Uma Heurística Aplicada à Uniformidade das Características Físicas de Produtos

Luis Henrique Leão do Nascimento
Departamento de Ciência da Computação, Universidade Federal de Ouro Preto
Campus Morro do Cruzeiro, Ouro Preto, Minas Gerais, 35400-000, Brasil.
luisccm143@gmail.com

Marco Antonio Moreira de Carvalho
Departamento de Ciência da Computação, Universidade Federal de Ouro Preto
Campus Morro do Cruzeiro, Ouro Preto, Minas Gerais, 35400-000, Brasil.
mamc@iceb.ufop.br

RESUMO

Em alguns contextos industriais, existe a necessidade que os produtos fabricados possuam uma qualidade satisfatória no que tange à uniformidade das características físicas dos mesmos, de forma que o preço de revenda e sua qualidade não sejam afetados. Para atender este critério, a linha de produção deve ser planejada e otimizada visando fabricar os produtos de um mesmo lote sem que haja interrupções da produção dos mesmos. Este problema, classificado como NP-Difícil, é conhecido como o Problema de Minimização de Descontinuidades (ou *Minimization of Discontinuities Problem*, MDP). Neste trabalho são propostos uma representação em grafos para o problema, uma heurística baseada em um algoritmo clássico na teoria do grafos e a implementação de uma metaheurística para solução do MDP. Experimentos computacionais demonstram que o método proposto é competitivo e obteve novas melhores soluções em seis de nove grupos de instâncias da literatura, superando estado da arte atual.

PALAVRAS CHAVE. Indústrias, Minimização de Descontinuidades,Metaheurística.

TIND, MH, OC.

ABSTRACT

In some industrial contexts, there is a need for manufactured products to have a satisfactory quality in terms of the uniformity of their physical characteristics, so that their sale price and quality are not affected. To meet this criterion, the production line must be planned and optimized in order to manufacture the products of the same lot without interrupting the production. This problem, classified as NP-Hard, is known as the Minimization of Discontinuities Problem (MDP). In this paper, we propose a graph representation for the problem, a heuristic based on a classical algorithm in graph theory and the implementation of a metaheuristic for MDP solution. Computational experiments demonstrate that the proposed method is competitive and has obtained new best solutions for six out of nine sets of instances of the literature, outperforming the current state-of-the-art.

KEYWORDS. Industries, Minimization of Descontinuites Problem,Metaheuristic.

TIND, MH, OC.

1. Introdução

Comumente, indústrias transformam unidades maiores de matéria prima em produtos menores com diferentes formas e tamanhos. Este processo pode ser caracterizado como o Problema de Corte de Estoque, que possui como objetivo realizar cortes em uma determinada unidade de matéria prima a fim de ser obter unidades menores (ou *peças*), respeitando algum objetivo pré-estabelecido, como por exemplo, maximizar a quantidade de unidades menores cortadas, ou minimizar a sobra de matéria-prima utilizada. Neste trabalho, considera-se que estas unidades maiores de matéria prima são chapas, por exemplo, de metal, vidro, papel ou madeira.

Um *padrão* de corte é caracterizado pela disposição das peças dentro de uma chapa de matéria prima, semelhante a um gabarito, utilizado para indicar os cortes a serem realizados e as peças a serem produzidas. Cada padrão é unicamente reconhecido por possuir uma quantidade finita de peças e posições associadas a cada uma delas. A disposição de cada peça dentro destes padrões interfere diretamente nos diferentes objetivos dos problemas de corte mencionados anteriormente. A produção é dividida em *estágios*, tal que, a cada estágio, um único padrão de corte é processado.

No contexto do tema abordado neste trabalho, é interessante minimizar o número de vezes em que a produção de uma peça qualquer é interrompida, ou seja, o número de *descontinuidades* em sua produção. Uma descontinuidade surge quando uma determinada peça está sendo produzida a partir de um padrão em um determinado estágio, e em estágios posteriores a referida peça não é produzida, porém, volta a sê-lo em algum estágio posterior.

Uma descontinuidade na produção de uma determinada peça pode ser ocasionada pela necessidade de produção de outros tipos de peças, por exemplo, ao processar um padrão diferente. Ao fabricar peças diferentes, a mesma matéria prima é utilizada e, posteriormente, ao retomar a produção de um tipo de peça específica, as características da matéria prima podem ser diferentes, originando variações de características físicas. Desta forma, a fim de se maximizar a uniformidade das características físicas dos produtos, busca-se minimizar as descontinuidades durante a produção.

O Problema de Minimização de Descontinuidades (ou *Minimization Descontinuites Problem*, MDP) é um problema $\mathcal{NP}$-Difícil [Garey e Johnson, 1979] que origina-se em um ambiente de produção industrial, em que é necessário realizar o corte de um conjunto S de diferentes padrões para produzir um conjunto P de peças com demandas específicas. A produção é considerada sequencial e incremental, ou seja, em um estágio todas as cópias de um padrão de corte específico devem ser cortadas antes que um padrão de corte diferente seja processado.

A Tabela 1 apresenta um exemplo de instância MDP, dada por uma matriz binária $M = \{m_{ij}\}$ que relaciona os padrões de corte e as peças a serem produzidas tal que $m_{ij} = 1$ caso o padrão i contenha a peça j e $m_{ij} = 0$ caso contrário. No referido exemplo, o conjunto de peças, enumeradas de 1 a 6, está representado na horizontal, e o conjunto dos padrões de corte, enumerados de p_1 a p_6, está representado na vertical. O padrão p_1 é composto pelas peças 1, 2, o padrão p_2 é composto pelas peças 1, 5 e assim por diante.

Tabela 1: Instância MDP.

	p_1	p_2	p_3	p_4	p_5	p_6
1	1	1	0	0	0	0
2	1	0	1	0	0	0
3	0	0	0	1	1	0
4	0	0	0	1	0	1
5	0	1	0	0	1	0
6	0	0	1	0	0	1

Uma solução para o MDP é dada por uma permutação π das colunas da matriz M, dando origem à matriz permutação $Q^{\pi} = \{q^{\pi}_{ij}\}$, que consiste nos mesmos elementos da Matriz M, porém, com as colunas permutadas na sequência estabelecida por π. Essa representação indica qual padrão

de corte será processado em cada um dos estágios da produção, representados pelas colunas de Q^{π}. A Tabela 2 apresenta dois possíveis sequenciamentos dos padrões com base na instância definida na Tabela 1. Os valores em negrito indicam em quais estágios da produção ocorrem descontinuidades. Por exemplo, na primeira solução há descontinuidades nas produção das peças 1 e 3, ao passo em que na segunda solução, há descontinuidades na produção das peças 1, 2, 4, 5 e 6.

Tabela 2: Duas possíves soluções para o MDP.

	p_5	p_2	p_4	p_6	p_3	p_1
1	0	1	**0**	**0**	**0**	1
2	0	0	0	0	1	1
3	1	**0**	1	0	0	0
4	0	0	1	1	0	0
5	1	1	0	0	0	0
6	0	0	0	1	1	0

(a)

	p_1	p_6	p_5	p_4	p_3	p_2
1	1	**0**	**0**	**0**	**0**	1
2	1	**0**	**0**	**0**	1	0
3	0	0	1	1	0	0
4	0	1	**0**	1	0	0
5	0	0	1	**0**	**0**	1
6	0	1	**0**	**0**	1	0

(b)

No sequenciamento ilustrado na Tabela 2(a), existem duas descontinuidades, uma para a peça 1 com duração de 3 estágios, e outra para a peça 3 com duração de 1 estágio. No segundo sequenciamento, apresentado na Tabela 2(b), há descontinuidades na produção das peças 1, 2, 4, 5 e 6 respectivamente por 4, 3, 1, 2 e 2 estágios. Claramente, o sequenciamento apresentado na primeira solução é melhor do ponto de vista de minimização de descontinuidades.

Neste trabalho, propõe-se a representação do MDP por meio de grafos e sua solução pela aplicação de uma heurística e pela aplicação da metaheurística Busca Local Iterada (*Iterated Local Search* – ILS). Esta é a primeira aplicação da ILS ao MDP reportada na literatura, e, de acordo com os experimentos computacionais reportados, este método foi capaz de aprimorar os melhores resultados em seis de nove conjuntos de instâncias da literatura.

O restante do trabalho está organizado da seguinte maneira: A revisão da restrita literatura sobre o MDP é apresentada na Seção 2 e a Seção 3 detalha as contribuições propostas; o método proposto é comparado com os melhores resultados da literatura na Seção 5 e, por fim, conclusões são realizadas a respeito do trabalho na Seção 6.

2. Revisão da Literatura

O primeiro trabalho a abordar o MDP foi o realizado por Dyson e Gregory [1974], integrando-o ao problema de corte de estoque. Inicialmente, foi utilizado o método simplex revisado para solução do problema de corte de estoque. A fase de sequenciamento de padrões foi modelada como o Problema do Caixeiro Viajante (PCV), em que os padrões são representados pelos vértices do grafo, havendo arestas entre todos os pares de vértices e as distâncias são calculadas levando em consideração a quantidade de peças que não são comuns a cada par de padrões. Entretanto, essa modelagem foi descartada, principalmente devido a limitação de desempenho computacional da época para solução do PCV.

Novamente, o MDP foi abordado conjuntamente ao problem de corte de estoque por Madsen [1979]. Para a resolução do MDP foi proposta uma representação em matrizes binárias, quadradas e simétricas, as quais indicam quais padrões possuem pelo menos um par de peças em comum. A partir desta representação, foi aplicada a heurística de *Cuthill-McKee* [Cuthill e McKee, 1969], originalmente projetada para o problema de Minimização de Largura de Banda em Matrizes Simétricas. Ao comparar o método proposto com a solução empregada em uma indústria de corte de vidros, houve uma significativa melhora no desperdício relacionado ao corte de estoque, entretanto, não houve relato de melhoria das descontinuidades na produção.

Linhares e Yanasse [2002], apresentaram um estudo detalhado sobre o relacionamento dos diferentes problemas de sequenciamento de padrões de corte, entre eles, o MDP, o Problema de Minimização de Espalhamento de Ordens e o Problema de Minimização de Pilhas Abertas. Foi

provado que, apesar de compartilharem a mesma base conceitual, os problemas não são equivalentes entre si.

O MDP também é estudado na literatura sob nome de *Problema de Minimização de Blocos Consecutivos* (ou *Consecutive Blocks Minimization*, CBM). Um *bloco consecutivo* é definido como uma sequência contígua de elementos não nulos em uma mesma linha de uma matriz binária. No MDP, um bloco consecutivo representa a fabricação contínua de um tipo de peça em estágios consecutivos, de forma que dois blocos consecutivos em uma mesma linha da matriz são separados por uma descontinuidade. Garey e Johnson [1979] mostraram que o CBM pertence à classe NP-Difícil, significando que não se conhece algoritmo eficiente para sua solução.

Recentemente, Haddadi et al. [2015], propôs um método heurístico polinomial baseado em melhorias locais para o CBM, no contexto de redução das dimensões de sistemas lineares. As melhorias locais são realizadas por movimentos clássicos em buscas locais, como 2-*swap* e *shift*. O primeiro consiste em trocar de posição duas colunas da matriz Q^{π}, ao passo em que o segundo consiste em remover uma coluna de sua posição original em Q^{π} e inserí-la novamente em outra posição. Cada movimento implementado é avaliado em tempo $O(m)$, em que m representa o número de colunas em Q^{π}. Esta complexidade foi possível devido à um esquema de Δ-avaliação, proposto no mesmo trabalho e detalhado na Seção 4.2.1. Para a realização dos testes foram utilizados um conjunto de instâncias artificiais geradas pelo próprio autor e outras instâncias reais. As instâncias artificiais foram fornecidas gentilmente pelo autor para comparação com o método proposto no presente trabalho e são descritas na Seção 5.2.

3. Contribuições Deste Trabalho

Neste trabalho são propostos uma representação em grafos para o MDP, uma heurística baseada em percurso em grafos e também a implementação de uma metaheurística. A seguir, cada uma destas contribuições são detalhadas.

3.1. Representação Computacional

O MDP foi representado computacionalmente utilizando-se grafos, de maneira similar à proposta em Yanasse [1997], com uma modificação adicional. Nestes grafos não direcionados, os vértices representam as peças, havendo ligação entre dois vértices quaisquer se as peças estão presentes em um mesmo padrão. Propõe-se a adição de pesos às arestas, indicando o número de vezes em que cada par de peças é produzido a partir de um mesmo padrão, de forma que as peças que aparecem com frequência juntas em diferentes padrões, terão adjacência no grafo com um peso elevado. Esta informação é utilizada posteriormente para dar origem a um método heurístico que visa processar estas peças prioritariamente.

A Tabela 3 apresenta uma instância MDP de exemplo e a Figura 1 apresenta o grafo MDP correspondente.

	p_1	p_2	p_3	p_4	p_5	p_6
1	1	0	0	0	0	1
2	1	0	1	0	0	1
3	0	0	0	1	1	0
4	0	0	0	1	1	0
5	1	1	1	1	1	1

Tabela 3: Instância MDP

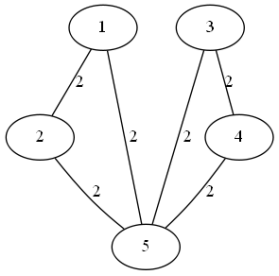


Figura 1: Grafo MDP.

O grafo MDP do exemplo é composto pelos cinco vértices que representam as cinco peças do problema. O padrão p_1 é formado pelas peças 1, 2 e 5, resultando na adjacência dos vértices que representam as peças do referido padrão. O padrão p_2 é composto pelas peças 1 e 5. Essa aresta já está presente no grafo, portanto, o peso da mesma é aumentado para duas unidades. Os padrões restantes passam pelos mesmos processos citados anteriormente, até ser obtido o grafo final.

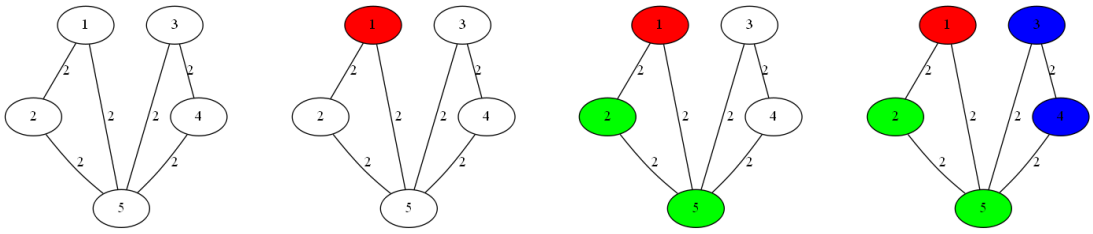
3.2. Uma Heurística Baseada em Percurso em Grafos

Dada a representação em grafos utilizada, propõe-se a utilização de um método guloso de percurso em grafos para obter uma sequência de peças, utilizada posteriormente para gerar informações importantes sobre como os padrões devem ser sequenciados. Esta estratégia tem sido utilizada com sucesso a diferentes problemas de sequenciamento de tarefas e outros correlatos [Paiva e Carvalho, 2016; Carvalho e Soma, 2015; Santos e Carvalho, 2015].

Neste trabalho, utiliza-se a Busca em Largura (ou *Breadth-First Search*, BFS) como algoritmo de percurso em grafos, com algumas modificações. Dado um vértice inicial, a BFS original explora toda a vizinhança deste vértice e repete o mesmo processo para cada um destes vértices explorados até que todo o grafo seja explorado. Caso haja mais de um componente no grafo, a BFS examina todo o componente, e logo após, um vértice presente em outro componente é selecionado para reiniciar a busca. Dessa maneira a BFS retorna uma lista com a ordem em que todos os vértices presentes no grafo foram explorados.

Especificamente na heurística proposta, a BFS seleciona como vértice inicial aquele que possui o menor grau, ou seja, aquele que possui menos adjacências com os demais vértices. Caso ocorra empate, o vértice inicial é selecionado de acordo com a ordem lexicográfica. A partir dos vizinhos do vértice inicial, o próximo vértice a ser expandido é aquele que possuir o maior grau. De maneira análoga, caso ocorra empate é selecionado o vértice de menor ordem lexicográfica. Repetem-se esse passos até que todos os vértices sejam visitados, retornando uma lista ϕ com a ordem de exploração dos vértices. O intuito por trás deste critério guloso é sequenciar as peças que são produzidas mais frequentemente em conjunto e utilizar esta informação no sequenciamento de padrões.

A Figura 2 apresenta a BFS modificada aplicada ao grafo da Figura 1. Inicialmente, nenhum vértice foi explorado, portanto a lista ϕ é vazia (Figura 2a). É preciso determinar o vértice inicial, porém os vértices 1, 2, 3, 4 possuem o mesmo grau, então por ordem lexicográfica o vértice 1 é selecionado (em vermelho) e adicionado à lista ϕ conforme a Figura 2b. Os vértices 2 e 5 (em verde), vizinhos do vértice 1 são visitados e consequentemente colocados em ordem lexicográfica em ϕ, dado que possuem o mesmo peso nas arestas, vide Figura 2c. Por fim o vértice 5 é selecionado, pois apresenta um grau maior que o vértice 2 e seus vizinhos (em azul) são visitados e colocados na lista ϕ, vide Figura 2d. Ao final, temos ϕ = [1,2,5,3,4].



(a) $\phi = []$ (b) $\phi = [1]$ (c) $\phi = [1,2,5]$ (d) $\phi = [1,2,5,3,4]$

Figura 2: Aplicação da BFS modificada no grafo de exemplo.

A lista ϕ gerada pela BFS retorna somente uma ordenação das peças, entretanto, uma solução viável para o MDP consiste na ordem de processamento dos padrões. É preciso então obter uma permutação π das colunas a partir da lista de peças ϕ. O método para este fim foi proposto por Becceneri et al. [2004] e consiste em percorrer toda a lista de peças gerada pela BFS de forma incremental, ou seja, uma peça analisada por vez, verificando se algum padrão possui somente a peça corrente ou qualquer subconjunto das peças já analisadas. Todos os padrões que atendem a algum destes dois critérios adicionados ao final de π, sem repetição. Caso mais que um padrão atenda ao critério de inserção na solução, o desempate é realizado em ordem lexicográfica.

A Tabela 4 apresenta o processo de sequenciamento de padrões dada a sequência de peças

$\phi = [1, 2, 5, 3, 4]$ gerada no exemplo anterior. Na referida tabela, a coluna ϕ indica quais peças já foram analisadas, e a coluna π indica quais padrões já foram sequenciados. Inicialmente, verifica-se que nenhum padrão possuiu somente a peça 1, a primeira em ϕ. Na segunda iteração, novamente nenhum padrão possui somente a peças 1, 2 ou ambas {1,2}. Ao analisar a peça, 5, é verificado que os padrões p_1 ,p_2, p_5 são compostos em sua totalidade por algum subconjunto das peças analisadas até então. Desta forma, os padrões são inseridos em ordem lexicográfica na solução. Na terceira iteração, a peça 3 e todos os subconjuntos de peças analisadas são verificados, de forma que os padrões p_4 e p_5 são inseridos na solução final. Como todos os padrões já foram verificados o algoritmo termina e retorna a solução π = [p_1, p_2, p_3, p_6, p_4, p_5].

Tabela 4: Sequenciamento de padrões a partir da BFS.

ϕ	π
1	
1,2	
1,2,5	p_1, p_2, p_3, p_6
1,2,5,3	$p_1, p_2, p_3, p_6, p_4, p_5$
1,2,5,3,4	$p_1, p_2, p_3, p_6, p_4, p_5$

4. Busca Local Iterada

Após a geração de uma solução inicial pela aplicação da heurística proposta, optou-se por realizar o aprimoramento da mesma usando a metaheurística Busca Local Iterada (ou *Iterated Local Search*, ILS), introduzida por Lourenço et al. [2003], que consiste em alternar entre intensificação e diversificação da busca no espaço de soluções. Esta estratégia é implementada pela aplicação iterativa de métodos de busca local e de perturbação à uma solução corrente até que atinja uma condição de parada.

Para compor este método foi utilizada a busca local por agrupamento de 1-*blocks* proposta por Paiva e Carvalho [2016] e o método 2-*opt* [Croes, 1958] como mecanismos de busca local e, como mecanismo de perturbação, o método 2-*swap*. Estes métodos são descritos nas seções subsequentes. A exploração realizada pela busca local influencia a qualidade das soluções e também o tempo de execução, ao passo que a perturbação influencia a taxa de convergência da ILS. Desta forma, optou-se por aplicar a perturbação a um percentual α da solução e por restringir a busca local a um percentual β do tamanho das vizinhanças utilizadas. A partir da composição dos métodos mecionados foi gerado o método ILS-MDP, cujo pseudo-código é apresentado no Algoritmo 1.

Algoritmo 1: ILS-MDP.

```
1  Entrada: solução inicial π, β, iter_max.
2  enquanto número de iterações<iter_max faça
3      π' ← π;
4      perturbe a solução π' em uma proporção α aplicando swap;
5      aplique o método 2-opt em π' em uma proporção β;
6      aplique a método de agrupamento de 1-blocks em π';
7      se π' melhor que π então
8          π ← π'
9      fim
10 fim
11 retorna π;
```

O algoritmo recebe a solução inicial π, a proporção de aplicação da perturbação α, a proporção de aplicação da busca local β e o número máximo de iterações *iter_max*. A cada uma

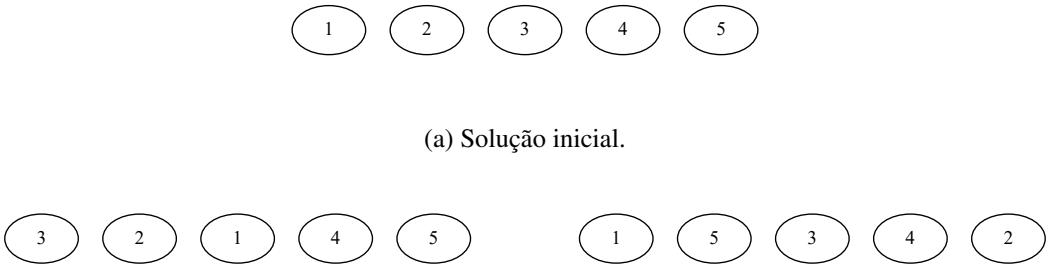
das *iter_max* iterações (laço das linhas 1 a 11), o método executa as operações de perturbação (linha 4) e busca local (linhas 5 e 6) nas proporções definidas por α e β, atualizando a solução π em caso de melhora (linhas 7 e 8). Ao final, é retornada a solução π de melhor valor obtido (linha 11).

A Seção 5.1 apresenta brevemente a definição dos parâmetros α, β e *iter_max*. A seguir, os métodos de busca local e perturbação são detalhados.

4.1. 2-*swap*

O método 2-*swap* consiste na realização de trocas de posições entre dois elementos de uma solução π, gerando uma nova solução π' que difere exatamente em 2 elementos da solução anterior. Por exemplo, considere um problema permutacional presente na Figura 3a composto por cinco elementos, e a configuração inicial $[1, 2, 3, 4, 5]$. A cada movimento realizado, dois elementos serão trocados de posição. A aplicação deste movimento neste exemplo específico produz uma uma vizinhança de 10 soluções, dada a combinação de 5 elementos tomados 2 a 2.

A Figura 3 apresenta duas possíveis soluções vizinhas com base na aplicação do 2-*swap* ao exemplo da Figura 3a. Para a primeira solução foram trocados o primeiro e o terceiro elemento, resultando na configuração $[3, 2, 1, 4, 5]$. Para a segunda solução foram trocados o segundo e quinto elemento resultando na configuração $[1, 5, 3, 4, 2]$.



(a) Solução inicial.

(b) Solução vizinha 1.

(c) Solução vizinha 2.

Figura 3: Aplicação do método *2-swap*.

A aplicação do método 2-*swap* como mecanismo de perturbação consiste em realizar movimentos de troca aleatórios, independente do valor da solução gerada. Conforme mencionado, apenas uma porcentagem α dos elementos da solução são trocados de posição.

4.2. Agrupamento de 1-*blocks*

A busca local proposta recentemente por Paiva e Carvalho [2016], denominada Agrupamento 1-*blocks*, tem como objetivo reduzir o número de blocos consecutivos de uma matriz binária. Embora tenha sido proposta originalmente para o Problema de Minimização de Trocas de Ferramentas, este método pode também ser aplicado ao MDP. Neste contexto, a redução do número de blocos consecutivos equivale a reduzir as descontinuidades na matriz Q^{π}.

O princípio deste método consiste em percorrer todas as linhas da matriz em ordem aleatória, procurando por dois ou mais blocos consecutivos. Ao encontrá-los, esses blocos são tomados dois a dois e tenta-se agrupá-los. Movimentam-se as colunas referentes ao primeiro bloco uma a uma, para antes ou depois do segundo bloco, o que resultar em menor valor da função objetivo; se ambos os movimentos piorarem a solução, esta coluna não é movimentada. Repete-se esse processo para os demais blocos da linha.

O Algoritmo 2 apresenta o pseudo-código desta busca local. O método recebe como entrada a matriz Q^{π} e, em seguida, cada linha desta matriz (linhas 1 a 11) é analisada em busca de dois ou mais 1-*blocks* (linhas 2 e 3) e então, movimenta-se todas as colunas do primeiro 1-*block*, uma a uma, para antes (representado pela função *antes*) ou depois (representado pela função

depois) das colunas do segundo 1-*block* (linhas 4 a 8). Cada coluna só é movimentada caso não haja piora no valor da solução; a não movimentação de colunas é representada pela função *nenhum*. Aplicando-se sucessivamente estas operações, busca-se agrupar os 1-*blocks* de cada linha.

Algoritmo 2: Agrupamento de 1-*Blocks*.

```
   Input: matriz Q^π
 1 para cada linha r ∈ Q^π aleatoriamente selecionada faça
 2     determine o primeiro 1-block i em r;
 3     enquanto existir um próximo 1-block j em r faça
 4         para cada coluna k em i faça
 5             movimentos ←{depois(k, j), antes(k, j), nenhum()};
 6             proximo_movimento ← movimento ∈ movimentos que resulta na melhor
                 solução;
 7             perform(proximo_movimento);
 8         fim
 9         i ← j;
10     fim
11 fim
12 retorna Q^π
```

Para exemplificar brevemente o funcionamento desta busca local, considere a matriz Q^π dada pela Tabela 2(b), obtida pela solução $\pi = [p_1, p_6, p_5, p_4, p_3, p_2]$, cujo o número de descontinuidades é 5. Inicialmente, seleciona-se a segunda linha para ser analisada e encontra-se dois blocos consecutivos, o primeiro composto pela coluna p_1 e o segundo pela coluna p_3. Avalia-se então as possíveis realocações da coluna p_1, gerando as soluções $\pi = [p_6, p_5, p_4, p_1, p_3, p_2]$ e $\pi = [p_6, p_5, p_4, p_3, p_1, p_2]$, resultando em 4 e 3 descontinuidades respectivamente. O melhor movimento é executado e a solução é atualizada para $\pi = [p_6, p_5, p_4, p_3, p_1, p_2]$. A busca local analisa as demais linhas da mesma maneira, até que toda a matriz tenha sido considerada.

Dada a representação de instâncias MDP por matrizes, a avaliação sucessiva de diferentes operações da busca local por Agrupamento 1-*blocks* pode se tornar custosa computacionalmente. Para acelerar esta busca local, empregou-se um método de avaliação rápida da movimentação de colunas, descrito a seguir.

4.2.1. Δ-Avaliação

Uma Δ-*avaliação* é um procedimento que visa avaliar de maneira rápida o valor de uma nova solução obtida por meio de uma alteração de uma solução anterior. Geralmente, esse tipo de avaliação consiste em verificar somente a parte da solução que foi alterada, na tentativa de melhorar o tempo de execução do algoritmo. No caso específico do MDP, uma função de avaliação que considere toda a matriz Q^π exige complexidade $O(|P| \times |S|)$, ao passo que a avaliação proposta por Haddadi et al. [2015] possui complexidade $\Theta(|P|)$.

Considerando um movimento de inserção da coluna b entre as colunas a e c em uma solução para o MDP, representada pela matriz Q^π, este método de avaliação consiste em averiguar em cada linha i a ocorrência de três elementos consecutivos $q_{ia}q_{ib}q_{ic}$ com valores 101 ou 010, o que implica no aumento do número de blocos consecutivos. A Equação 1 sintetiza a lógica booleana empregada.

$$(1 - q_{ia}) \times q_{ib} \times (1 - q_{ic}) + q_{ia} \times (1 - q_{iba}) \times q_{ic} \tag{1}$$

Utilizando-se operações *bit* a *bit*, é possível considerar todo o conteúdo de cada coluna em uma única comparação, realizado análises em lote ao invés de linha a linha. Desta forma, a lógica da Equação 1 pode ser compactada como $\Delta = b - ab - bc + ac$. Caso o valor de Δ seja positivo, ocorreu um aumento do número de blocos consecutivos; em caso de valor nulo, o

número de blocos permaneceu constante e, em caso de número negativo ocorreu, uma diminuição do número de blocos consecutivos.

4.3. 2-*Opt*

Proposto por Croes [1958], método 2-*opt* é aplicado em problemas permutacionais e consiste em inverter as posições dos elementos dentro de um dado intervalo. Por exemplo, seja a solução π=[p_1, p_2, p_3, p_4, p_5, p_6] e o intervalo aberto (2,6). Após a aplicação do método 2-*opt* temos π=[p_1, p_2, p_5, p_4, p_3, p_6], dada a inversão dos elementos nas posições de 3 a 5.

Neste trabalho, o método 2-*opt* foi utilizado como um método de busca local, aplicando movimentos de inversão aleatório e aceitando somente melhorias na solução. Dado o alto número de movimentos gerados a partir do 2-*opt*, optou-se do explorar apenas uma porção β desta vizinhança, definida *a priori*.

5. Experimentos Computacionais

Os métodos propostos foram implementados utilizando a linguagem C++ e compilados utilizando g++ 4.4.1 e a opção de otimização -O3. Os experimentos foram realizados em um computador com processador Intel i5 Quad Core de 3.2GHz, 8 GB de RAM, e sistema operacional Ubuntu 15.10.

Foram realizados neste trabalho dois tipos distintos de experimentos. Os experimentos preliminares, descritos na Seção 5.1, envolveram o ajuste de parâmetros utilizados na ILS. Considerando a melhor configuração do algoritmo proposto, foi realizada a sua comparação com o método de melhores resultados disponíveis na literatura, descrito na Seção 5.2.

5.1. Ajuste de Parâmetros

Conforme apresentado no Algoritmo 1, existem três parâmetros da ILS que necessitam ser ajustados a fim de se definir a sua melhor configuração: a porcentagem de aplicação de busca local, a porcentagem de aplicação de perturbação e o número de iterações executadas. Para o ajuste destes parâmetros foi utilizado o *irace* [López-Ibáñez et al., 2016], um pacote que implementa uma série de procedimentos de configuração automática, em particular, o procedimento de corrida iterada, que vem sendo usado com sucesso para configurar automaticamente vários algoritmos do estado da arte.

No experimento realizado foram escolhidas aleatoriamente instâncias representativas de todos os conjuntos considerados neste trabalho. Para os parâmetros foram pré-selecionados alguns valores, conforme apresentados na Tabela 5, em que a coluna *Pârametros* indica qual parâmetro foi analisado e a coluna *Valores* apresenta o conjunto dos diferentes valores possíveis para cada parâmetro.

Tabela 5: Ajuste de parâmetros usando *irace*.

Parâmetros	Valores
α (%)	{5, 10, 15, 20, 25, 30, 35, 40, 45, 50}
β (%)	{10, 20, 30, 40, 50, 60, 70, 80, 90, 100}
iter_max	{50, 100, 150}

Entre as configurações recomendadas pelo *irace*, optou-se por $\alpha = 10\%$, $\beta = 80\%$ e *iter_max* = 150.

5.2. Comparação de Resultados

Dentre as instâncias consideradas nos trabalhos descritos na Seção 2, somente as propostas por Haddadi et al. [2015] puderam ser obtidas. Este conjunto de quarenta e cinco instâncias artificiais é dividido em nove grupos (A-I), com cinco instâncias cada. Conforme contextualizado na Seção 2, estas instâncias foram originalmente geradas para o Problema de Minimização de Blocos Consecutivos, um problema equivalente ao MDP. Desta forma, os resultados são apresentados para ambos os objetivos destes problemas.

Nas tabelas apresentadas, a coluna *Grupo* identifica cada um dos grupos de instâncias, $|P|$ representa o número de peças e $|S|$ indica o número de padrões. Os tempos médios de execução (expressos em segundos) são apresentados na coluna T, o valor médio das soluções obtidas em 10 execuções independentes é apresentado na coluna *Média* e a melhor solução obtida é apresentado na coluna *Melhor*. A coluna *gap* apresenta a distância percentual entre os resultados obtidos e os de referência, calculada como $100\times$((*ILS_MDP* - *Referência*) / *Referência*). Por fim, a coluna σ indica o desvio padrão entre as soluções encontradas entre as execuções independentes do método proposto.

Na Tabela 6 é apresentada a comparação entre o método proposto neste trabalho e os resultados obtidos por Haddadi et al. [2015]. Os valores apresentados na referida tabela são as médias de cada um dos grupos.

Tabela 6: Comparação de resultados quanto ao número de blocos consecutivos.

Grupo	$\|P\|$	$\|S\|$	Haddadi et al. [2015]		*ILS-MDP*			
			Média	T	Média	Mínimo	*gap* (%)	T
A	100	200	253,0	0,45	267,06	262,0	3,56	74,19
B	100	200	695,8	0,62	675,42	671,6	-3,48	451,50
C	100	200	1358,6	0,79	1311,30	1307,2	-3,78	769,34
D	100	500	552,0	3,32	607,06	601,2	8,91	1.015,27
E	100	500	1616,0	4,59	1585,68	1576,2	-2,46	4.220,57
F	100	500	3308,4	5,52	3169,16	3162,2	-4,42	8.761,34
G	100	1000	1072,4	14,03	1185,40	1178,2	9,87	4.772,54
H	100	1000	3125,4	22,20	3068,78	3067,2	-1,86	7.612,72
I	100	1000	6375,4	27,12	6124,40	6109,6	-4,17	16.631,09

O *gap* médio em relação ao método de referência é 2,17%. Ainda, o método proposto foi capaz de determinar novas melhores soluções em seis grupos de instâncias, dentre os nove apresentados. Dentre os grupos com novas melhores soluções, o grupo F apresentou a maior redução (4,42%) ao passo que o grupo H apresentou a menor redução (1,86%). É possível destacar que para os conjuntos mais esparsos (A, D e G) o método proposto não conseguiu igualar as soluções de referência, obtendo valores de *gap* que variam entre 3,56% e 9,87%.

A ILS também foi capaz de reduzir em aproximadamente 32,32% o número inicial de blocos consecutivos. O grupo G obteve a maior redução, com diminuição de 48,13%, ao passo que o grupo C apresentou a menor redução, equivalente à 26,94%, ainda assim, um valor considerável.

Embora não seja justa a comparação dos tempos de execução de métodos executados em arquiteturas computacionais diferentes, é clara a vantagem do método de referência. Este método apresentou tempos de execução muito baixos variando entre 0,45 s e 27,12 s, enquanto o método proposto apresentou variação de 74,19 s ate 16.631,09 s. Entretanto, cabe ressaltar que, devido às dimensões das instâncias (500 e 1000 colunas) e à qualidade das soluções geradas, esse valor é considerado aceitável na prática.

Em média, o algoritmo proposto convergiu para a melhor solução obtida nos 44,65% iniciais do tempo de execução, sendo que o valor máximo observado para esta estatística foi de 49,53%. Esses dados demonstram que para este conjunto de instâncias, em média a melhor solução é encontrada em menos que a metade do tempo de execução do algoritmo, evidenciando a rápida convergência relativa.

Os dados reportados indicam que o método proposto é notavelmente melhor para instâncias densas, dado que em todos os casos superou o método que representa o estado da arte. Por outro lado, o tempo de execução é muito superior quando comparado a este mesmo método, e também quando se considera instâncias esparsas. A Tabela 7 apresenta os valores obtidos para as mesmas

instâncias, porém, em relação ao número de descontinuidades.

Tabela 7: Resultados quanto ao número de descontinuidades.

Grupo	$\|P\|$	$\|S\|$	Média	Melhor	σ	T
A	100	200	167,06	162	2,95	74,19
B	100	200	575,42	571,6	2,28	451,5
C	100	200	1211,3	1297,2	2,44	769,34
D	100	500	507,06	501,2	3,73	1.015,27
E	100	500	1485,68	1476,2	5,4	4.220,57
F	100	500	3069,16	3062,2	4,35	8.761,34
G	100	1000	1085,4	1078,2	4,5	4.772,54
H	100	1000	2968,78	2967,2	7,09	7.612,72
I	100	1000	6024,4	6009,6	8,01	16.631,09

Não foi possível realizar comparações sobre o número de descontinuidades, dado que o trabalho de referência trata apenas do Problema de Minimização de Blocos Consecutivos, cuja função objetivo não é diretamente proporcional à função objetivo do MDP. Entretanto, os resultados obtidos são apresentados para futuras comparações. Adicionalmente, é possível avaliar a consistência do método, cujo desvio padrão médio é apenas 4,52. O grupo G apresentou o maior desvio padrão, com valor igual à 8,01. Entretanto, isto significa menos do que 1% do número de padrões para essas instâncias, portanto, trata-se de um valor baixo.

6. Conclusões e Trabalhos Futuros

Neste trabalho foi abordado o Problema de Minimização de Descontinuidades (MDP), um problema NP-Difícil de sequenciamento de tarefas com ampla aplicação industrial. A solução deste importante problema pode reduzir os custos operacionais da produção de bens de consumo e também auxiliar na uniformização das características físicas dos mesmos. Apesar da importância do problema, existem poucas propostas na literatura para a sua resolução. Neste trabalho, foram propostos uma nova representação em grafos para o problema, um heurística gulosa baseada em um algoritmo clássico da teoria dos grafos e a aplicação inédita da metaheurística Busca Local Iterada (ou ILS).

Para o único conjunto *benchmark* disponível na literatura, a ILS gerou novos melhores resultados para seis de seus nove grupos. Não é possível precisar o número exato de novas melhores soluções, dado que o método comparado somente reporta os resultados médios por grupo. De acordo com as análises realizadas, evidenciou-se que a ILS possui melhor performance em instâncias densas do que em instâncias esparsas. Em geral, a ILS apresentou consistência no que tange ao valor da solução em múltiplas execuções, apresentando baixo desvio padrão. Foi verificado também a rápida convergência do algoritmo proposto que no pior caso a melhor solução foi encontrada em aproximadamente metade do tempo de execução.

Os trabalhos futuros se concentrarão em gerar novas instâncias *benchmark* e também em aprimorar o algoritmo proposto, a fim de se obter melhores resultados para instâncias esparsas e a diminuição do tempo de execução do mesmo.

7. Agradecimentos

Esta pesquisa foi apoiada pelo Conselho Nacional de Desenvolvimento Científico e Tecnológico, pela Fundação de Apoio à Pesquisa do Estado de Minas Gerais e pela Universidade Federal de Ouro Preto.

Referências

Becceneri, J. C., Yanasse, H. H., e Soma, N. Y. (2004). A method for solving the minimization of the maximum number of open stacks problem within a cutting process. *Comput. Oper. Res.*, 31 (14):2315–2332. ISSN 0305-0548.

Carvalho, M. A. M. e Soma, N. Y. (2015). A breadth-first search applied to the minimization of the open stacks. *Journal of the Operational Research Society*, 66(6):936–946.

Croes, G. A. (1958). A method for solving traveling-salesman problems. *Operations research*, 6 (6):791–812.

Cuthill, E. e McKee, J. (1969). Reducing the bandwidth of sparse symmetric matrices. In *Proceedings of the 1969 24th national conference*, p. 157–172. ACM.

Dyson, R. e Gregory, A. (1974). The cutting stock problem in the flat glass industry. *Operational Research Quarterly*, p. 41–53.

Garey, M. R. e Johnson, D. S. (1979). *A Guide to the Theory of NP-Completeness*. WH Freemann, New York.

Haddadi, S., Chenche, S., Cheraitia, M., e Guessoum, F. (2015). Polynomial-time local-improvement algorithm for consecutive block minimization. *Information Processing Letters*, 115(6):612–617.

Linhares, A. e Yanasse, H. H. (2002). Connections between cutting-pattern sequencing, vlsi design, and flexible machines. *Computers & Operations Research*, 29(12):1759–1772.

López-Ibáñez, M., Dubois-Lacoste, J., Cáceres, L. P., Birattari, M., e Stützle, T. (2016). The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, 3:43–58.

Lourenço, H. R., Martin, O. C., e Stützle, T. (2003). Iterated local search. In *Handbook of metaheuristics*, p. 320–353. Springer.

Madsen, O. B. (1979). Glass cutting in a small firm. *Mathematical Programming*, 17(1):85–90.

Paiva, G. S. e Carvalho, M. A. M. (2016). Um método para planejamento da produção em sistemas de manufatura flexível. In *Anais do XLVIII Simpósio Brasileiro de Pesquisa Operacional*, p. 3717–3724.

Santos, J. V. M. e Carvalho, M. A. M. (2015). Uma heurística aplicada à produção em microeletrônica. In *Anais do XLVII Simpósio Brasileiro de Pesquisa Operacional*.

Yanasse, H. H. (1997). On a pattern sequencing problem to minimize the maximum number of open stacks. *European Journal of Operational Research*, 100(3):454–463.