indivíduos de uma população, é definido na formulação que a minimização de *MOS* como objetivo primário e a minimização de *TOS* como secundária. Assim sendo, definido como:

$$g(s_k) = I.J.MOS(s_k) + TOS(s_k), \text{ou} \tag{3.4}$$

$$g(s_k) = I \ . \ J \ . \max_{j \in (1,...,J)} \sum_{i=1}^{I} m_{ij}^1 + \sum_{i=1}^{I} \sum_{j=1}^{J} m_{ij} \tag{3.5}$$

O produto entre I e J é um peso de reforço da primeira parte do objetivo que é o máximo de pilhas abertas simultaneamente (MOS) e deixando-o proporcional à segunda parte do objetivo referente ao tempo de pilhas abertas(TOS)(OLIVEIRA; LORENA, 2002). Como o objetivo deste trabalho é encontrar o número máximo de pilhas abertas, o valor do *MOS* é o valor que interessa. Para melhor visualização, podemos ter como exemplo a figura 2 onde o *MOS* é igual a 3. O *TOS* representa o tempo em que as pilhas permanecem abertas, podendo ser calculado pela soma de todos os 1s presentes em M_{ij}. Esse seria um outro objetivo no *MOSP*: fazer com que as pilhas se fechem o mais rápido possível, permitindo assim, que os pedidos dos clientes sejam realizados e disponibilizados.

3.3 Codificador e decodificador

Um cromossomo codifica uma solução para o problema como um vetor de chaves aleatórias. Uma chave aleatória é um número real aleatório no intervalo contínuo de [0,1). Diretamente representado, um cromossomo é tido como uma solução do problema original e é chamado de genótipo, enquanto que em uma representação indireta não representa uma solução, logo são necessários procedimentos especiais para obter dele uma solução, chamada de fenótipo. No contexto atual, as soluções serão representadas indiretamente por parâmetros que no mais tardar serão usadas por um procedimento de decodificação para obtenção da solução(fenótipo).

Uma solução para o *MOSP* é representada indiretamente pela estrutura do cromossomo :

$$chromossome = (gene_1, ..., gene_j), \tag{3.6}$$

onde j representa o número de padrões. O mapeamento, ou decodificação dos j genes de cada cromossomo em uma sequencia de inserção de padrão ou *PIS*(do inglês, *pattern insertion sequence*), que serão usados pelo gerador de solução, é efetuado pela ordenação dos padrões em ordem crescente dos valores dos genes correspondentes (GONçALVES; RESENDE; COSTA, 2014). Um exemplo do processo

de decodificação é visto na Figura 5. Nesse exemplo, tem-se cinco padrões. Os genes ordenados correspondem a *PIS* = (1, 2, 4, 5, 3).

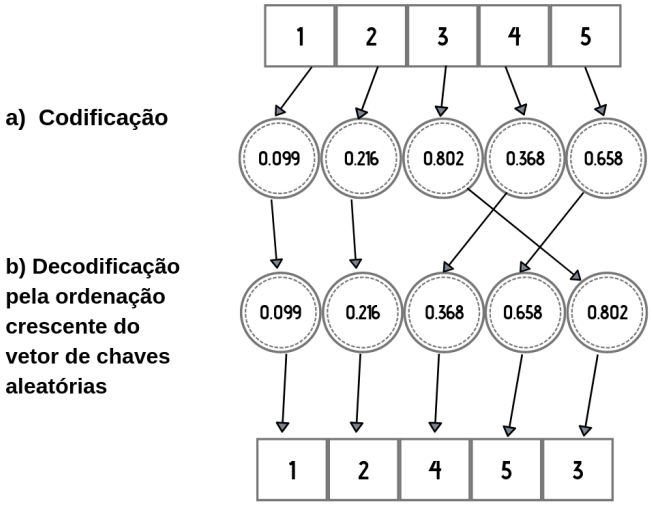

Figura 5 – Exemplo de codificação e decodificação de chaves aleatórias
Fonte:(Autor)

Neste trabalho, utiliza-se o algoritmo genético híbrido, o *BRKeCS*. O codificador e decodificador utilizados seguem a premissa da Equação 3.7(FONSECA, 2018) que basicamente remete às definições descritas nesta seção.

$$\Phi(\mathbb{R}^n) \rightarrow \mathbb{N} \tag{3.7}$$

Onde n é representa o tamanho do indivíduo(FONSECA, 2018).

3.4 Busca local

Em ciência da computação, busca local é uma heurística para resolução e problemas de otimização. Busca local pode ser usado em problemas formulados para encontrar a solução onde o critério é a maximização entre as soluções candidatas. Os algoritmos de busca local percorrem solução por solução no espaço de soluções candidatas (espaço de busca), aplicando mudanças locais até que a solução seja a ótima ou o critério de parada seja satisfeito. Como o *BKReCS* emprega a estrutura do *CS* para explorar o espaço de busca, usa-se a busca local para controlar melhor tanto a intensificação, quanto a rapidez na convergência(COSTA; OLIVEIRA; LORENA, 2010).

A heurística *2-Opt*, proposta por Croes (1958) é utilizada neste trabalho para exploração da vizinhança baseando-se nos indivíduos promissores que forem encontrados pelo *BKReCS*. O que o algoritmo realmente faz, é melhorar gradualmente uma resposta possível (busca local), inicialmente dada, até que ela atinja um ótimo local e nenhuma melhoria possa ser feita. Melhorias são feitas usando o que o autor chama de inversões.

A heurístia 2-Opt foi proposta primeiramente por Croes (1958) baseando-se em trocas entre pares de arestas de grafos que representem soluções para problemas de permutação. Esse movimento de troca remove duas arestas, anulando o percurso em dois caminhos, e os reconecta da outra maneira possível. São verificadas todas as possíveis trocas para o grafo, realizando aquela que fornecer o melhor ganho. Movimentos 2-Opt somente são realizados se as novas arestas incluídas possuírem custo menor que as removidas. E esperado que a substituição de arestas de maior custo por outras de menor custo reduza o custo total do circuito ou seja a distância total entre os pontos. Esta técnica pode ser aplicada em qualquer problema de sequenciamento de padrões(OLIVEIRA; LORENA, 2007)

.

3.5 Parâmetros do BRKeCS

O conjunto de parâmetros de desempenho do BRKeCS pode ser dividido em 3 grupos: pârametros da busca local, parâmetros da metaheurística e parâmetros de agrupamento(FONSECA, 2018).

1. Parâmetros da busca local *2-opt*:
    - *altura*: Parâmetro responsável por definir um número de trocas realizadas pela busca local 2-opt dentro de um mesmo segmento de largura;
    - *largura*: Responsável por definir o segmento de tarefas dentro de uma solução do MOSP a qual seriam realizadas possíveis trocas;
    - *rmax*: esse parâmetro define o número de vezes que a busca local é realizada em um mesmo *cluster* tido como promissor.
2. Parâmetros do BRKeCS:
    - P_e: porcentagem da população seguinte que será preenchida com os melhores indivíduos da população;
    - P_m: porcentagem da população seguinte que será formada por modificações, isto é, mutações nos indivíduos da anterior que não pertencem a elite;

- P: número de indivíduos amostrados em cada geração.

3. Parâmetros do CS:

   - *numcl*: quantidade de *clusters* amostrados em cada geração;
   - λ: porcentagem de indivíduos em um cluster com relação ao tamanho da população para que o mesmo seja considerado promissor.

Em FONSECA (2018), para encontrar os melhores valores dos parâmetros de entrada para resolução do *FlowShop Permutacional* via BRKeCS, o autor utiliza uma técnica de de sintonização automática de metaheurísticas que emprega a abordagem de corrida e validação cruzada, realizando a análise de diferentes ajustes de parâmetros entre diferentes grupos de instâncias, afim de possibilitar a racionalização do tempo de execução do experimento. Esta técnica é a *Cross Validated Racing* (CVR).

O autor realizou testes com a comparação da metaheurística sintonizada pelo *CVR* com outras heurísticas projetadas para o problema de sequenciamento. A tabela 2, apresenta os paramêtros encontrados pelo *CVR*.

Tabela 2 – Configuração de parâmetros gerada pelo *CVR*

altura	*largura*	p_e	p_m	*rmax*	λ	p	*numcl*
5	5	0,3003	0,2818	10	0,598	733	13

Esta configuração de parâmetros obteve o melhor desempenho para resolução do *Flowshop Permutacional* via *BRKeCS*. Como o *CVR* foi utilizado para sintonizar um grupo de heurísticas para o problema de seuqenciamento, para este trabalho, serão os parâmetros utilizados para resolução do *MOSP*, pois se trata de um problema de sequenciamento .

4 Resultados

4.1 Apresentação das instâncias

Para os testes computacionais foram usadas as mesmas instâncias utilizadas no *First Constraint Modeling Chalenge* (SMITH; GENT, 2005), competição que tinha como desafio minimizar o número máximo de pilhas abertas, descritas na tabela 3. O algoritmo está implementado em C e cada instância foi executada dez vezes e tirado a média como resultado. A realização dos experimentos ocorreram no cluster da *Central de Computação de Alto Desempenho* (CCAD), localizada na Universidade Federal do Maranhão - *UFMA* e gerenciada pelo *Laboratório de Aprendizagem Computacional, Métodos de Otimização e Robotica* (*LACMOR*).

Tabela 3 – Instâncias do *MOSP*

Classe	Descrição	Fonte
Harvey	2130 instâncias aleatórias propostas por Harvey. Três referências são propostas("wbo", "wbp"e "wbop"), cada uma contendo 10 classes;	Smith e Gent (2005)
Simonis	3620 instâncias aleatórias propostas por Simonis. Agrupadas em 10 classes.	Smith e Gent (2005)
Shaw	Uma classe de 25 instâncias aleatórias propostas por Shaw, com 20 padrões e tipos de itens.	Smith e Gent (2005)
Miller e Wilson	Wilson apresenta 12 instâncias (GP1-GP4, NWRS1-NWRS6 e SP1-SP2) Miller apresenta uma instância.	Smith e Gent (2005)
Faggioli & Bentivoglio	300 instâncias aleatórias contendo n = 10, 20, 30, 40, 50 e m = 10, 15, 20, 25, 30, 40. Cada uma das nxm combinações foram replicadas dez vezes.	Faggioli e Bentivoglio (1998a)
SCOOP	8 instâncias de duas empresas de corte de madeira. Denotadas de "A"e "B"	Dísponivel em : http:// www.scoop-project.net

As instâncias passaram por um pré-processamento na leitura, onde foram invertidas as matrizes de entrada. A razão para isso, dá-se pela representação da matriz mais recente onde as linhas representam os itens e as colunas representam os padrões, visto em Gonçalves, Resende e Costa (2014). A adaptação ocorreu em todas as instâncias presentes na tabela 3.

Em cada tabela apresenta a seguir, os valores da melhor solução encontrada pelo *BRKGA* e pelo *BRKeCS*. A classe Faggioli Bentivoglio apresenta resultados também do *OPT*, algoritmo proposto em Faggioli e Bentivoglio (1998b). Ademais, nas tabelas encontra-se o valor de *gap*. Cada *gap* g é calculado pela Equação 4.1:

$$g = (\frac{Xa - Xm}{Xm}) * 100 \tag{4.1}$$

Onde Xa representa o valor da solução atual a ser calculado e Xm representa a melhor solução conhecida para aquela instância. O cálculo do *gap* implica no desempenho de um determinado algoritmo em relação ao(s) outro(s), onde a melhor solução encontrada é a melhor solução entre os algoritmos em questão de uma determinada

instância. Por exemplo, na Tabela 4, a instância 10x10 da classe Simonis, apresenta *MOSP* igual a 8,0 para o *BRKGA* e 8,03 para o *BRKeCS*. Assim, calculando o *gap* do BRKGA, temos que a solução atual e a melhor solução conhecida entre os algoritmos utilizados é igual a 8,0. Logo, o valor do *gap* do *BRKGA* tende a zero, pois a solução atual também é a melhor solução conhecida. Seguindo essa linha de raciocínio, calculando o *gap* do *BRKeCS* temos que a solução atual é 8,03 e a melhor solução conhecida é 8,0 (solução do *BRKGA*). Assim, o valor de *gap* do *BRKeCS* é de 0,4 (em porcentagem), no que resulta num desempenho inferior. Infere-se dizer que quanto maior o valor do *gap*, inferior é o desempenho do algoritmo em relação ao outro.

4.2 Resultados computacionais

As tabelas com os resultados obtidos são compostos do nome da classe, *n* que remete ao número de linhas e *m* quantidade de colunas. O *MOSP* remete a melhor solução encontrada (tirada por meio da média), o tempo de execução em segundos T, e o *gap*. Os valores em negrito representam onde o *BRKeCS* se sobressaiu em relação ao *BRKGA*. Os resultados do *BRKGA* e do *OPT* apresentados foram retirados da literatura, como apresenta Gonçalves, Resende e Costa (2014).

Tabela 4 – Tabela com instâncias Simonis, Shaw e Harvey

				BRKGA			BRKeCS		
	n	m	elementos	MOSP	T(s)	gap	MOSP	T(s)	gap
Simonis	10	10	550	8,0	0,0	0,0	8,03	0,0	0,4
	10	20	550	8,9	0,0	0,0	8,9	0,0	0,0
	15	15	550	12,9	0,0	0,8	**12,8**	0,0	0,0
	15	30	550	14,0	0,0	0,0	14,0	0,0	0,0
	20	10	220	15,9	0,0	0,6	15,8	0,0	0,0
	20	20	550	18,0	0,0	0,0	18,0	0,0	0,0
	30	10	220	24,0	0,0	0,4	**23,9**	0,0	0,0
	30	15	110	26,0	0,0	0,4	**25,9**	0,0	0,0
	30	30	220	28,3	1,30	0,0	28,4	0,1	0,4
	40	20	110	36,4	0,43	0,0	36,5	0,0	0,3
Shaw	20	20	25	13,7	0,0	0,0	13,7	0,0	0,0
wbo	10	10	40	5,9	0,0	0,0	6,5	0,0	10,2
	10	20	40	7,4	0,0	0,0	9	0,0	21,6
	10	30	40	8,2	0,0	0,0	8,6	0,0	4,9
	15	15	60	9,4	0,0	0,0	9,6	0,0	2,1
	15	30	60	11,6	0,0	0,0	11,9	0,0	2,6
	20	10	70	12,9	0,0	0,0	12,9	0,0	0,0
	20	20	90	13,7	0,0	0,0	13,8	0,0	0,7
	30	10	100	20,1	0,0	9,2	**18,4**	0,0	0,0
	30	15	120	21,0	0,0	1,0	**20,8**	0,0	0,0
	30	30	140	22,6	1,08	0,0	22,6	0,0	0,0
wbop	10	10	40	6,8	0,0	0,0	7	0,0	2,9
	10	20	40	8,1	0,0	0,0	8,1	0,0	0,0
	10	30	40	8,6	0,0	0,0	9,0	0,0	4,7
	15	15	60	10,4	0,0	5,1	**9,9**	0,0	0,0
	15	30	60	12,2	0,0	0,0	12,2	0,0	0,0
	20	10	40	14,3	0,0	12,6	**12,7**	0,0	0,0
	20	20	90	14,9	0,0	3,5	**14,4**	0,0	0,0
	30	10	40	22,5	0,0	17,8	**19,1**	0,0	0,0
	30	15	60	22,4	0,0	9,3	**20,5**	0,0	0,0
	30	30	140	23,8	0,0	2,1	**23,3**	0,0	0,0
wbp	10	10	40	7,3	0,0	13,2	**6,45**	0,0	0,0
	10	20	70	8,7	0,0	4,8	**8,3**	0,0	0,0
	10	30	100	9,3	0,0	3,3	**9,0**	0,0	0,0
	15	15	60	11,1	0,0	9,9	**10,1**	0,0	0,0
	15	30	120	13,1	0,0	3,1	**12,7**	0,0	0,0
	20	10	40	15,1	0,0	18,0	**12,8**	0,0	0,0
	20	20	90	15,4	0,0	6,9	**14,4**	0,0	0,0
	30	10	40	23,2	0,8	18,1	**19,65**	0,0	0,0
	30	15	60	23,0	0,19	13,9	**20,2**	0,0	0,0
	30	30	140	24,5	0,98	5,2	**23,3**	0,0	0,0

Tabela 5 – Instâncias presentes em Faggioli Bentivoglio

n	m	OPT	gap	BRKGA	gap	BRKeCS	gap
10	10	5,5	0,0	5,5	0,0	5,5	0,0
	15	6,6	0,0	6,6	0,0	6,6	0,0
	20	7,5	0,0	7,5	0,0	7,5	0,0
	25	8,0	0,0	8,0	0,0	8,0	0,0
	30	7,8	0,0	7,8	0,0	7,8	0,0
	40	8,4	0,0	8,4	0,0	8,4	0,0
20	10	6,2	0,0	6,2	0,0	6,2	0,0
	15	7,2	0,0	7,2	0,0	7,2	0,0
	20	8,5	0,0	8,5	0,0	8,5	0,0
	25	9,8	0,0	9,8	0,0	10,0	2,0
	30	11,1	0,0	11,1	0,0	11,3	1,8
	40	13,0	0,0	13,0	0,0	13,3	2,3
30	10	6,1	0,0	6,1	0,0	6,1	0,0
	15	7,4	0,0	7,4	0,0	7,4	0,0
	20	8,8	0,0	8,8	0,0	9,6	9,1
	25	10,5	0,0	10,5	0,0	10,8	2,9
	30	12,2	0,0	12,2	0,0	12,4	1,6
	40	14,5	0,0	14,5	0,0	15,0	3,4
40	10	7,7	0,0	7,7	0,0	7,7	0,0
	15	7,3	1,4	7,2	0,0	7,3	1,4
	20	8,5	0,0	8,5	0,0	8,7	2,4
	25	10,5	0,0	10,5	0,0	10,6	1,0
	30	12,1	0,0	12,1	0,0	12,3	1,7
	40	15,0	0,7	14,9	0,0	15,6	4,7
50	10	8,2	0,0	8,2	0,0	8,2	0,0
	15	7,4	0,0	7,4	0,0	7,4	0,0
	20	7,9	0,0	7,9	0,0	8,0	1,3
	25	10,0	0,0	10,0	0,0	10,2	2,0
	30	11,2	0,0	11,2	0,0	11,5	2,7
	40	14,6	0,0	14,6	0,0	15,5	6,2

Tabela 6 – Instâncias apresentadas por Miller e Wilson

			BRKGA			BRKeCS		
	n	m	MOSP	T(s)	gap	MOSP	T(s)	gap
Miller	20	40	13	0,0	0	13	0,0	0,0
GP1	50	50	45	0,0	0	49	0,0	8,9
GP2	50	50	40	0,0	0	47,9	0,0	19,8
GP3	50	50	40	0,0	0	47,9	0,0	19,8
GP4	50	50	30	0,0	0	46,9	0,0	56,3
NWRS1	10	20	3	0,0	0	6	0,0	100,0
NWRS2	10	20	4	0,0	0	7	0,0	75,0
NWRS3	15	25	7	0,0	0	11	0,0	57,1
NWRS4	15	25	7	0,0	0	11,2	0,0	60,0
NWRS5	20	30	12	0,0	0	17,1	0,0	42,5
NWRS6	20	30	12	0,0	0	17,1	0,0	42,5
SP1	25	25	9	0,0	0	**7**	0,0	0,0
SP2	50	50	19	0,0	0	20,3	0,0	6,8

Tabela 7 – Instâncias da empresa "A"

	n	m	BRKGA	gap	BRKeCS	gap
A AP-9,d3	20	16	6	20,0	**5**	0,0
A AP-9d6	31	20	5	25,0	**4**	0,0
A AP-9,d10	20	13	6	13,2	**5,3**	0,0
A AP-9,d11	27	21	6	0,0	7	16,7

Tabela 8 – instâncias da empresa "B"

	n	m	BRKGA	gap	BRKeCS	gap
B-GTM18A-139	24	20	5	0,0	6	16,7
B-CARLET-137	14	12	5	0,0	5	0,0
B-18CR1-33	20	18	4	0,0	5	0,0
B-22X18-50	14	11	10	42,9	**7**	0,0

Na Tabela 4 o BRKeCS apresentou melhoria em quatro instâncias propostas por Simonis (SMITH; GENT, 2005). Nas instâncias propostas por Shaw, o BRKeCS obteve duas melhorias em relação ao BRKGA. Por fim, das instâncias propostas por Harvey, 18 obtiveram resultados bem mais colocados do que o BRKGA.

Na Tabela 5 o BRKGA e o OPT obtiveram os melhores desempenho comparados ao BRKeCS. na execução dessas instâncias, o BRKeCS obteve seu pior desempenho.

Na Tabela 6 as instâncias de Miller GP1-GP4, NWRS1-NWRS6 e SP2 obtiveram desempenho inferior*BRKeCS* comparado ao BRKGA. Entretanto para a instância SP1 o desempenho do *BRKeCS* superou o *BRKeCS*.

Na Tabela 7, a única instância que não obteve um resultado que competisse, insto é, que obtivesse resultados iguais ou melhores, foi a instância A AP-9,d11. O restante obteve um bom desempenho comparado ao OPT e ao BRKGA.

Na Tabela 8, a única instância que obteve um resultado que sobressaísse foi a instância B-22X18-50.O restante obteve um desempenho igual ou inferior comparado ao OPT e ao BRKGA.

Tabela 9 – Médias entre classes de instâncias

	Média gap		
	BRKGA	BRKeCS	OPT
Miller	0,0	0,0	-
Shaw	0,0	0,0	-
Simonis	0,2	0,1	-
Faggioli & Bentivoglio	0,0	1,5	10,7
Wilson	0,0	40,7	-
Harvey	5,1	1,6	-
Empresa A	14,6	4,2	-
Empresa B	10,7	4,2	-

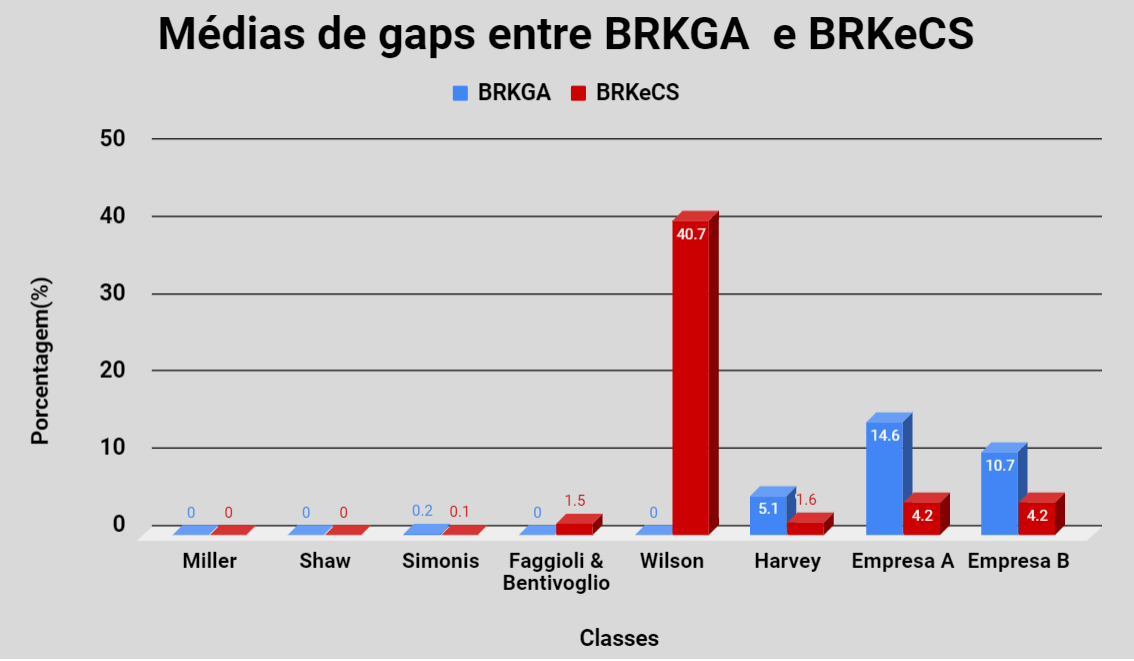

Figura 6 – Representação gráfica entre as médias de *gaps* entre *BRKGA* e *BRKeCS*
Fonte:(Autor)

Para cada classe de instâncias foi calculado a média dos *gaps*. A Tabela 9 mostra os valores encontrados para cada classe. Assimila-se que para cada valor zerado, a solução atual Xa é também a melhor solução conhecida Xm. Logo, para cada valor acima de zero, representa-se um desempenho não muito bom. O gráfico

na Figura 6 representa esses valores e apresenta uma visão de quais instâncias apresentam um valor de *gap* inferior. A representação (-) indica que as médias não foram calculadas para aquele algoritmo.

Para o *BRKeCS*, a classe Wilson e a classe Faggioli & Bentivoglio obtiveram os valores de *gap* maiores, isto é, o desempenho inferior às demais classes. Entretanto, nas classes Harvey, Simonis, Empresa A e Empresa B os valores de *gap* do *BRKeCS* foram menores, logo o desempenho relativamente melhor.

5 Conclusão

Neste trabalho, foi abordado o problema de minimização de pilhas abertas que tem como objetivo minimizar o número máximo de pilhas abertas durante o processo de corte, o MOSP. A abordagem do trabalho utiliza o *BRKGA* juntamente com a heurística de busca evolutiva guiada por agrupamentos, ou *ECS*, para resolução desse problema. O diferencial dessa metaheurística é o uso de busca local dentro do espaço de agrupamentos buscando por soluções promissoras.

Quanto aos resultados, as tabelas 4, 6, 7 e 8 apresentam resultados promissores comparados ao *BRKGA* proposto por Gonçalves, Resende e Costa (2014). O cálculo dos *gaps* ajuda a melhor visualizar o desempenho do *BRKeCS*, assim, o *BRKeCS* mostra-se um bom candidato a resolução dos problemas de sequenciamento de padrões, assim como o *BRKGA*.

A resolução do *MOSP* via *BRKeCS*, demonstra que as metaheurísticas possuem um papel importante para criação de abordagens eficientes e admissíveis para problemas de sequenciamento de padrões.

5.1 Trabalhos futuros

Resolução do problema de sequenciamento de padrões *MOSP* com a configuração de parâmetros presentes em Gonçalves, Resende e Costa (2014), afim de comparar as duas configurações via resultados das instâncias.

Resolução do problema de sequenciamento de padrões *MOSP* com a função de aptidão presentes em Gonçalves, Resende e Costa (2014), afim de comparar as duas configurações via resultados das instâncias.

Referências

BEAN, J. C. Genetic algorithms and random keys for sequencing and optimization. *ORSA journal on computing*, INFORMS, v. 6, n. 2, p. 154–160, 1994. Citado na página 20.

CARVALHO, M. A. M. d.; SOMA, N. Y. MÃsimplificados para o problema de minimizaÃ§Ãde pilhas abertas. *GestÃ ProduÃ§Ã*, scielo, v. 18, p. 299 – 310, 00 2011. ISSN 0104-530X. Disponível em: <http://www.scielo.br/scielo.php?script=sci_arttext&pid=S0104-530X2011000200006&nrm=iso>. Citado na página 17.

COSTA, T. S.; OLIVEIRA, A. C. M. de; LORENA, L. A. N. Advances in clustering search. In: SPRINGER. *Soft Computing Models in Industrial and Environmental Applications, 5th International Workshop (SOCO 2010)*. [S.l.], 2010. p. 227–235. Citado 2 vezes nas páginas 21 e 28.

CROES, G. A. A method for solving traveling-salesman problems. *Operations research*, INFORMS, v. 6, n. 6, p. 791–812, 1958. Citado na página 29.

ESHELMAN, L. J.; SCHAFFER, J. D. Real-coded genetic algorithms and interval-schemata. In: *Foundations of genetic algorithms*. [S.l.]: Elsevier, 1993. v. 2, p. 187–202. Citado na página 23.

FAGGIOLI, E.; BENTIVOGLIO, C. A. Heuristic and exact methods for the cutting sequencing problem. *European Journal of Operational Research*, Elsevier, v. 110, n. 3, p. 564–575, 1998. Citado na página 31.

FAGGIOLI, E.; BENTIVOGLIO, C. A. Heuristic and exact methods for the cutting sequencing problem. *European Journal of Operational Research*, v. 110, n. 3, p. 564 – 575, 1998. ISSN 0377-2217. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0377221797002683>. Citado na página 31.

FINK, C. *O problema de minimização de pilhas abertas-novas contribuições*. Tese (Doutorado) — Universidade de São Paulo, 2012. Citado na página 14.

FINK, C.; YANASSE, H. H.; COSTA, A. M. Análise do desempenho de variaçoes de uma formulaçao linear para o problema de minimizaçao do número máximo de pilhas abertas. *41 SIMPÓSIO BRASILEIRO DE PESQUISA OPERACIONAL*, SBPO Porto Seguro, 2009. Citado na página 14.

FONSECA, T. H. L. *Heurística para sintonização de metaheurísticas aplicada ao Problema de FlowShop Permutacional.* Dissertação (Mestrado), 2018. DEPARTAMENTO DE INFORMÁTICA/CCET. Disponível em: <https://tedebc.ufma.br/jspui/handle/tede/tede/2286>. Citado 6 vezes nas páginas 22, 23, 24, 28, 29 e 30.

GONçALVES, J. F.; RESENDE, M. G. C.; COSTA, M. D. A biased random-key genetic algorithm for the minimization of open stacks problem. *International Transactions in Operational Research*, v. 23, n. 1-2, p. 25–46, 2014. Disponível em:

<https://onlinelibrary.wiley.com/doi/abs/10.1111/itor.12109>. Citado 12 vezes nas páginas 13, 14, 15, 18, 19, 20, 21, 26, 27, 31, 32 e 38.

LAGUNA, M.; MARTI, R. *Scatter search: methodology and implementations in C*. [S.l.]: Springer Science & Business Media, 2012. v. 24. Citado na página 24.

LIMA, J. R.; CARVALHO, M. A. M. de. Métodos de descida rápida e descida em vizinhança variável aplicados à resolução do problema de minimização de pilhas abertas. 2016. Citado na página 15.

LINHARES, A.; YANASSE, H. H. Connections between cutting-pattern sequencing, vlsi design, and flexible machines. *Computers Operations Research*, v. 29, n. 12, p. 1759 – 1772, 2002. ISSN 0305-0548. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0305054801000545>. Citado na página 19.

LOPES, I. d. S. *Pattern sequencing models in cutting stock problems*. Tese (Doutorado), 2011. Citado na página 14.

OLIVEIRA, A. C.; LORENA, L. A. Detecting promising areas by evolutionary clustering search. In: SPRINGER. *Brazilian Symposium on Artificial Intelligence*. [S.l.], 2004. p. 385–394. Citado 4 vezes nas páginas 14, 21, 22 e 24.

OLIVEIRA, A. C.; LORENA, L. A. Hybrid evolutionary algorithms and clustering search. In: *HYBRID evolutionary algorithms*. [S.l.]: Springer, 2007. p. 77–99. Citado 2 vezes nas páginas 21 e 29.

OLIVEIRA, A. C. M. de; LORENA, L. A. N. 2-opt population training for minimization of open stack problem. In: SPRINGER. *Brazilian Symposium on Artificial Intelligence*. [S.l.], 2002. p. 313–323. Citado 4 vezes nas páginas 14, 25, 26 e 27.

PINTO, J. M.; YANASSE, H. O problema de corte e sequenciamento de padrões: uma abordagem integrada. 05 2019. Citado na página 17.

RESENDE, M. G. Introdução aos algoritmos genéticos de chaves aleatórias viciadas. *Simpósio Brasileiro de Pesquisa Operacional*, p. 3680–3691, 2011. Citado na página 20.

SMITH, B.; GENT, I. Constraint modelling challenge 2005. In: *IJCAI 2005 Fifth Workshop on Modelling and Solving Problems with Constraints*. [S.l.: s.n.], 2005. p. 1–8. Citado 2 vezes nas páginas 31 e 35.

SPEARS, W. M.; JONG, K. D. D. *On the virtues of parameterized uniform crossover*. [S.l.], 1995. Citado na página 20.

YANASSE, H. Pre-processing operations for the minimization of open stacks problem. 05 2010. Citado na página 18.

YANASSE, H. H.; PINTO, M. J. Uma nova formulação para um problema de sequenciamento de padroes em ambientes de corte. *XXXV SBPO-Simpósio Brasileiro de Pesquisa Operacional, Natal, RN*, p. 1516–1524, 2003. Citado na página 14.

YANASSE, H. H.; SENNE, E. L. F. The minimization of open stacks problem: A review of some properties and their use in pre-processing operations. *European Journal of Operational Research*, Elsevier, v. 203, n. 3, p. 559–567, 2010. Citado 3 vezes nas páginas 14, 19 e 25.