LUIS HENRIQUE LEÃO DO NASCIMENTO

Orientador: Marco Antonio Moreira de Carvalho

UM ALGORITMO HEURÍSTICO APLICADO A HOMOGENEIZAÇÃO DAS CARACTERÍSTICAS FÍSICAS DE PRODUTOS

Ouro Preto

Março de 2017

Universidade Federal de Ouro Preto
Instituto de Ciências Exatas
Bacharelado em Ciência da Computação

UM ALGORITMO HEURÍSTICO APLICADO A HOMOGENEIZAÇÃO DAS CARACTERÍSTICAS FÍSICAS DE PRODUTOS

Monografia apresentada ao Curso de Bacharelado em Ciência da Computação da Universidade Federal de Ouro Preto como requisito parcial para a obtenção do grau de Bacharel em Ciência da Computação.

LUIS HENRIQUE LEÃO DO NASCIMENTO

Ouro Preto
Março de 2017

UNIVERSIDADE FEDERAL DE OURO PRETO

FOLHA DE APROVAÇÃO

Um Algoritmo Heurístico Aplicado a Homogeneização das Características Físicas de Produtos

LUIS HENRIQUE LEÃO DO NASCIMENTO

Monografia defendida e aprovada pela banca examinadora constituída por:

Dr. Marco Antonio Moreira de Carvalho – Orientador
Universidade Federal de Ouro Preto

Dr. André Luís Silva
Universidade Federal de Ouro Preto

Msc. Marcelo Luiz Silva
Universidade Federal de Ouro Preto

Ouro Preto, Março de 2017

Resumo

A evolução e a disseminação da tecnologia, permitiu que em ambientes industriais, os diferentes processos produtivos sejam confiados a métodos computacionais, os quais têm sido aplicado com eficiência, contribuindo para redução de custos, aumento da produtividade, entre outros. Os processos produtivos devem ser otimizados visando algum critério pré-estabelecido como, por exemplo, a minimização da utilização de matéria prima ou a minimização de mão de obra. Em certas indústrias existe a necessidade que os produtos fabricados possuam uma qualidade satisfatória no que tange as características físicas dos mesmos, para que seu 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 a necessidade de interrupção de produção dos mesmos. Este problema é conhecido como o *Problema de Minimização de Descontinuidades* (MDP). Neste trabalho são propostos uma representação em grafos para o problema, uma heurística baseada em um algoritmo clássico existente na teoria do grafos e a implementação de uma metaheurística composta por métodos de busca locais e de perturbação. Experimentos computacionais extensos demonstram que o método proposto é competitivo e obteve novas melhores soluções para algumas instâncias da literatura, superando estado da arte atual. Para outras instâncias são apresentados valores inéditos que poderão ser utilizados como base de comparação para trabalhos futuros.

Abstract

The development and dissemination of technology allowed that, in industrial environments, the different production processes to be entrusted to computational methods, which have been applied effectively, helping to reduce costs and increase productivity, among others. The production processes must be optimized considering predefined criteria. For example, minimizing the raw material waste or minimizing the use of manpower. In certain industries there is a need that the manufactured products have a satisfactory quality regarding the physical characteristics, so their sale price and quality are not affected. To meet this criterion the production line should be planned and optimized aiming to manufacture the products of the same lot without the need to interrupt production. This problem is known as the Minimization of Descontinuites Problem. This work proposes a graph representation for the problem, a heuristic based on a existing classic algorithm in graph theory and the implementation of a metaheuristic composed of local search methods and perturbation. Extensive computational experiments demonstrate that the proposed method is competitive and has obtained new best solutions for some of literature instances, surpassing current state of the art. For other instances, unpublished values are presented that can be used as a basis for comparison for future works.

Dedico este trabalho a minha família, namorada e amigos que me apoiaram nesta jornada.

Agradecimentos

Agradeço a todos professores do departamento, em especial ao meu orientador Marco.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Algoritmos

Capítulo 1

Introdução

O advento da Revolução Industrial na segunda metade do século XVIII, proporcionou um grande avanço tecnológico em todo o mundo, principalmente no contexto industrial onde ocorreu a substituição em larga escala de processos manuais por processos automatizados ou semi-automatizados. Esse acontecimento possibilitou a otimização da mão de obra, do tempo de trabalho e do uso de matéria prima, com o intuito de aumentar a produtividade e consequentemente a lucratividade a partir do comércio dos bens produzidos.

Nesse contexto, atualmente, a inteligência computacional e seus métodos têm sido utilizados nas indústrias, basicamente em todas as fases da produção, desde a obtenção da matéria-prima até a entrega do produto acabado ao consumidor final. Etapas intermediárias incluem o planejamento da produção, o processamento de matéria prima, a estocagem e a manipulação de produtos.

Em algumas indústrias há a necessidade de transformar unidades maiores de matéria prima em produtos menores com diferentes formas e tamanhos. Este processo pode ser caracterizado como um Problema de Corte, 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 plano (ou *padrão*) de corte pode ser caracterizado como a disposição das peças dentro de uma chapa de matéria prima, semelhante a um gabarito. Cada padrão é unicamente reconhecido por possuir uma quantidade finita de peças e para cada uma há uma posição associada. A disposição de cada peça dentro destes padrões, interfere diretamente nos diferentes objetivos dos problemas de corte.

Para atender a demanda de toda a linha de produção, ou seja, das peças que precisam ser produzidas, é preciso realizar o corte dos diferentes padrões várias vezes. Desta forma, caracteriza-se o Problema de Corte de Estoque, amplamente estudado na literatura e que

geralmente tem como objetivo minimizar a quantidade de padrões a serem cortados.

Um *estágio* pode ser definido como uma subdivisão de um processo em uma sequência de sub-processos. No meio industrial um estágio da produção pode ser caracterizado como uma unidade de tempo em que determinado padrão de corte é processado. A ordem em que os padrões são processados, pode interferir diretamente na produtividade, ocasionando custos desnecessários. É preciso então planejar a sequência em que os padrões serão cortados atendendo algum critério de qualidade, como por exemplo:

- produzir peças de um mesmo lote que possuam características físicas semelhantes;
- minimizar o estoque intermediário;
- minimizar o número de interrupções na produção para a realização de ajustes na máquina de produção.

Cada diferente critério estabelece um problema de otimização diferente, definindo assim os Problemas de Sequenciamento de Padrões.

No contexto do tema abordado neste trabalho, é interessante minimizar o número de vezes em que o processamento de uma peça qualquer é interrompido, ou seja, o número de *descontinuidades* em sua produção. Uma descontinuidade surge quando uma determinada peça está sendo processada a partir de um padrão em um determinado estágio, e em nos 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 outro tipo de peça. Esse fato pode ocasionar uma pausa na linha de produção para a troca da cor da tinta utilizada pela máquina ou a troca de lâminas de corte, por exemplo. 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.

Em nichos específicos de mercado, as características físicas dos produtos são um fator determinante no seu valor de revenda e aceitação pelos consumidores. Diferentes métricas podem ser utilizadas para inspecionar a qualidade de um produto, porém, uma que se destaca particularmente nas indústrias de processamento de madeira, vidro e cerâmica é a homogeneidade das características físicas, tais como textura, cor, padronagem e tonalidade.

Nestas indústrias, diferentes peças são utilizadas para a montagem de um único bem de consumo, como móveis, ou usadas em conjunto, principalmente na construção civil, como placas de revestimento e vidros. A introdução de variações acentuadas nestas características nos produtos de um mesmo lote pode causar uma diminuição considerável no valor de revenda dos mesmos. Ou seja, produtos pertencentes ao mesmo lote devem ser o mais parecidos possíveis, para que o seu preço de venda não seja afetado pelas variações acentuadas nas características físicas. Para atender o critério anteriormente citado é preciso sequenciar os padrões

visando diminuir o número de descontinuidades presentes, caracterizando assim o *Problema de Minimização de Descontinuidades(Minimization of Descontinuites Problem*,MDP).

Neste trabalho, apresenta-se um estudo aprofundado sobre o MDP, realizando uma pesquisa de geração de embasamento teórico e revisão bibliográfica. Além disto é apresentada a implementação de uma metaheurística inédita para o MDP, utilizando dois métodos clássicos e um novo método para a composição da mesma. Por fim, foi realizada uma gama de experimentos com vários conjuntos diferentes de instâncias com o intuito de verificar a qualidade e a consistência do algoritmo proposto.

1.1 Motivação

Há duas motivações principais para esse trabalho. Primeiramente, pela sua relevância teórica, em que o trabalho apresentado pertence a classe NP-Difícil. Segundo, por se tratar de um problema de aplicação prática no contexto industrial, tornando-se essencial para o planejamento e execução do processo produtivo de diversas indústrias, tais como as relacionadas a produtos de madeira, vidro, papel e cimentícios. Ainda, o tema é relevante nas áreas de movelaria e construção civil, com aplicações práticas diretas na engenharia e indústria. A criação de novos modelos para solução do problema em questão pode contribuir para melhoria da eficiência da indústria nacional.

1.2 Objetivos

De maneira geral, esse trabalho consiste em desenvolver um método de inteligência computacional para a resolução do MDP. A partir do objetivo principal, temos os seguintes objetivos específicos:

- realizar pesquisa para geração de embasamento teórico e revisão bibliográfica sobre MDP;
- elaborar uma heurística consistente que possa ser utilizada no problema e que permita a obtenção rápida de soluções próximas da solução ótima sem que se perca a vantagem da busca sistemática;
- pesquisar e implementar uma metaheurística para melhoria fina das soluções;
- avaliar o método implementado considerando dados reais e também com problemas teste publicamente disponíveis, realizando uma análise crítica considerando outros métodos da literatura.

Outros produtos deste projeto de pesquisa serão trabalhos publicados em eventos nacionais, os quais contribuem para a promoção da Universidade Federal de Ouro Preto e também do tema tratado.

1.3 Organização do Trabalho

Nas próximas seções o trabalho é detalhado. O Capítulo 2 apresenta a revisão da literatura de 1974 até 2015. A base conceitual do MDP é apresentada no Capítulo 3. No Capítulo 4 é apresentado uma implementação do método proposto. Os experimentos utilizados são apresentados no Capítulo 5. As conclusões sobre este trabalho são apresentadas no Capítulo 6.

Capítulo 2

Revisão da Literatura

Nesse capítulo, os principais trabalhos sobre o Problema de Minimização de Descontinuidades são descritos brevemente. Os trabalhos são divididos em:

- práticos: propõem heurísticas e métodos exatos para a solução do problema;
- teóricos: apresentam a complexidade do problema e o estudam o relacionamento entre os problemas de sequenciamento de padrões;
- relacionados: apresentam certa similaridade com o problema tratado neste trabalho mas são utilizados para a resolução de outros problemas.

2.1 Trabalhos Práticos

O primeiro trabalho a abordar o MDP foi o realizado por **?**, integrado ao problema de corte de estoque. Inicialmente, foi utilizado o método simplex revisado para determinar a composição dos padrões de corte para atender toda a demanda por peças com o menor desperdício de matéria-prima possível. A fase de sequenciamento de padrões foi modelada como o *Problema do Caixeiro Viajante(Travelling Salesman Problem-TSP)*, em que os padrões são os vértices do grafo, havendo arestas entre todos os pares de vértices. As distâncias foram 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 TSP.

Como alternativa, os autores propuseram um método que resolve o problema de corte de estoque utilizando *branch-and-bound* e cria o sequenciamento dos padrões na mesma ordem que são gerados. Para isto foi utilizada a seguinte metodologia: em cada interação o *branch-and-bound* resolve o problema de corte de estoque considerando todos os padrões restantes a fim de determinar o menor desperdício, sendo que determinadas peças têm o valor de sua área aumentado por um parâmetro para de serem escolhidas primeiro. Esse parâmetro é calculado

para todas as peças que já foram processadas em algum padrão anterior e que ainda precisam ser cortadas para produzir a quantidade mínima para atender a demanda da indústria.

Os resultados obtidos mostraram que a heurística proposta comparada com cálculos manuais apresentou pouca melhoria em uma indústria de produção de vidro.

?, primeiramente resolve o Problema de Corte de Estoque para calcular o mínimo de desperdício possível. Para a resolução do MDP foi proposta uma abordagem utilizada criando uma matriz C binária, quadrada e simétrica para representar quais padrões possuem pelo menos um par de peças em comum. Cada entrada c_{ij} da matriz é preenchida da seguinte maneira:

$$c_{ij} = \begin{cases} 1, & \text{se o padrão } i \text{ possui uma peça em comum com o padrão } j \\ 0, & \text{caso contrário} \end{cases}$$

Foi aplicado então a heurística de *Cuthill-McKee* (**?**) para o problema de Minimização de Largura de Banda em Matrizes Simétricas. Ao comparar o método proposto com a solução usada na empresa, houve uma significativa melhora no desperdício. Não houve relatos de melhoria das descontinuidades na empresa.

?, foi o primeiro artigo a focar somente no sequenciamento de padrões, visando diminuir o espalhamento de ordens de compras de um conjunto de clientes. O *Problema de Minimização de Espalhamento de Ordens*(*Minimization of Order Spread Problem* – MORP) tem como objetivo atender a ordens de compra dos clientes o mais rápido possível, de maneira que uma vez iniciada a produção das peças de uma determinada ordem, todas as outras peças relacionadas sejam produzidas imediatamente, visando minimizar a estocagem intermediária dos produtos.

Foi utilizada a formulação do TSP proposta por (**?**) para o MDP, para resolver o problema de estoque intermediário de uma pequena empresa de vidro em um intervalo de 14 dias, para isso foi proposto uma heurística de três estágios.

Primeiramente foi utilizado o método proposto por **?**, para resolver o Problema de Corte em Estoque sem levar em consideração o espalhamento de ordens.

No seguinte estágio foi proposta uma matriz C de dimensão $n \times n$, em que n é a quantidade de padrões existentes, da seguinte forma:

- inicialmente foi adicionado um padrão fictício, com nenhuma peça de corte, para que seja possível o aspecto circular doTSP;
- a diagonal principal foi preenchida com um valor alto (10.000);
- para todas as ligações entre o padrão fictício a outro padrão qualquer foi atribuído o valor 1.000;
- para padrões que possuem apenas peças singulares, ou seja, peças que não estão contidas em outros padrões foi atribuído o valor 110;

- para as posições restantes, os padrões que possuem n peças em comum o valor de $c_{i,j}$ é definido por $100 - 10 \times n$.

Os pesos foram atribuídos dessa maneira, para que ao solucionar o TSP os padrões que possuem mais peças em comuns sejam cortados o mais próximo possível. Os padrões compostos apenas por peças singulares serão os últimos a serem processados, pois não existe descontinuidades em nenhuma de suas peças.

Para solução do TSP foi utilizada a heurística *3-opt*. Este é um método de busca local em que três arestas presentes na solução $(k_1, k_2), (j_1, j_2)$ e (i_1, i_2) são trocadas por outras três novas arestas presentes no grafo e não presentes na solução, a fim de se obter uma melhoria na solução atual. Se existir alguma nova configuração melhor que a anterior, mantem-se essa solução. Caso contrário, escolhe-se novamente outras três arestas para análise.

Comparando os resultados obtidos a partir do primeiro estágio com a solução do TSP, surpreendentemente, os melhores resultados obtidos foram relativos à minimização das descontinuidades, na qual foi reportada uma melhoria média de 30%. Já para o estoque intermediário ocorreu uma diminuição média de 18% entre os resultados.

2.2 Trabalhos Teóricos

Com o nome de *Consecutive Blocks Minimization*, **?** mostraram que o MDP pertence a classe NP-Difícil, ou seja, não existe algoritmo conhecido que resolva este problema em tempo determinístico polinomial. **?** mostra que este problema aparece em várias aplicações práticas como:

- escalonamento de Tarefas;
- recuperação de Informação;
- otimização de Linhas Férreas;
- bioinformática;
- arqueologia;
- resolução de Sistemas Lineares;

?, apresentaram um estudo detalhado sobre o relacionamento dos diferentes problemas de sequenciamento de padrões, mostrando que, apesar de possuírem mesma base conceitual, não são equivalentes entre si.

2.3 Trabalhos Relacionados

Recentemente, **?**, utilizando o *Problema de Minimização de Blocos Consecutivos(Consecutive Blocks Minimization-CBM)* propôs dois métodos heurísticos para a redução da dimensão de sistemas lineares. Para a resolução do problema, primeiramente foi proposta uma matriz binária na qual as linhas correspondem ao número de equações presentes no sistema, e as colunas correspondem ao número de incógnitas do problema. É atribuído um valor não nulo para todas as posições em que uma incógnita qualquer do sistema está multiplicada por um coeficiente diferente de 0.

Através da matriz resultante é utilizado método *2-swap* (método 1), que consiste na troca de duas colunas quaisquer para tentar melhorar a solução e um método de *shift* (método 2), que consiste em colocar uma coluna em uma determinada posição e deslocar todas as outras uma posição para a direita. Foi proposto uma heurística que utiliza os dois métodos simultaneamente. Primeiramente o método 2 é aplicado até que não ocorra mais melhoria na solução, posteriormente, o método 1 é aplicado de maneira semelhante.

Para a realização dos testes foram utilizadas um conjunto de instâncias geradas pelo próprio autor e outras instâncias reais de um problema equivalente (relacionado estações de trem). Estas instâncias foram divididas em categorias conforme suas dimensões e densidades. As instâncias foram fornecidas gentilmente pelo autor do artigo, para uma posterior comparação com o método proposto neste trabalho.

Comparando os resultados obtidos, o método 2 foi superior ao método 1 em todas as instâncias mostradas pelo autor quando aplicados individualmente, tanto no quesito do valor da solução quanto no tempo de execução dos algoritmos. Os resultados obtidos a partir da aplicação da heurística para todas as instâncias mostraram uma melhora em relação ao valor inicial da solução, em relação a quantidade de blocos consecutivos. Para as instâncias reais não foi possível comparar os resultados obtidos com os resultados originais, pois estes dados não foram disponibilizados.

Capítulo 3

Fundamentação Teórica

Este capítulo apresenta em detalhes o Problema de Minimização de Descontinuidades e suas propriedades, assim como a fundamentação dos algoritmos de busca local presentes na literatura que foram utilizados neste trabalho e a metaheurística Busca Local Iterada. Detalhes específicos da aplicação destes métodos ao problema abordado são apresentados no Capítulo 5.

3.1 O Problema de Minimização de Descontinuidades

O Problema de Minimização de Descontinuidades(ou MDP) 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 finito de peças P para atender uma determinada demanda. 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.

Durante o processo produtivo, uma descontinuidade ocorre na produção de uma peça quando um padrão que contém a referida peça é processado em um determinado estágio e no estágio seguinte nenhuma cópia da peça é produzida, sendo que outras cópias serão produzidas em estágios posteriores, caracterizando assim uma interrupção na produção daquela peça. Ao fabricar peças diferentes, a mesma matéria prima é utilizada e, posteriormente, ao retomar a produção deste tipo de peça específica, as características da matéria prima podem ser diferentes, originando variações de características físicas. Sendo assim, descontinuidades na produção de peças podem ocasionar variações de textura, cor, padronagem e tonalidade em produtos de um mesmo lote. É preciso então um planejamento da linha de produção que equilibre e, se necessário, priorize a minimização do número de descontinuidades, independente da duração das mesmas.

Uma instância MDP é representada por uma matriz M binária que relaciona os padrões de corte e as peças. Cada elemento m_{ij} $(i \in P, j \in S)$ da matriz M é preenchida da seguinte

Tabela 3.1: 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

forma:

$$m_{ij} = \begin{cases} 1, & \text{se o padrão } p_j \text{ possui a peça } i \\ 0, & \text{caso contrário} \end{cases}$$

A Tabela 3.1 apresenta um exemplo de instância MDP. 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.

No referido exemplo, o padrão p_1 é composto pelas peças 1, 2, e o padrão p_2 é composto pelas peças 1, 5 e assim por diante.

Uma solução para o MDP é dada por uma permutação π das colunas da matriz M, dando origem à uma Matriz Q^π, que consiste nos mesmos elementos da Matriz M, porém, com as colunas permutadas. Essa representação indica qual padrão de corte será processado em cada estágio da produção. A Tabela 3.2 apresenta dois possíveis sequenciamentos dos padrões com base na instância definida na Tabela 3.1. Na referida Tabela, para a primeira solução existe descontinuidades nas produção das peças 1 e 3. Já para a segunda solução, há descontinuidades na produção das peças 1, 2, 4, 5 e 6.

No primeiro sequenciamento, definido na Tabela 3.2(a), temos $\pi = [p_5, p_2, p_4, p_6, p_3, p_1]$, o que indica que no primeiro estágio serão processadas as cópias do padrão p_5, no segundo estágio serão processadas as cópias do padrão p_2 e assim sucessivamente até que no último estágio sejam processadas as cópias do padrão p_1. O segundo sequenciamento, na Tabela 3.2(b), apresenta uma solução diferente, com um sequenciamento de padrões diferente, $\pi = [p_1, p_6, p_5, p_4, p_3, p_2]$. Os valores em negrito presentes na Tabela 3.2 indicam em quais estágios da produção ocorreu descontinuidade de uma determinada peça.

Uma maneira aproximada de determinar o número de descontinuidades na matriz Q^π é determinar o número de inversões de 0 para 1 em cada linha da mesma matriz. Embora o número de inversões seja maior do que o número de descontinuidades, ambos são proporcionais. Tendo definido a matriz Q^π, a quantidade de descontinuidades presentes é calculada pela Função 3.1.

Tabela 3.2: Dois possiveis sequenciamentos.

	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)

$$Z^{\pi}_{MDP}(Q^{\pi}) = \sum_{j=1}^{J} \sum_{i=1}^{I} q^{\pi}_{ij}(1 - q^{\pi}_{ij-1}) \tag{3.1}$$

No sequenciamento ilustrado na Tabela 3.2(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 (Tabela 3.2(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, calculada de acordo com a Função 3.1. O primeiro sequenciamento apresenta 7 inversões e o segundo um total de 11 inversões.

A partir da análise da Função 3.1 é possível definir a função objetivo do Problema de Minimização de Descontinuidades conforme mostrado na Função 3.2, que visa determinar uma permutação π das colunas da matriz M (dentre o conjunto de todas as permutações possíveis Π) que minimize o número máximo de descontinuidades.

$$\min_{\pi \in \Pi} \; Z^{\pi}_{MDP}(M) \tag{3.2}$$

3.1.1 Pré-Processamento por Dominância

O pré-processamento por dominância é uma técnica que consiste na remoção de padrões que possuem em sua composição todas as peças existentes em algum outro padrão do problema, a fim de eliminar redundâncias. Um padrão p_i que possui em sua composição um subconjunto de peças que engloba todas as peças de um padrão p_j é denominado como padrão *dominante*, ao passo que o padrão p_i é dito *dominado*.

Os padrões dominados podem ser removidos da entrada, diminuindo assim o tamanho do problema tratado. A partir da solução final proposta por algum método, os padrões dominados são sequenciados imediatamente após seus respectivos padrões dominantes em π, sem perda de otimalidade. A Tabela 3.3 apresenta um exemplo de instância antes (a) e após (b) da aplicação do pré-processamento por dominância entre padrões.

Tabela 3.3: Exemplo de nova instância após a aplicação do pré-processamento por dominância.

	p_1	p_2	p_3	p_4	p_5	p_6
1	1	1	0	**1**	1	0
2	0	1	0	0	0	1
3	1	1	**1**	**1**	1	1
4	1	0	0	0	1	1
5	0	0	0	**1**	1	1
6	1	0	**1**	0	0	0
7	1	1	0	0	1	1

(a)

	p_1p_3	p_2	p_5p_4	p_6
1	1	1	1	0
2	0	1	0	1
3	1	1	1	1
4	1	0	1	1
5	0	0	1	1
6	1	0	0	1
7	1	1	1	1

(b)

Analisando a Tabela 3.3(a) verifica-se que a composição do padrão p_3, formado pelas peças 3 e 6 (ressaltadas em negrito) é um subconjunto das peças que compõem o padrão p_1, formado pelas peças 1, 3, 4, 6 e 7. De maneira análoga, o padrão p_4, formado pelas peças 1, 3 e 5 (também ressaltadas em negrito) consiste em um subconjunto das peças que compõem o padrão p_5, formado pelas peças 1, 3, 4, 5 e 7. Dessa forma, o pré-processamento por dominância entre padrões elimina os padrões p_3 e p_4 da entrada, resultando em uma instância reduzida, conforme a Tabela 3.3(b). Em uma possível solução, os padrões p_1 e p_3 devem obrigatoriamente serem sequenciados consecutivamente, assim como os padrões p_5 e p_4, sem causar qualquer prejuízo a solução do problema.

3.2 Geração da Solução Inicial

Foram considerados dois diferentes métodos para geração de uma solução inicial. O primeiro, proposto por **?** consiste na aplicação de aleatoriedade para a geração da mesma. O segundo método, entretanto, é um algoritmo clássico de busca em grafos utilizado de maneira inédita ao MDP.

3.2.1 Busca em Largura

A Busca em Largura (*Breadth-First Search - BFS*) (**?**) é um algoritmo de busca em grafos que, dado um vértice inicial, explora toda a vizinhança deste vértice. Durante a exploração, os vértices são visitados e inseridos em uma estrutura de fila. Terminada essa exploração, o primeiro vértice da fila é removido da mesma e é selecionado como novo vértice inicial e repete-se esse processo até que todo o grafo seja explorado, não visitando um vértice mais de uma vez. Caso exista mais de um componente, a busca em largura examina todo o componente, e logo após, um vértice presente em outro componente é selecionado para reiniciar a busca. Dessa maneira a busca em largura retorna uma lista com a ordem em que todos os vértices presentes no grafo foram explorados.

A Figura 3.1 apresenta um exemplo da BFS aplicada a uma grafo de exemplo mostrado na Figura 3.1a. O vértice 1 foi escolhido como inicial escolhido (em vermelho), como visto na Figura 3.1b. A partir desse vértice todos os vizinhos são visitados (em verde) vide Figura 3.1c e inseridos na fila f na ordem f=[2, 3, 4]. Os vértices 2 e 3 não possuem vizinhos que não tenham sido explorados ainda, portanto, são removidos de f e o próximo vértice a ter sua vizinhança explorada é o 4. O único vértice vizinho que não foi visitado é o vértice 5 (em azul) como mostrado na Figura 3.1d, portanto, este é inserido ao final de f. Todos os vértices de um mesmo componente foram visitados, portanto a f está vazia e a busca recomeça pelo vértice 6 (em amarelo), vide Figura 3.1e. Como não existem mais vértices não visitados, o algoritmo termina e retorna a sequência [1, 2, 3, 4, 5, 6] em que os vértices foram percorridos.

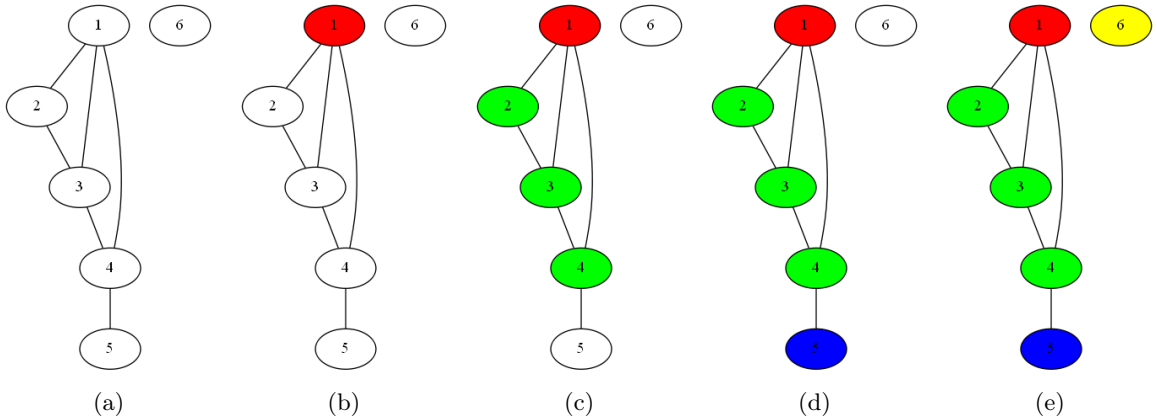


Figura 3.1: Aplicação da BFS em um grafo.

3.3 Busca Local Iterada

Para percorrer um maior espaço do conjunto de soluções **?** propuseram a metaheurística Busca Local Iterada (*Iterated Local Search– ILS*), que emprega um conjunto de busca locais para gerar novas soluções correntes. Também são aplicados modificações sutis das soluções, mudando elementos da mesma, denominadas *perturbações*, com o objetivo de mudar o foco da busca local para outra região do espaço de busca e levar à soluções que gerem melhorias na solução global.

O Algoritmo 1 apresenta a Busca Local Iterada, onde inicialmente é determinada uma solução inicial e logo após é aplicado um método de busca local. A partir dessa nova solução é realizada a perturbação na solução, aplica-se a busca local e é verificado se esta solução atende ao critério de aceitação. Caso o critério seja aceito, esta se torna a nova solução corrente. Caso contrário esta nova solução é rejeitada. Este procedimento é executado até que o critério de parada seja atendido, que pode ser número máximo de iterações, número máximo de iterações sem melhora ou tempo máximo de execução, por exemplo.

Algoritmo 1: Pseudocódigo da Busca Local Iterada

```
1  s_0 = GeraSolucaoInicial;
2  s = BuscaLocal(s_0);
3  enquanto critério de parada não atendido faça
4  |   s' = Perturbe(s);
5  |   s'' = BuscaLocal(s');
6  |   se s'' atender critério de aceitação então
7  |   |   s ← s'';
8  |   fim
9  fim
10 retorna s;
```

3.4 Métodos de Busca Local e Perturbação

Métodos de busca local podem ser definidos como um processo iterativo de melhoria de uma solução de um problema de otimização combinatória. Este processo se inicia a partir de uma solução inicial s entre todas as soluções S possíveis para o problema, e a partir desta são realizadas operações denominadas *movimentos* que geram novas soluções s' denominadas *vizinhas* da solução anterior.

Os diferentes movimentos definem diferentes *vizinhanças* que representam diferentes regiões do espaço de busca. Com base nesta estrutura de vizinhança é selecionada a melhor solução vizinha da solução corrente, para se tornar a solução atual. Este processo é repetido iterativamente até que se encontre a melhor solução possível, ou seja, até que seja atingido um *ótimo local*. A melhor solução entre todas as vizinhanças é caracterizada como *ótimo global*.

A operação de perturbação é utilizada para modificar a estrutura da solução com o intuito de ser mudar de vizinhança como uma maneira de deslocar a exploração para outra região do espaço de busca. Estes procedimentos tentam manter uma parte significativa das características da solução atual, com a adição de um fator de variação para que o espaço de busca seja melhor examinado

A operação de perturbação é feita da seguinte forma: dada uma solução, modifica-se sua estrutura para que a mesma se pareça menos com sua vizinhança atual e mantenha alguns aspectos de otimalidade, ou seja, parte da solução é modificada resultando em uma nova solução que pertence a uma nova vizinhança e sirva de solução candidata para o algoritmo de busca local.

A seguir, são apresentados três métodos para busca local e perturbação, utilizados neste trabalho. Dois deles são métodos clássicos encontrados na literatura e outro, mais recente, é aplicado pela primeira vez no contexto do MDP.

3.4.1 Método *k-swap*

O método *k-swap* é um método que consiste na realização de trocas de posição entre k elementos de uma solução s, gerando uma nova solução s' que difere exatamente de k elementos da solução anterior. Por exemplo, considere um problema permutacional presente na Figura 3.2 composto por 5 elementos, e a configuração inicial $[1, 2, 3, 4, 5]$ e o valor de $k = 2$, ou seja, a cada instante dois elementos serão trocados de lugar. A aplicação deste método produz uma uma vizinhança de 10 elementos, combinados 2 a 2.

A Figura 3.3 apresenta duas possíveis configurações com base na aplicação do 2-swap ao exemplo da Figura 3.2. Para a primeira configuração foram trocados o primeiro e o terceiro elemento, resultando na configuração $[3, 2, 1, 4, 5]$. Para a segunda configuração foram trocados o segundo e quinto elemento resultando na configuração $[1, 5, 3, 4, 2]$.

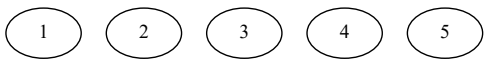


Figura 3.2: Configuração Inicial.

Esta estratégia é amplamente utilizada como método de busca local, principalmente em problemas cujas soluções são representadas por meio de permutações, assim como o MDP.

Neste trabalho, o método de *k-swap* foi utilizado como método de perturbação aplicando movimentos de troca aleatórios.

(a) configuração 1. (b) configuração 2.

Figura 3.3: Aplicação de *2-swap*.

3.4.2 Busca Local de *Agrupamento de 1-blocks*

Busca local proposta por **?** que tem como objetivo diminuir o número de *blocos consecutivos* da matriz Q^{π}. O principio é percorrer todas as linhas da matriz Q^{π} de maneira aleatória procurando por dois ou mais *blocos consecutivos*. Ao encontrá-los, esses blocos são tomados dois a dois e tentam-se agrupá-los da seguinte maneira:

- Movimentam-se as colunas referentes ao primeiro bloco uma a uma, para antes ou depois do segundo bloco, o que resultar em um 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 da busca local citada.

Algoritmo 2: Pseudocódigo da Busca Local de agrupamento de 1-blocks

```
1  Entrada: Matriz Q^π.
2  enquanto todas as linhas não forem analisadas faça
3  |  Selecione uma linha aleatoriamente l de Q^π que não foi analisada;
4  |  Examine a linha l até encontrar um primeiro bloco consecutivo j;
5  |  enquanto existir um próximo bloco consecutivo k faça
6  |  |  Para cada coluna de j decida com base com o número de descontinuidades se
   |  |  esta deve ser inserida antes ou depois de k ou ser mantida em j;
7  |  |  j ← k;
8  |  fim
9  |  retorna Q^π;
10 fim
```

Para exemplificar o funcionamento deste algoritmo, tome como exemplo a instância do MDP da Tabela 3.1 e a solução da Tabela 3.2(b), com sequenciamento de padrões $\pi = [p_1, p_6, p_5, p_4, p_3, p_2]$, cujo o número de descontinuidades é 5. Inicialmente seleciona-se aleatoriamente a segunda linha para ser analisada e encontra-se dois blocos consecutivos, o primeiro composto pela coluna 1 e o segundo pela coluna 3. Avalia-se então as possíveis realocações da coluna 1, gerando as seguintes configurações $\pi = [p_6, p_5, p_4, p_1, p_3, p_2]$ e

$\pi = [p_6, p_5, p_4, p_3, p_1, p_2]$, resultando em 4 e 3 descontinuidades. Como há um movimento de melhora este é executado e a solução parcial é atualizada para $\pi = [p_6, p_5, p_4, p_3, p_1, p_2]$. Logo após, o método segue examinando as linhas da matriz.

3.4.3 2-*Opt*

Este método foi proposto por **?** originalmente para resolução do TSP. A idéia principal de funcionamento é eliminar duas arestas presentes na solução e inserir duas arestas não presentes na mesma de forma cruzada, ou seja, seleciona-se duas arestas (k_1, k_2) e (j_1, j_2) presentes na solução e adiciona-se as arestas (k_1, j_2) e (k_2, j_1). Se esta configuração for melhor que a anterior ela é mantida, caso ao contrário, são escolhidas outras duas arestas para a análise.

A Figura 3.4 apresenta um exemplo de aplicação do método *2-opt* em que são selecionadas as arestas $(2, 6)$ e $(7, 3)$ (em vermelho), presentes na solução e são trocadas pelas as arestas $(2, 3)$ e $(6, 7)$ (em azul), caso esta nova sequência seja melhor, esta é mantida.

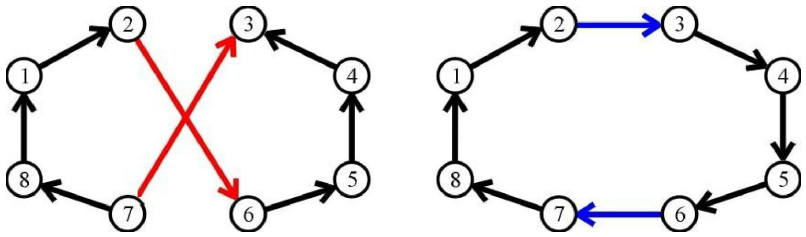


Figura 3.4: Exemplo 2-opt.

Esta estratégia é amplamente utilizada como método de busca local, principalmente em problemas cujas soluções são representadas por meio de permutações, assim como o MDP.

Neste trabalho, o método de *2-Opt* foi utilizado como método de busca local aplicando movimentos de troca aleatório e aceitando somente melhorias na solução.

Capítulo 4

Desenvolvimento

Neste capítulo é descrita com detalhes a heurística proposta e sua representação computacional, além de exemplos a fim de facilitar o entendimento dos métodos utilizados. A heurística tem como idéia a representação do MDP na forma de um grafo e então a utilização de um algoritmo básico de percurso no mesmo, a fim de diminuir o número de descontinuidades.

4.1 Representação Computacional

As informações de entrada para o MDP são dados referentes à composição de cada padrão de corte e de cada peça. Essas informações são representadas em uma matriz binária semelhante a matriz descrita na Seção 3.

Para a representação do problema, a partir de uma instância é construído um grafo ponderado e não direcionado como descrito em (**?**), em que os vértices representam as peças, havendo ligação entre dois vértices quaisquer se as peças estão presentes em um mesmo padrão. A cada ocorrência desta característica, o peso da aresta correspondente é aumentado em uma unidade.

O grafo é construído da maneira citada anteriormente visando coletar a partir das peças, informações sobre os padrões. As peças que aparecem com frequência juntas em diferentes padrões, terão adjacência no grafo com um peso elevado. Portanto, os padrões que contém determinadas peças devem ser processados em sequência.

A Tabela 4.1 apresenta uma instância MDP de exemplo e a Figura 4.1 apresenta o grafo correspondente.

Inicialmente, o grafo 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 referido grafo final.

Tabela 4.1: Instância MDP.

	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

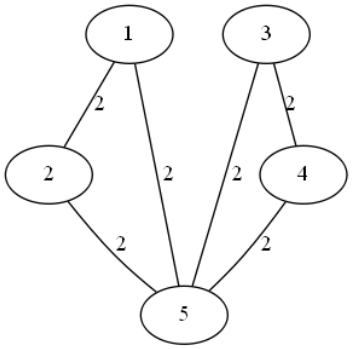


Figura 4.1: Grafo correspondente a instância anterior.

4.2 Busca em Largura aplicada ao MDP

A heurística aplicada ao MDP utiliza a BFS com algumas modificações. Primeiramente o vértice inicial é 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. Repete-se esse passos até que todos os vértices sejam visitados, retornando uma lista ϕ com a ordem de exploração dos vértices.

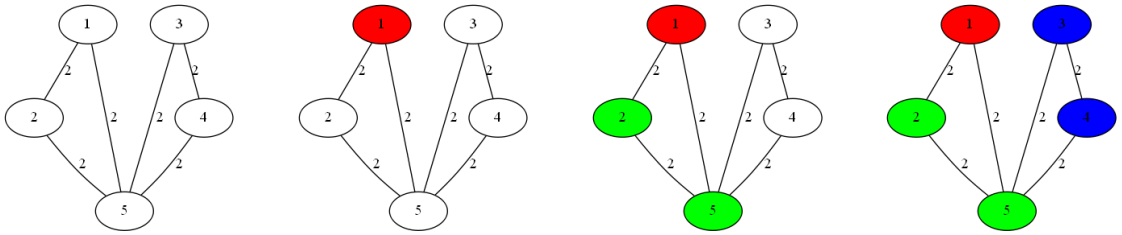
A BFS foi escolhida por ser um algoritmo simples para se percorrer um grafo de acordo com as adjacências de cada vértice (ou seja, aproveitando a vantagem da modelagem em grafos específica para este problema) e pela sua complexidade computacional. Como cada vértice entra na fila apenas uma vez e que lista de adjacências de cada vértice é examinada apenas uma vez a complexidade será $O(n + m)$, em que n representa o número de vértices e m representa o número de arestas.

O Algoritmo 3 apresenta o pseudocódigo para o sequenciamento das peças. A entrada consiste do grafo G obtido a partir da instância de entrada (linha 1). A função $f(G)$ retorna o vértice que possui o menor grau entre todos presentes no grafo (linha 2), a cada instante é verificado todos os vizinhos não explorados e adicionados na lista Q descendentemente pelo grau do mesmo e o primeiro vértice é adicionado na lista ϕ de peças (laço das linhas 6-18).

Algoritmo 3: Pseudocódigo da BFS modificada.

```
1  Entrada: Grafo de peças G.
2  v ← f(G);
3  Marque v como visitado;
4  φ ← ∅;
5  Fila Q ← v;
6  enquanto Q ≠ ∅ faça
7      T ← ∅;
8      para todo vértice w vizinho de v hacer
9          se w é marcado como não explorado então
10             Insira w ao final de T;
11             Marque w como exlorado;
12         fim
13     fin
14     Ordena T descendentemente pelo grau do vértice;
15     Insira todos os elementos de T no final de Q;
16     φ ← v;
17     Remova v de Q;
18 fim
```

A Figura 4.2 apresenta a BFS modificada aplicada ao grafo da Figura 4.1. Inicialmente, nenhum vértice foi explorado, portanto a lista ϕ é vazia (Figura 4.2a). 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 4.2b. 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 4.2c. 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 4.2d. Ao final, temos ϕ = [1,2,5,3,4].



(a) ϕ = [] (b) ϕ = [1] (c) ϕ = [1,2,5] (d) ϕ = [1,2,5,3,4]

Figura 4.2: Aplicação da BFS modificada no grafo de exemplo.

4.3 Sequenciamento de Padrões

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 (**?**) 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 unicamente a peça corrente ou qualquer subconjunto da mesma com as peças analisadas anteriormente. Todos os padrões que atendem o critério descrito anteriormente são 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.

O Algoritmo 4 apresenta o pseudocódigo para o sequenciamento dos padrões. A entrada consiste na lista de peças ϕ obtida a partir do sequenciamento das peças obtido pela BFS modificada (linha 1). A função c(pi) retorna quais peças compõem o padrão pi (linha 5), e, a cada instante, o conjunto A de peças, representa as peças disponíveis a cada iteração do algoritmo (laço das linhas 3-8).

Algoritmo 4: Pseudocódigo do sequenciamento de padrões

```
1 Entrada: Lista de peças φ.
2 A ← ∅;
3 para cada peça i ∈ φ hacer
4 |   A ← A ∪ i;
5 |   se c(p_i) ⊆ A então
6 |   |   Insira p_i ao final de π;
7 |   fim
8 fin
```

A Tabela 4.2 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, cada linha indica quais peças já foram analisadas. Primeiramente é verificado que nenhum padrão possuiu somente a peça {1}. Na segunda iteração, novamente nenhum padrão possui somente a peça {1}, {2} ou {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 e 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 ordem em que os padrões serão processados.

A tabela 4.3 apresenta a matriz resultante a partir da $\pi = [p_1, p_2, p_3, p_6, p_4, p_5]$, havendo na mesma duas descontinuidades: na peça 1 e na peça 2.

Tabela 4.2: 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

Tabela 4.3: Instância MDP depois do sequenciamento.

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

4.4 Busca Local Iterada

Após a geração da solução incial obtida através da BFS-MDP é aplicada a ILS descrita na Seção 3.3. Para compor este método foi utilizada a busca local por agrupamento de *1-blocks* descrita na seção 3.4.2 e o método *2-Opt* descrito na seção 3.4.3 como mecanismos de busca local e, como mecanismo de perturbação o método *k-swap* com $k = 2$. A partir destes métodos foi obtida a ILS-MDP em que o pseudo-código é apresentado no Algoritmo 5.

Para realizar o processo de perturbação, inicialmente foi criado um vetor com todas as possíveis combinações de padrões tomados dois a dois: $(1, 2)$, $(1, 3)$, $(1, 4)$, ...,$(n - 1, n)$. Este vetor é embaralhado e é selecionada então uma porcentagem α desses conjuntos para que os respectivos padrões sejam trocados na solução corrente, não importando o valor da solução.

Para o método *2-Opt*, o processo é semelhante ao descrito anteriormente. Depois de criado o vetor com todas as combinações este é embaralhado e então é selecionada uma porcentagem β desses conjuntos para que os respectivos padrões sejam trocados. Se houver piora na solução a troca é desfeita e examinado-se o próximo par de padrões. Caso contrário a troca é mantida, o vetor de combinações é embaralhado novamente e reinicia-se o processo. Este processo é repetido até que a porcentagem β do total de padrões seja analisada.

Algoritmo 5: Pseudocódigo da Busca Local Iterada aplicada ao MDP.

Entrada: π, β, κ.
Gerar solução inicial com a BFS-MDP;
Aplique a busca local de agrupamento de *1-blocks*;
enquanto *critério de parada não for atendido* **faça**
    $\pi' \leftarrow \pi$;
    Perturbe a solução π' em uma proporção α aplicando *swap*;
    Aplique o método *2-Opt* em π' em uma proporção β;
    Aplique a busca local de agrupamento de *1-blocks* em π';
    se $Z^{\pi}_{MDP}(Q^{\pi'}) < Z^{\pi}_{MDP}(Q^{\pi})$ **então**
        $\pi \leftarrow \pi'$
    fim
fim
retorna Q^{π};

Empiricamente, definiu-se os valores dos parâmetros $\alpha = 10\%$ e $\beta = 80\%$; O critério de aceitação utilizado é a diminuição no valor da solução, e o critério de parada é definido como 150 iterações.

4.5 Δ-Avaliação

Uma Δ-*Avaliação* é um método que visa verificar de maneira rápida uma nova solução de um determinado problema obtida através de *movimentos* realizados em uma solução anterior. Geralmente essa avaliação consiste em verificar somente a parte da solução que foi alterada, na tentativa de melhorar o tempo de execução do algoritmo. Neste trabalho foi utilizada a delta avaliação proposta por **?** no método de busca local de agrupamento de *1-blocks* (descrito na seção 3.4.2) a qual consiste em verificar a alteração do número de blocos consecutivo considerando apenas as colunas envolvidas no movimento.

Esta é uma maneira eficiente de avaliar se um movimento específico resulta em uma melhora da solução. Para calcular o número de blocos resultantes é utilizada a expressão 4.1

$$(1 - a) \times b \times (1 - c) + a \times (1 - b) \times c \tag{4.1}$$

Ou a fórmula compacta: $b - ab - bc + ac$ (em que a, b, c são três colunas consecutivas da solução) a fim de determinar a ocorrência de 101 ou 010, que significa o aumento do número de blocos consecutivos. A Tabela 4.4 mostra a aplicação desta expressão na tabela verdade.

A partir da expressão citada anteriormente é calculada o número de blocos da coluna i que foi retirada e o número de blocos da posição j em que a mesma foi re-inserida conforme a

Tabela 4.4: Cálculo do número de blocos consecutivos.

a	b	c	$b - ab - bc + ac$
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0

expressão 4.2, que a partir dos valores das colunas citadas verificam se houve ocorrência dos padrões: 101 ou 010.

$$\delta = -\delta(\pi(i-1), \pi(i), \pi(i+1)) + \delta(\pi(j-1), \pi(j), \pi(j+1)) \tag{4.2}$$

É repetido este processo para todas as linhas e somado iterativamente o valor de δ. Caso o resultado 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. Como são analisadas somente seis colunas da entrada, esta avaliação possuiu complexidade $O(n)$ em que n representa o número de linhas presentes na entrada. Vale ressaltar que a avaliação normal possui complexidade $O(n^2)$ e é repetida várias vezes, valendo a pena o uso da Δ-*Avaliação*.

Capítulo 5

Experimentos

Os experimentos computacionais foram realizados em um computador *Intel core i5 Quad Core* de 3.0 GHz com 8GB RAM sob o sistema operacional Ubuntu 15.10. O código do método proposto foi implementado em linguagem C++, compilado com o compilador g++ 4.4.1 e opção de otimização -O3.

Foram realizados neste trabalho dois tipos distintos de experimentos: experimentos preliminares, descritos na Seção 5.1, que envolvem a geração da solução inicial e o ajuste de parâmetros utilizados na ILS. A partir da melhor configuração do algoritmo proposto foi realizado o segundo conjunto de experimentos, descrito na Seção 5.2, a fim de determinar a número de *descontinuidades* e a quantidade de *blocos consecutivos* obtidos para as instâncias propostas e posterior comparação com os resultados disponíveis na literatura.

5.1 Experimentos Preliminares

No intuito de determinar a melhor configuração do algoritmo proposto, foi realizado um conjunto de testes descritos nas três próximas seções. O primeiro experimento visou encontrar a partir de quatro implementações diferentes da BFS, qual possuía o resultado mais adequado no que tange ao problema abordado neste trabalho. Posteriormente, foi implementado um diferente método de geração da solução inicial encontrado na literatura e comparado com o melhor resultado do experimento anterior. Por fim, foi realizado um experimento visando ajustar os parâmetros da ILS a partir de um conjunto pré-selecionado de valores.

5.1.1 Seleção do Critério Guloso da BFS

Conforme mencionado na Seção 3.2.1, a Busca Largura utiliza uma política de fila para armazenar e selecionar os vértices que serão explorados pela mesma. A maneira em que estes vértices são inseridos na fila pode modificar a busca, gerando soluções diferentes para um mesmo grafo. Neste trabalho a BFS foi utilizada com o propósito de gerar uma solução

inicial, portanto foram implementadas quatro versões diferentes da mesma, que diferem entre si pelo critério guloso que estabelece a maneira em que os vértices são inseridos na fila.

No primeiro critério da BFS os vértices são inseridos de maneira decrescente pelo peso da aresta que liga ao vértice atual até o momento. O segundo critério da BFS verifica e insere os vértices decrescente a partir do seu grau. O terceiro critério da BFS analisa os pesos das arestas dos vértices analisados e insere de maneira decrescente a partir do somatório das mesmas. O quarto critério é semelhante ao anterior diferindo na maneira que os vértices são inseridos, em que são inseridos de maneira crescentes a partir do somatório das arestas.

Os experimentos foram realizados com todos os 5 conjuntos de instâncias considerados neste trabalho. A Tabela 5.1 apresenta um *ranking* ordinal das versões da BFS, em que a classificação, é determinada pelo número de descontinuidades, de maneira que é atribuída a primeira posição ao critério da BFS que apresentou o melhor resultado e assim sucessivamente. Na referida tabela, a coluna *Conjunto* refere-se ao conjunto de instância analisado e as colunas BFS-1, BFS-2, BFS-3, BFS-4, apresentam o *ranking* ordinal médio para os métodos comparados.

Tabela 5.1: Comparação de resultados da BFS.

Conjunto	BFS-1	BFS-2	BFS-3	BFS-4
Hadaddi	2,40	2,05	2,57	2,21
SCOOP	1,65	1,65	2,10	1,55
First Constraint Modeling Challenge	2,09	1,93	2,33	2,47
VLSI	1,68	1,47	1,47	1,32
Chu e Stuckey	2,30	2,08	2,41	2,30
Média	**2,23**	**2,03**	**2,36**	**2,18**

Analisando os dados apresentados, é possível destacar que a versão BFS-2 apresentou melhor *ranking* médio em 3 conjuntos de instâncias seguida pela versão BFS-4 em 2 conjuntos. As versões BFS-1 e BFS-3 apresentaram resultados medianos, não apresentando melhor *ranking* médio em nenhum conjunto de instâncias. Ao analisar o *ranking* médio total obtido é possivel concluir que a versão BFS-2 obteve o melhor resultado com 2,03 de *ranking* médio seguida por BFS-4, BFS-3 e BFS-1. Portanto a versão BFS-2 foi selecionada para gerar a solução inicial neste trabalho.

5.1.2 Geração da Solução Inicial

Conforme mencionado na Seção 3.3, a Busca Local Iterada inicia-se com a aplicação de um método para a geração de uma solução inicial. Neste trabalho foram implementados dois diferentes métodos e comparados em relação ao valor da solução e o tempo gasto para a execução.

O primeiro método foi a BFS mencionada na Seção 3.2.1. O segundo método para a geração da solução inicial foi semelhante ao proposto por **?**, que baseia-se na criação de soluções

aleatórias e através destas, selecionar a que possui melhor valor conforme a função objetivo.

Os experimentos foram realizados com os conjuntos A e D das instâncias disponibilizadas por (**?**). Para o segundo método, foram criadas 500 soluções aleatórias. Os resultados obtidos foram analisados e foi constatado que em cerca de 60% das instâncias, a solução obtida pela BFS foi superior a solução aleatória. Além disso, em 60% dos casos o tempo de execução da BFS foi menor que o da solução aleatória. De acordo com estes dados, optou-se pelo uso da BFS como solução inicial.

5.1.3 Ajuste de Parâmetros

Conforme apresentado no Algoritmo 5, existem três parâmetros da ILS que precisam ser ajustados a fim de se encontrar a melhor configuração: a porcentagem de aplicação de busca local, a porcentagem de aplicação de perturbação e o número de iterações que este método será executado. Para a escolha dos melhores valores foi utilizada o *irace* (**?**), um pacote que implementa uma série de procedimentos de configuração automática, em particular oferece o procedimento de corrida iterada, que vem sendo usado com sucesso para configurar automaticamente vários algoritmos do estado da arte.

Nos testes realizados, foram escolhidas aleatoriamente instâncias representativas de todos os conjuntos disponíveis neste trabalho. Para os parâmetros foram pré-selecionados alguns valores conforme apresentados na Tabela 5.2, 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 5.2: Ajuste de parâmetros usando *irace*.

Parâmetros	*Valores*
Busca Local(%)	{10, 20, 30, 40, 50, 60, 70, 80, 90, 100}
Perturbação(%)	{5, 10, 15, 20, 25, 30, 35, 40, 45, 50}
Iterações	{50, 100, 150}

Como resultado, o *irace* recomendou três configurações que foram as melhores. É preciso destacar que uma configuração não domina a outra, ou seja, em alguns testes uma configuração foi superior a outra em outros testes aconteceu o resultado inverso:

- Busca Local = 100%, Perturbação = 30% e Iterações = 150;
- Busca Local = 100%, Perturbação = 5% e Iterações = 150;
- Busca Local = 80%, Perturbação = 10% e Iterações = 150.

Entre as configurações recomendadas, foi escolhida a terceira, por apresentar uma menor porcentagem de busca local o que pode gerar uma menor tempo de execução do algoritmo. Ainda assim, conforme reportado na seção seguinte, este conjunto de valores não foi suficiente

para executar o método proposto em tempo aceitável para as maiores instâncias. Nos casos em que isto ocorreu, os valores foram diminuídos, observando-se a qualidade das soluções geradas e o tempo de execução.

5.2 Comparação de Resultados

São consideradas as instâncias propostas por **?**. Este conjunto de quarenta e cinco instâncias artificiais é dividido em nove grupos (A-I), com cinco instâncias cada. Cada instância foi gerada aleatoriamente atendendo o critério de que cada coluna possua pelo menos elemento não nulo, e cada linha possua pelo menos dois valores não nulos. Estas instâncias foram originalmente utilizadas para o Problema de Minimização de Blocos Consecutivos, um problema correlato ao Problema de Minimização de Descontinuidades, conforme discutido no Capítulo 2. Desta forma, os resultados são apresentados para ambos os objetivos.

Adicionalmente, foram realizados experimentos computacionais com quatro conjuntos de instâncias que possuem o mesmo formato do conjunto anterior, apesar de serem originalmente propostas para problemas correlatos ao MDP. Este conjuntos são divididos em diferentes seções na sequência. Os resultados obtidos para estas instâncias são inéditos no que tange ao Problema de Minimização de Descontinuidades e ao Problema de Blocos Consecutivos, portanto não foi possível realizar comparações com outros autores. Entretanto, os resultados obtidos neste trabalho poderão ser utilizados como base de comparação para trabalhos futuros, contribuindo desta forma para a ampliação do conjunto de *benchmark* do MDP e ao Problema de Blocos Consecutivos. Os resultados detalhados podem ser obtidos sob consulta do autor deste trabalho.

Nas tabelas apresentadas nas seções 5.2.1 à 5.2.5 a coluna *Instância* refere-se ao nome das instâncias, |P| representa a quantidade de peças e |S| indica a quantidade de padrões . Os tempos médios de execução (expressos em segundos) são apresentados na coluna T, o valor da melhor solução obtida para o MDP é apresentado na coluna *ILS-MDP* e a melhor solução obtida para Blocos Consecutivos é apresentado na coluna *ILS-Blocos*. Na coluna *Média-MDP* é apresentado a média das descontinuidades após as 10 execuções, de maneira análoga a coluna *Média-Blocos* apresenta a média em relação ao número de blocos. Por fim, a coluna σ indica o desvio padrão entre as soluções encontradas para a instância após as 10 execuções individuais, vale ressaltar que o desvio padrão é igual para ambas as funções objetivos. Adicionalmente foi verificado o índice de convergência do algoritmo, ou seja, em qual iteração o mesmo encontrou a melhor solução em cada execução. Estes dados foram analisados e descritos para cada conjunto de instâncias.

5.2.1 Instâncias ?

Experimentos preliminares apontaram que se tornaria inviável executar o algoritmo proposto com a melhor configuração encontrada, citada na Seção 5.1.3. Portanto, os valores de porcentagem de busca local, porcentagem de perturbação, o número de iterações e a porcentagem de linhas da matriz exploradas pela busca local agrupamento de *1-blocks* foram modificados com o intuito de viabilizar os experimentos para este conjunto de instâncias. Nota-se que as definições originais foram modificadas somente para instâncias dos grupos de E até I. A versão final do algoritmo para este conjunto utilizou os valores de parâmetros apresentados abaixo:

- Para as instâncias do grupo A até D: Busca Local = 80%, Perturbação = 10%, Iterações = 150, Linhas Verificadas = 100%;
- Para as instâncias do grupo E até G: Busca Local = 50%, Perturbação = 10%, Iterações = 100, Linhas Verificadas = 50%;
- Para as instâncias do grupo H e do grupo I: Busca Local = 20%, Perturbação = 10%, Iterações = 30, Linhas Verificadas = 20%.

Na Tabela 5.3 é apresentada uma comparação entre o método proposto neste trabalho e os resultados obtidos por **?**, que disponibilizou as instâncias e originalmente trata do problema de minimização de blocos consecutivos. Os valores apresentados na referida tabela são as médias de cada um dos grupos. Adicionalmente, a tabela apresenta a coluna $|B|$, que indica a quantidade média de blocos iniciais em cada instância do grupo. O valor médio para o número de blocos obtido pelo trabalho tomado como referência para cada conjunto de instância é mostrado na coluna *Haddadi*, assim como o tempo de execução (em segundos) apresentado na coluna *T*. A coluna *gap* apresenta a distância percentual entre os resultados obtidos e os de referência, calculada como $100\times((ILS\_Blocos$ - $Haddadi)$ / $Haddadi)$. Resultados individuais para estas instâncias podem ser visualizados no Apêndice desta monografia.

Tabela 5.3: Comparação de resultados.

Grupo	|P|	|S|	|B|	*ILS-Blocos*	*Média-Blocos*	*T*	σ	*Haddadi*	*T*	*gap*
A	100	200	416,0	262	267,06	74,19	2,95	253,00	0,45	3,56
B	100	200	955,6	671,60	675,42	451,50	2,28	695,80	0,62	-3,48
C	100	200	1789,4	1307,20	1311,30	769,34	2,44	1358,60	0,79	-3,78
D	100	500	1027,2	601,20	607,06	1.015,27	3,73	552,00	3,32	8,91
E	100	500	2370,2	1576,20	1585,68	4.220,57	5,40	1616,00	4,59	-2,46
F	100	500	4529,2	3162,20	3169,16	8.761,34	4,35	3308,40	5,52	-4,42
G	100	1000	2078,8	1178,20	1185,40	4.772,54	4,50	1072,40	14,03	9,87
H	100	1000	4778,4	3067,20	3068,78	7.612,72	7,09	3125,40	22,20	-1,86
I	100	1000	8998,8	6109,60	6124,40	16.631,09	8,01	6375,40	27,12	-4,17

A Figura 5.1 apresenta a comparação do número de blocos obtidos pelas soluções geradas pelos métodos comparados.

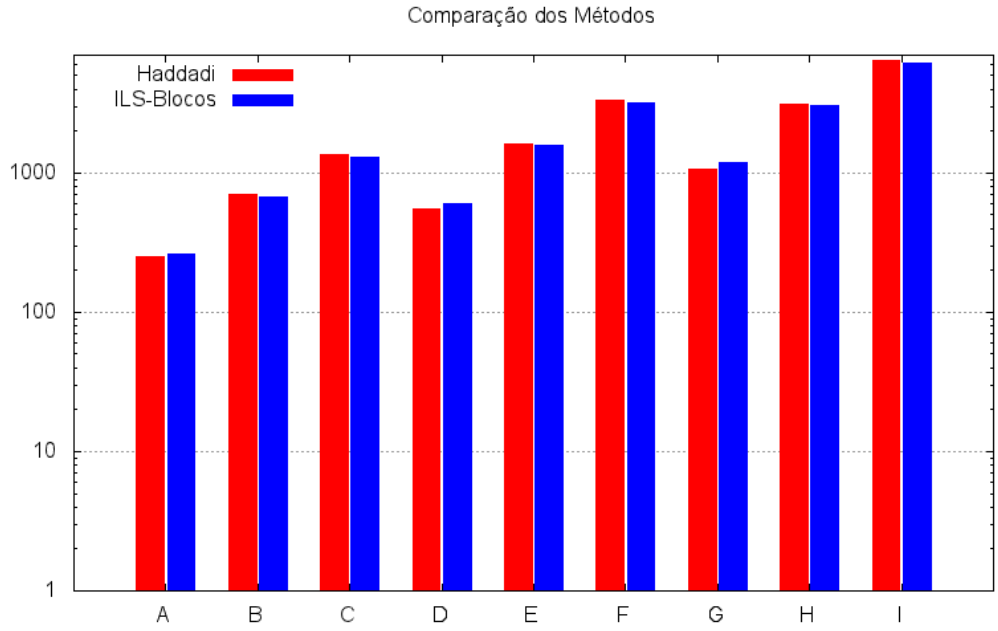


Figura 5.1: Gráfico de comparação de resultados.

De acordo com os dados apresentados, é possível verificar que a *ILS* conseguiu reduzir em aproximadamente 32,32% o número inicial de blocos consecutivos, considerando os melhores resultados encontrados pela ILS. 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.

Em relação ao método de referência, o *gap* médio foi de apenas 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, G) o método proposto não conseguiu igualar as soluções de referência, obtendo valores de *gap* que varia, entre 3,56% e 9,87%.

Comparando o tempo de execução 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, que representa para o maior grupo uma diferença de aproximadamente 613,24 vezes no tempo de execução. Entretanto, cabe ressaltar que, devido às dimensões das instâncias (500 e 1000 colunas) e à qualidade das soluções geradas, esse valor é considerado aceitável. 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, a melhor solução é encontrada em menos que a metade das do tempo de execução do algoritmo, evidenciando a rápida convergência relativa.

No que tange a robustez, é notável a qualidade da solução, em que o desvio padrão médio geral foi 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. Outro fato que demonstra a qualidade das soluções é que justamente para os grupos em que se utilizou parâmetros com valores reduzidos, foram encontrados valores melhores que o estado da arte.

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, mesmo reduzindo consideravelmente os parâmetros envolvidos na execução do algoritmo. A Tabela 5.4 apresenta os valores obtidos para as mesmas instâncias, porém, em relação ao número de *descontinuidades*.

Tabela 5.4: Resultados para *descontinuidades*.

Instância	|P|	|S|	*ILS-MDP*	*Média-MDP*	T	σ
A	100	200	162	167,06	74,19	2,95
B	100	200	571,60	575,42	451,50	2,28
C	100	200	1297,20	1211,30	769,34	2,44
D	100	500	501,20	507,06	1.015,27	3,73
E	100	500	1476,20	1485,68	4.220,57	5,40
F	100	500	3062,20	3069,16	8.761,34	4,35
G	100	1000	1078,20	1085,40	4.772,54	4,50
H	100	1000	2967,20	2968,78	7.612,72	7,09
I	100	1000	6009,60	6024,40	16.631,09	8,01

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.

5.2.2 Instâncias SCOOP

Este conjunto contém instâncias reais fornecido pelo SCOOP *Consortium*[1], possuindo 187 instâncias do *Problema de Minimizição de Pilhas Abertas* (*Minimization of Open Stacks Problem* – MOSP), um problema correlato com o MDP que visa minimizar a utilização de estoque intermediário, bem como a manipulação desnecessária dos produtos fabricados. Porém, a maioria destas instâncias são muito pequenas (por exemplo, 2 linhas e 2 colunas) tendo soluções triviais, podendo ser descartadas. Deste conjunto, 24 instâncias com dimensões significativas,

[1] http://www.scoop-project.net

variando de 10 linhas e 7 colunas a 49 linhas e 134 colunas, foram selecionadas para serem usadas nos experimentos. A Tabela 5.5 apresenta os valores obtidos para cada instância.

Tabela 5.5: Resultados para as instâncias selecionadas do *SCOOP*.

Instância	|P|	|S|	*ILS-MDP*	*ILS-Blocos*	*Média-MDP*	*Média-Blocos*	T	σ
A_AP-9.d_3	20	16	5	25	5,00	25,00	0,15	0,00
A_AP-9.d_6	31	20	4	35	4,50	35,50	0,35	0,53
A_AP-9.d_10	20	13	5	25	5,00	25,00	0,09	0,00
A_AP-9.d_11	27	21	7	34	7,80	34,80	0,23	0,63
A_FA+AA-_1	107	37	17	122	17,70	122,70	13,24	0,48
A_FA+AA-_2	75	19	8	83	8,20	83,20	0,75	0,42
A_FA+AA-_6	79	21	12	91	12,00	91,00	1,78	0,00
A_FA+AA-_8	100	100	16	98	16,00	98,00	4,28	0,00
A_FA+AA-_11	99	28	12	111	12,70	111,70	4,39	0,48
A_FA+AA-_12	75	25	8	83	9,20	84,20	0,82	0,63
A_FA+AA-_13	134	37	31	165	31,70	165,70	14,19	0,48
A_FA+AA-_15	68	18	6	74	6,00	74,00	0,83	0,00
B_12F18_11	21	15	5	26	5,50	26,50	0,13	0,53
B_12M18_12	31	22	9	37	9,00	37,00	0,44	0,00
B_18AB1_32	15	11	7	21	7,70	21,70	0,03	0,48
B_18CR1_33	20	18	3	22	3,00	22,00	0,07	0,00
B_22X18_50	14	11	13	27	13,00	27,00	0,03	0,00
B_23B25_52	29	21	3	30	3,30	30,30	0,25	0,48
B_39Q18_82	14	10	0	14	0,70	14,70	0,01	0,48
B_42F22_93	18	10	2	19	2,60	19,60	0,03	0,52
B_CARLET_137	14	12	4	17	4,60	17,60	0,02	0,52
B_CUC28A_138	37	26	7	38	7,80	38,80	0,40	0,42
B_GTM18A_139	24	20	3	27	3,30	27,30	0,18	0,48
B_REVAL_145	60	49	13	73	14,90	74,90	9,29	1,29

Analisando os dados é possível concluir que a *ILS* apresentou bons resultados para este conjunto, apresentando desvio padrão igual à 0,00 para 8 instâncias e o maior desvio padrão encontrado foi de apenas 1,29, demostrando a robustez do método proposto. O tempo de execução médio foi de 2,17 segundos sendo fortemente influenciado pelas instâncias *A_FA+AA-_1,A_FA+AA-_13,B_REVAL_145*, as maiores instâncias do conjunto. O desvio padrão médio em relação ao tempo foi de apenas 0,04, reforçando o argumento de robustez. Para este conjunto de instâncias, o percentual de convergência médio foi muito baixo, com valor de 14,70%. Destaca-se a instância *B_18CR1_33* em que o valor de convergência da mesma foi de 0,00, ou seja, a solução inicial foi a melhor encontrada, evidenciando a qualidade e a importância da solução inicial para o algoritmo proposto.

5.2.3 Instâncias VLSI ?

Oriundas do problema de Leiaute de Matrizes de Portas (problema equivalente ao MOSP), este conjunto possui 24 instâncias de empresas da Ásia e foram introduzidas por **?**. Estas

instâncias possuem matrizes de dimensões que variam entre 5×7 e 202×141. A Tabela 5.6 apresenta os resultados obtidos pelo ILS para este conjunto de instâncias.

Tabela 5.6: Resultados para as instâncias selecionadas VLSI.

Instância	|P|	|S|	*ILS-MDP*	*ILS-Blocos*	*Média-MDP*	*Média-Blocos*	T	σ
v4000	10	17	14	24	14,60	24,60	0,04	0,84
v4050	13	16	11	24	11,20	24,20	0,04	0,42
v4090	23	27	27	50	29,20	52,20	0,15	1,40
v4470	37	47	27	64	28,80	65,80	7,89	1,14
vc1	15	25	21	36	21,80	36,80	0,05	0,42
vw1	5	7	5	10	5,00	10,00	0,00	0,00
vw2	8	8	4	12	4,00	12,00	0,01	0,00
w1	18	21	7	25	7,40	25,40	0,07	0,52
w2	48	33	44	92	44,30	92,30	1,45	0,48
w3	84	70	74	158	76,00	160,00	46,15	1,83
w4	202	141	128	330	130,70	332,70	1.113,14	1,42
wan	9	7	4	13	4,00	13,00	0,00	0,00
wli	11	10	5	16	5,60	16,60	0,01	0,52
wsn	17	25	14	31	14,00	31,00	0,16	0,00
x0	40	48	32	72	33,40	73,40	5,90	1,17
x1	5	10	6	11	6,00	11,00	0,02	0,00
x2	5	15	10	16	10,00	16,00	0,05	0,00
x3	7	21	15	22	15,00	22,00	0,13	0,00
x4	12	8	0	12	0,00	12,00	0,01	0,00
x5	24	16	0	24	0,00	24,00	0,18	0,00
x6	49	32	0	48	0,00	48,00	2,96	0,00
x7	7	8	3	10	3,00	10,00	0,01	0,00
x8	11	16	7	18	7,00	18,00	0,09	0,00
x9	19	32	15	34	15,00	34,00	1,05	0,00

Novamente, é possível destacar a robustez *ILS* no que tange ao valor da solução, em que o desvio padrão em 13 das 24 das instâncias foi igual à 0,00 e o maior desvio padrão foi 1,83. O tempo de execução médio foi 47,18 segundos fortemente influenciado pela instância *w4* que apresentou média de tempo de 1.113,14 segundos, principalmente por suas dimensões (202×141), serem muito maiores que o restante das instâncias do conjunto. Essa instância também apresentou o maior desvio padrão em relação ao tempo, com valor de 73,92 enquanto o segundo maior valor foi de apenas 2,49. Pode-se observar que o tempo de execução do método proposto cresce muito com o aumento do número de padrões e peças. Para este conjunto de instâncias o percentual média de convergência foi extremamente baixo com valor de 4,79%, o que demonstra rapidez do algoritmo para determinar a melhor solução. Destaca-se adicionalmente a qualidade da solução inicial, em que metade das instâncias o percentual de convergência foi de 0,00.

5.2.4 Instâncias ?

O terceiro conjunto de instâncias proposto por **?**, contém 200 instâncias MOSP de dimensões que variam entre 30×30 e 125×125. Os problemas foram agrupados em subconjuntos de 25 instâncias de acordo com o número de padrões e peças. A Tabela 5.7 apresenta os resultados obtidos.Os valores apresentados na referida tabela são as médias de cada um dos grupos.

Tabela 5.7: Resultados para as instâncias Chue e Stuckey.

Grupo	|P|	|S|	*ILS-MDP*	*ILS-Blocos*	*Média-MDP*	*Média-Blocos*	T	σ
Random-30-30	30	30	70,5	100,88	71,72	102,11	2,05	0,84
Random-40-40	40	40	101,72	141,72	103,02	143,02	6,85	0,93
Random-50-50	50	50	116,70	166,70	118,29	168,29	17,81	0,99
Random-50-100	50	100	293,20	343,20	296,22	346,22	154,83	1,83
Random-75-75	75	75	235,12	310,12	237,22	312,22	94,82	1,21
Random-100-50	100	50	139,28	239,28	140,61	240,61	32,90	0,78
Random100-100	100	100	335,33	435,33	338,01	438,01	311,40	1,47
Random125-125	125	125	420,20	545,20	422,90	547,90	800,99	1,53

A *ILS* mais uma vez demostrou robustez, apresentado desvio padrão médio geral igual à 1,20, sendo que o grupo que apresentou o maior desvio padrão médio foi o *Random-50-100* com o valor de 1,83. Pode-se observar que o tempo de execução cresce de maneira mais rápida quando se aumenta o número de padrões em relação ao aumento do número de peças. Esse fato pode ser comprovado para o grupo de instâncias *Random-50-50* com tempo médio de 17,81s: ao dobrar o número de peças e manter o número de padrões o tempo de execução aumentou para 32,90 segundos (conjunto *Random-100-50*). Já ao manter o número de peças e dobrar o número de padrões, o tempo de execução aumentou para 154,83 segundos (conjunto *Random-50-100*), o que representa um aumento de aproximadamente 9,08 vezes. Para este conjunto o percentual médio de convergência do algoritmo foi de 38,30%, variando de 29,64% para o grupo *Random-30-30* até 43,91% para o grupo *Random100-100*. Estes dados demonstram para este conjunto o melhor valor é encontrado aproximadamente nos 38,66% iniciais do tempo de execução da ILS, um valor consideravelmente baixo.

5.2.5 Instâncias do *First Constraint Modeling Challenge* (?)

Este conjunto contém 46 instâncias MOSP geradas aleatoriamente propostas para o *First Constraint Modeling Challenge* (**?**), desafio mundial em que vários autores submeteram diferentes instâncias e métodos de solução. Os resultados obtidos são apresentados na Tabela 5.8.

Para este conjunto pode-se observar que o desvio padrão individual foi baixo, variando de 0,00 para quatro instâncias a 3,31 para a instância *NWRS7*. O desvio padrão médio geral també é baixo, igual à 0,82. Esses dados demonstram que o método é robusto para esse

conjunto de instâncias. Com relação ao tempo de execução, o desvio padrão médio para todas as instâncias foi baixo, apenas 0,52s. Os subconjuntos que exigiram o maior esforço computacional foram *GP* e *SP*, respectivamente com tempo médio de execução de 192,83s e 88,34s. Isto é devido aos subconjuntos conterem as maiores instâncias no que tange ao número de padrões e peças, por exemplo, dimensões 50×50 e 100×100. Novamente, o percentual médio de convergência do algoritmo foi baixo com valor de 28,83%, o que representa que o o melhor valor foi encontrado em média na iteração 44 de 150, um valor que demonstra a rápida convergência do algoritmo para este conjunto.

Tabela 5.8: Resultados para as instâncias selecionadas do *First Constraint Modeling Challenge*.

Instância	|P|	|S|	*ILS-MDP*	*ILS-Blocos*	*Média-MDP*	*Média-Blocos*	T	σ
GP1	50	50	144	194	144,90	194,90	12,86	0,88
GP2	50	50	237	287	239,50	289,50	23,38	1,51
GP3	50	50	226	276	227,20	277,20	25,94	1,03
GP4	100	100	110	160	112,10	162,10	9,88	1,45
GP5	100	100	646	746	647,70	747,70	423,13	0,82
GP6	100	100	562	662	564,80	664,80	429,36	1,55
GP7	100	100	586	686	589,80	689,80	427,49	2,20
GP8	100	100	449	549	451,10	551,10	378,67	1,45
Miller_20_40	20	40	25	45	25,00	45,00	4,79	0,00
NWRS1	10	20	8	18	8,00	18,00	0,01	0,00
NWRS2	10	20	8	18	9,00	19,00	0,04	0,82
NWRS3	15	25	20	35	22,00	37,00	0,11	1,15
NWRS4	15	25	24	39	24,80	39,80	0,17	0,63
NWRS5	20	30	53	73	55,30	75,30	0,56	2,21
NWRS6	20	30	50	70	52,10	72,10	0,83	0,99
NWRS7	25	60	67	92	70,90	95,90	2,68	3,31
NWRS8	25	60	92	117	96,80	121,80	5,45	2,44
SP1	25	25	15	40	17,00	42,00	0,22	1,56
SP2	50	50	64	114	65,40	115,40	10,31	1,35
SP3	75	75	136	211	137,30	212,30	75,70	0,67
SP4	100	100	240	340	242,80	342,80	267,14	1,48
Shaw1	20	20	24	44	24,10	44,10	0,46	0,32
Shaw10	20	20	25	45	25,60	45,60	0,38	0,52
Shaw11	20	20	26	46	26,60	46,60	0,47	0,52
Shaw12	20	20	32	52	32,10	52,10	0,50	0,32
Shaw13	20	20	21	41	21,10	41,10	0,38	0,32
Shaw14	20	20	25	45	25,90	45,90	0,33	0,57
Shaw15	20	20	25	45	25,70	45,70	0,45	0,67
Shaw16	20	20	24	44	26,00	46,00	0,44	1,41
Shaw17	20	20	23	43	23,40	43,40	0,46	0,52
Shaw18	20	20	28	48	28,10	48,10	0,45	0,32
Shaw19	20	20	24	44	25,40	45,40	0,36	0,84
Shaw2	20	20	23	43	23,10	43,10	0,38	0,32
Shaw20	20	20	26	46	26,80	46,80	0,49	0,63
Shaw21	20	20	21	41	21,30	41,30	0,45	0,48
Shaw22	20	20	25	45	26,20	46,20	0,38	0,63
Shaw23	20	20	25	45	26,00	46,00	0,37	0,82
Shaw24	20	20	30	50	30,20	50,20	0,53	0,42
Shaw25	20	20	26	46	26,40	46,40	0,46	0,52
Shaw3	20	20	25	45	25,90	45,90	0,45	0,32
Shaw4	20	20	23	43	23,20	43,20	0,53	0,42
Shaw5	20	20	27	47	27,00	47,00	0,38	0,00
Shaw6	20	20	32	52	32,60	52,60	0,45	0,70
Shaw7	20	20	25	45	25,50	45,50	0,37	0,53
Shaw8	20	20	27	47	27,00	47,00	0,46	0,00
Shaw9	20	20	21	41	21,00	41,00	0,39	0,00

Capítulo 6

Conclusões

Neste trabalho foi abordado o Problema de Minimização de Descontinuidades (MDP), problema NP-Difícil de sequenciamento de padrões de corte com ampla aplicação industria. Este é um importante problema no contexto industrial, que pode reduzir os custos operacionais da produção de bens de consumo e também auxiliar na homogeneização das características físicas dos mesmos. Apesar da importância do problema, existem poucas propostas na literatura para a sua resolução.

Foram propostos uma nova representação em grafos para o problema, a utilização da Busca em Largura para a geração da solução inicial e a estratégia inédita de Busca Local Iterada (ou *ILS*) na qual foi utilizado os métodos *2-Opt* e busca local de agrupamento de *1-blocks* como busca locais e o método *2-swap* como perturbação.

Os experimentos computacionais envolveram aproximadamente 350 instâncias de cinco diferentes conjuntos da literatura, incluindo instâncias reais e artificiais. Para o único conjunto *benchmark* disponível na literatura, a ILS gerou novos melhores resultados para seis dos nove grupos. De acordo com as análises realizadas, evidenciou-se que a ILS possui melhor performance em instâncias densas do que em instâncias esparsas. Para outros conjuntos de instâncias inéditos no contexto do MDP, foram apresentados resultados para formação de novos *benchmarks*. Em geral, a ILS apresentou robustez no que tange ao valor da solução e o tempo de execução, apresentando em ambos os casos desvio padrão baixo. Foi verificado também a rápida convergência do algoritmo proposto que no pior caso a melhor solução foi encontrada na metade do tempo de execução.

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

Apêndice

Nas nove tabelas a seguir, a coluna *Instância* refere-se ao nome das instâncias, $|P|$ representa a quantidade de peças e $|S|$ indica a quantidade de padrões. Os tempos médios de execução (expressos em segundos) são apresentados na coluna T, o valor da melhor solução obtida para o MDP é apresentado na coluna *ILS-MDP* e a melhor solução obtida para Blocos Consecutivos é apresentado na coluna *ILS-Blocos*. Na coluna *Média-MDP* é apresentado a média das descontinuidades após as 10 execuções independentes do método proposto. De maneira análoga, a coluna *Média-Blocos* apresenta a média em relação ao número de blocos. Por fim, a coluna σ indica o desvio padrão entre as soluções encontradas para a instância após as 10 execuções independentes. Vale ressaltar que o desvio padrão é igual para ambas as funções objetivos.

Tabela 6.1: Instâncias do grupo A.

Instância	|P|	|S|	*ILS-MDP*	*ILS-Blocos*	*Média-MDP*	*Média-Blocos*	T	σ
A1	100	200	168	258	162,3	262,3	74,52	2,67
A2	100	200	157	257	161,1	261,1	69,21	2,33
A3	100	200	148	248	153,5	253,5	75,23	3,66
A4	100	200	162	272	179,3	279,3	79,45	3,56
A5	100	200	175	275	179,1	279,1	71,25	2,51

Tabela 6.2: Instâncias do grupo B.

Instância	|P|	|S|	*ILS-MDP*	*ILS-Blocos*	*Média-MDP*	*Média-Blocos*	T	σ
B1	100	200	578	678	579,7	679,7	445,58	1,52
B2	100	200	573	673	577,2	677,2	450,25	2,06
B3	100	200	568	668	572,7	672,7	427,59	3,20
B4	100	200	582	682	587,1	687,1	449,13	3,07
B5	100	200	557	657	559,6	659,6	458,00	1,56

Tabela 6.3: Instâncias do grupo C.

Instância	|P|	|S|	*ILS-MDP*	*ILS-Blocos*	*Média-MDP*	*Média-Blocos*	T	σ
C1	100	200	1191	1291	1195,8	1295,8	445,58	1,99
C2	100	200	1189	1289	1193,9	1293,9	450,25	2,47
C3	100	200	1195	1295	1198,30	1298,30	427,59	2,75
C4	100	200	1204	1304	1209,70	1309,70	449,13	3,65
C5	100	200	1257	1357	1258,80	1358,80	458,00	1,32

Tabela 6.4: Instâncias do grupo D.

Instância	|P|	|S|	*ILS-MDP*	*ILS-Blocos*	*Média-MDP*	*Média-Blocos*	T	σ
D1	100	500	522	622	528,50	628,50	1065,49	4,25
D2	100	500	524	624	532,10	632,10	1163,60	4,75
D3	100	500	471	571	475,10	575,10	880,70	3,07
D4	100	500	511	611	415,60	615,60	1083,23	2,59
D5	100	500	478	578	484,00	584,00	886,44	3,97

Tabela 6.5: Instâncias do grupo E.

Instância	|P|	|S|	*ILS-MDP*	*ILS-Blocos*	*Média-MDP*	*Média-Blocos*	T	σ
E1	100	500	1485	1585	1491,40	1591,40	4379,62	3,75
E2	100	500	1468	1568	1479,90	1579,90	3921,81	6,77
E3	100	500	1461	1561	1473,10	1573,10	4075,23	5,86
E4	100	500	1466	1566	1472,80	1572,80	4280,69	5,59
E5	100	500	1501	1601	1511,20	1611,20	4459,56	5,05

Tabela 6.6: Instâncias do grupo F.

Instância	|P|	|S|	*ILS-MDP*	*ILS-Blocos*	*Média-MDP*	*Média-Blocos*	T	σ
F1	100	500	3010	3110	3017,80	3117,80	8.877,21	4,57
F2	100	500	3065	3165	3072,10	3172,10	8.756,13	4,72
F3	100	500	3070	3170	3077,60	3177,60	8.719,98	5,10
F4	100	500	3075	3175	3082,00	3182,00	8.733,59	4,00
F5	100	500	3091	3191	3096,30	3196,30	8.719,74	3,37

Tabela 6.7: Instâncias do grupo G.

Instância	|P|	|S|	*ILS-MDP*	*ILS-Blocos*	*Média-MDP*	*Média-Blocos*	T	σ
G1	100	1000	1068	1168	1077,20	1177,20	4.548,29	5,61
G2	100	1000	1060	1160	1066,70	1166,70	4.858,52	5,62
G3	100	1000	1096	1196	1102,30	1202,30	4.951,51	4,08
G4	100	1000	1125	1225	1130,90	1230,90	5.071,77	3,48
G5	100	1000	1042	1142	1049,90	1149,90	4.432,63	3,73

Tabela 6.8: Instâncias do grupo H.

Instância	|P|	|S|	*ILS-MDP*	*ILS-Blocos*	*Média-MDP*	*Média-Blocos*	T	σ
H1	100	1000	3005	3105	3022,00	3122,00	7.925,21	9,07
H2	100	1000	2914	3014	2921,80	3021,80	7.566,65	6,16
H3	100	1000	2995	3095	3006,10	3106,10	7.528,95	5,78
H4	100	1000	2836	2936	2849,80	2949,80	7.125,43	5,77
H5	100	1000	3086	3186	3097,40	3197,40	7.917,33	8,68

Tabela 6.9: Instâncias do grupo I.

Instância	|P|	|S|	*ILS-MDP*	*ILS-Blocos*	*Média-MDP*	*Média-Blocos*	T	σ
I1	100	1000	6034	6134	6047,60	6147,60	16.894,57	7,44
I2	100	1000	6059	6159	6083,30	6183,30	16.606,34	9,87
I3	100	1000	5968	6068	5981,10	6081,10	16.368,00	8,49
I4	100	1000	6002	6102	6012,30	6112,30	16.598,18	8,68
I5	100	1000	5985	6085	5997,70	6097,70	16.688,36	6,22

Referências Bibliográficas

Becceneri, J. (1999). O problema de sequenciamento de padrões para minimização do número máximo de pilhas abertas em ambientes de corte industriais. *São José dos Campos. Tese (Doutorado em Ciências no Curso de Engenharia Eletrônica e Computação na Área de Informática)–ITA, Centro Tecnológico Aeroespacial*.

Chu, G. e Stuckey, P. J. (2009). Minimizing the maximum number of open stacks by customer search. In *International Conference on Principles and Practice of Constraint Programming*, pp. 242–257. Springer.

Cormen, T. H. (2009). *Introduction to algorithms*. MIT press.

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*, pp. 157–172. ACM.

Dom, M. (2009). *Recognition, Generation, and Application of Binary Matrices with the Consecutive Ones Property*. Cuvillier.

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

Garey, M. R. e Johnson, D. S. (1979). A guide to the theory of np-completeness. *WH Freemann, New York*.

Gilmore, P. C. e Gomory, R. E. (1961). A linear programming approach to the cutting-stock problem. *Operations research*, 9(6):849–859.

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.

Hu, Y. H. e Chen, S.-J. (1990). Gm plan: a gate matrix layout algorithm based on artificial intelligence planning techniques. *IEEE transactions on computer-aided design of integrated circuits and systems*, 9(8):836–845.

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*, pp. 320–353. Springer.

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

Madsen, O. B. (1988). An application of travelling-salesman routines to solve pattern-allocation problems in the glass industry. *Journal of the Operational Research Society*, pp. 249–256.

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*, pp. 3717–3724.

Smith, B. e Gent, I. (2005). Constraint modelling challenge 2005. In *IJCAI 2005 Fifth Workshop on Modelling and Solving Problems with Constraints*, pp. 1–8.

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.