JÚNIOR RHIS LIMA

Orientador: Marco Antonio Moreira de Carvalho

ALGORITMOS METAHEURÍSTICOS APLICADOS AO PROBLEMA DE MINIMIZAÇÃO DE PILHAS ABERTAS

Ouro Preto

Agosto de 2016

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

ALGORITMOS METAHEURÍSTICOS APLICADOS AO PROBLEMA DE MINIMIZAÇÃO DE PILHAS ABERTAS

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.

JÚNIOR RHIS LIMA

Ouro Preto
Agosto de 2016

UNIVERSIDADE FEDERAL DE OURO PRETO

FOLHA DE APROVAÇÃO

Algoritmos Metaheurísticos Aplicados ao Problema de Minimização de Pilhas Abertas

JÚNIOR RHIS LIMA

Monografia defendida e aprovada pela banca examinadora constituída por:

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

Dr. Joubert de Castro Lima
Universidade Federal de Ouro Preto

Dr. André Luís Silva
Universidade Federal de Ouro Preto

Ouro Preto, Agosto de 2016

Resumo

Este trabalho tem como objetivo propor novos algoritmos para solução do Problema de Minimização de Pilhas Abertas, um problema de sequenciamento de padrões de corte cujo objetivo é minimizar a utilização de estoque intermediário, bem como a manipulação desnecessária dos produtos fabricados. Nesta monografia é reportada a aplicação de dois métodos metaheurísticos a este problema NP-Difícil de ampla aplicação prática: Descida em Vizinhança Variável (*Variable Neighborhood Descent*) e Descida Mais Rápida (*Steepest Descent*). Vale ressaltar que esta é a primeira aplicação reportada do método Descida em Vizinhança Variável aplicado a solução deste problema. Os experimentos computacionais envolveram 595 instâncias de cinco diferentes conjuntos da literatura. Os resultados reportados demonstram que não houve dominância entre os métodos propostos, os quais foram capazes de obter boa parte das soluções ótimas para as instâncias consideradas, obtendo uma pequena distância percentual das mesmas nos demais casos, o que mostra a robustez dos métodos propostos.

Abstract

This work aims to propose new algorithms to solve the Minimization Open Stacks Problem, a cutting pattern sequencing problem, whose objective is to minimize the use of intermediate stock, as well as the unnecessary handling of manufactured products. This monography reports the application of two metaheuristics to this NP-Hard problem of wide practical application: Variable Neighborhood Descent and Steepest Descent. It is noteworthy that is the first reported application of Variable Neighborhood Descent to the solution of this problem. Computational experiments involved 595 instances from five different set of the literature. The reported results demonstrate that there was no dominance between the proposed methods and that they were able to obtain a good part of the optimal solutions for the instances considered, abtaining a small gap in the other cases, which shows the robustness of the proposed methods.

Dedico este trabalho aos meus pais que sempre estiveram presentes me apoiando nos momentos de dificuldade.

Agradecimentos

A Deus por me dar vida e saúde para seguir em frente.

Aos meus pais pelo incentivo e apoio incondicional.

Ao meu orientador, pelo empenho e dedicação que tornaram este trabalho possível.

A Universidade Federal de Ouro Preto.

A todos que direta ou indiretamente fizeram parte da minha formação.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Algoritmos

Capítulo 1

Introdução

Em contextos industriais, problemas de corte de estoque são problemas de otimização combinatória que consistem em cortar unidades maiores de matéria-prima para produzir unidades menores (ou *peças*) satisfazendo-se algum objetivo, como por exemplo, minimizar as sobras das unidades maiores. Estes são problemas de grande importância prática, sendo relevantes nas indústrias que realizam o processamento de matérias-primas tais como metal, madeira, vidro e papel para a fabricação de bens de consumo. As unidades maiores podem ser chapas, barras, bobinas ou outros objetos, as quais devem ser cortadas em unidades menores. A disposição espacial das peças dentro das unidades maiores para realização do corte define um *padrão de corte*.

Após a definição destes padrões de corte, passa-se à etapa de processamento dos mesmos. A sequência na qual os diferentes padrões são cortados pode afetar diferentes objetivos, tais como a utilização de estoque intermediário, o ritmo de atendimento a diferentes ordens de compra e a homogeneidade das características físicas das peças produzidas. Torna-se desejável então determinar o sequenciamento dos padrões a serem processados de acordo com tais critérios, definindo assim os problemas de sequenciamento de padrões.

Depois de produzida, cada peça é mantida em uma pilha que armazena peças de um mesmo tipo ao redor da máquina que as produziu. Quando a última peça de um tipo for produzida, a pilha referente a este tipo específico de peça pode ser removida da área de produção para outro local. Considera-se que há uma limitação física para o armazenamento de pilhas no ambiente de produção, de modo que não é possível haver pilhas abertas para todos os tipos de peça simultaneamente. Desse modo, deseja-se minimizar o número de pilhas abertas simultaneamente por meio do sequenciamento dos padrões, caracterizando assim o Problema de Minimização de Pilhas Abertas (ou MOSP, de *Minimization of Open Stacks Problem*), conforme descrito por Yuen (1991).

Pode ser necessária a remoção temporária de pilhas ainda não fechadas do ambiente de produção para que novas pilhas sejam abertas e as peças continuem sendo produzidas. A remoção constante de pilhas abertas afeta o custo associado à produção das peças, exigindo a

alocação de mão-de-obra, maquinário e tempo adicionais para o transporte e a administração das pilhas removidas temporariamente, ao mesmo tempo introduzindo riscos à integridade dos produtos fabricados, o que não é desejável, principalmente em se tratando de indústrias onde a manipulação excessiva dos produtos cause danos gerando prejuízo para a indústria. Diante disso, é desejável que as pilhas sejam retiradas do ambiente de produção uma única vez, seja para armazenamento e/ou distribuição.

1.1 Motivação

Com a evolução e a disseminação da tecnologia, os setores industriais buscam cada vez mais tratar questões operacionais através de métodos computacionais, os quais contribuem para a otimização de sistemas de produção.

O MOSP é um problema de grande importância prática na indústria, sendo relevante nas indústrias que realizam o processamento de uma variedade de tipos de matérias-primas para a fabricação de bens de consumo. Trata-se de um problema NP-Difícil (Linhares e Yanasse, 2002), que possui formulação equivalente a diferentes problemas, conforme mostra o trabalho de (Möhring, 1990): Problema de Corte Modificado (*Modified Cutwidth*), Leiaute de Matrizes de Portas (*Gate Matriz Layout*), Dobradura de Arranjos Lógicos Programáveis (*Programmable Logic Array Folding*, ou *PLA Folding*), *Interval Thickness, Node Search Game, Edge Search Game, Narrowness, Split Bandwidth, Graph Pathwidth, Edge Separation e Vertex Separation.*

A criação de novos modelos para solução do problema em questão pode contribuir ainda para melhoria da eficiência da indústria nacional de bens de consumo.

1.2 Objetivos

Elaborar métodos metaheurísticos consistentes e robustos que possam ser utilizadas no contexto do Problema de Minimização de Pilhas Abertas e permitam a obtenção rápida de soluções próximas da solução ótima sem que se perca a vantagem da busca sistemática – inicialmente considerando problemas específicos, mas com uma possibilidade de generalização. São objetivos específicos:

1. Propor um modelo de otimização que contemple apropriadamente as especificidades de diferentes aplicações do problema abordado com o uso dos modelos descritos na literatura;

2. Implementar computacionalmente um método para determinação de soluções de alta qualidade para o problema abordado usando ferramentas de Inteligência Computacional, o que inclui a utilização de métodos metaheurísticos;

3. Avaliar a performance do método implementado considerando dados reais e também com problemas teste publicamente disponíveis;

4. Pesquisar técnicas para melhoria fina de soluções (*polishing*);

5. Publicar trabalhos em periódicos e eventos nacionais e internacionais, os quais contribuem para a promoção do Departamento de Computação da Universidade Federal de Ouro Preto.

1.3 Organização do Trabalho

O trabalho está organizado da seguinte maneira: No Capítulo 2 é apresentada a revisão da literatura para o problema tratado. No capítulo 3, é apresentada a fundamentação teórica necessária para o entendimento do trabalho. O capítulo 4 contém as contribuições deste trabalho, o que inclui os métodos propostos para resolução do MOSP. No capítulo 5 são descritos e discutidos os experimentos computacionais realizados com os métodos propostos. O capítulo 6 apresenta a conclusão do trabalho e aponta direções para os trabalhos futuros.

Capítulo 2

Revisão da Literatura

O Problema de Minimização de Pilhas Abertas é estudado sob esta denominação desde o início da década de 90. A partir de então, foram realizados vários estudos sobre o problema e também determinou-se a equivalência entre este problema e outros da literatura. Este capítulo apresenta um exame dos trabalhos publicados sob a denominação considerada neste trabalho.

O trabalho de Linhares e Yanasse (2002) apresenta diversos resultados teóricos sobre o MOSP. A complexidade do MOSP é provada NP-Difícil devido a equivalência com o *Problema de Largura de Corte Modificado em Grafos (Modified Cutwidth).*

Diferentes heurísticas construtivas gulosas foram propostas em (Yuen, 1991) e posteriormente aprimoradas em (Yuen, 1995). Nos dois trabalhos são propostas seis heurísticas rápidas com implementações simples, baseadas na relação entre pilhas abertas e padrões ainda não sequenciados. Uma outra heurística gulosa também foi proposta por (Faggioli e Bentivoglio, 1998), porém, com uma fase de refinamento adicional, realizada pelo método Busca Tabu. Posteriormente, (Fink e Voß, 1999) compararam a aplicação de Busca Tabu em diferentes versões, *Simulated Annealing* e diversas heurísticas construtivas aplicadas ao MOSP e problemas correlatos. Os resultados obtidos com a aplicação da Busca Tabu combinada com movimentos de escape demonstraram superioridade em relação aos demais métodos comparados. A *Heurística de Nó de Custo Mínimo*, baseada em busca em grafos e proposta por (Becceneri et al., 2004) é considerada a de melhor desempenho atualmente. Outra heurística baseada em busca em grafos, mais rápida e mais robusta que a anterior – porém, menos precisa – é apresentada em (Carvalho e Soma, 2014).

Diferentes formulações *Branch and Bound* foram apresentadas para o MOSP (Yuen e Richardson, 1995; Yanasse, 1997b; Faggioli e Bentivoglio, 1998; Yanasse e Limeira, 2004), porém, sem aplicação prática devido a restrições de tempo de execução. Outras importantes contribuições incluem a identificação de casos especiais solucionáveis em tempo determinístico polinomial (Yanasse, 1996), a modelagem utilizando grafos (Yanasse, 1997a) e diferentes operações de pré-processamento (Yanasse e Senne, 2010).

Devido a sua complexidade e equivalência com vários outros problemas, além da aplica-

ção em manufatura, em 2005 o MOSP foi escolhido como tema do *First Constraint Modelling Challenge* (Smith e Gent, 2005). O modelo vencedor do desafio foi o de programação dinâmica proposto por (de la Banda e Stuckey, 2007). Posteriormente, (Chu e Stuckey, 2009) aprimoraram este modelo pela inclusão de uma nova estratégia de busca e diferentes procedimentos de poda de soluções parciais. Atualmente, este método representa o estado da arte relacionado a métodos exatos aplicados ao MOSP.

Mais recentemente (Gonçalves et al., 2016) propuseram a aplicação de Algoritmos Genéticos de Chaves Aleatórias Viciadas ao MOSP. Os experimentos computacionais abrangentes reportados indicam que este método foi capaz de obter as soluções ótimas para uma grande variedade de instâncias da literatura, tornando-o o estado da arte relacionado a metaheurísticas aplicadas ao MOSP. Os experimentos reportados envolveram 6141 instâncias existentes na literatura. Dentre as instâncias testadas encontram-se as instâncias do *First Constraint Modelling Challenge*[1] (Smith e Gent, 2005): *Harvey*, *Simonis*, *Shaw*, *Miller* e *Wilson*. Foram usadas ainda as instâncias *Faggioli & Bentivoglio* (Faggioli e Bentivoglio, 1998), as instâncias *VLSI* (Hu e Chen, 1990) e as instâncias SCOOP[2]. Estes conjuntos foram selecionados para teste das implementações propostas neste trabalho.

[1] https://ipg.host.cs.st-andrews.ac.uk/challenge/
[2] Disponíveis em: `http://www.scoop-project.net`

Capítulo 3

Fundamentação Teórica

Neste Capítulo, são detalhados o problema tratado e os métodos disponíveis na literatura que serviram como base para os métodos propostos. A fundamentação apresentada servirá de apoio para compreensão dos métodos desenvolvidos neste trabalho e apresentados no Capítulo 4.

3.1 O Problema de Minimização de Pilhas Abertas

O Problema de Minimização de Pilhas Abertas, como descrito por (Yuen, 1995) remonta à um ambiente de produção em que uma única máquina de corte processa um conjunto P de diferentes padrões para produzir um conjunto C de peças com demandas específicas. A produção é dividida em estágios, e é considerado que em cada estágio um lote de um determinado padrão de corte diferente é processado, ou seja, todas as cópias de um padrão de corte específico devem ser cortadas antes que um padrão de corte diferente seja processado.

Durante a produção de um determinado tipo de peça, todas as suas cópias são armazenadas temporariamente em uma pilha mantida ao redor da máquina que as produziu; cada pilha armazena somente um único tipo específico de peça. Quando a primeira peça de um dado tipo tiver de ser produzida, a respectiva pilha é *aberta* e assim permanece até que a última peça do mesmo tipo seja produzida, quando então a pilha é *fechada*, podendo ser removida do local de produção. É necessário determinar uma sequência π de processamento dos elementos de P tal que o número máximo de pilhas simultaneamente abertas é minimizado.

Uma instância MOSP é representada por uma matriz M binária que relaciona os padrões de corte e as peças. A Tabela 3.1 apresenta um exemplo de instância MOSP. 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 valor 1, na linha i e coluna j da matriz (ou seja, $m_{ij} = 1$) indica que o padrão p_j contém a peça i – o valor 0 indica o contrário (ou seja, $m_{ij} = 0$). No referido exemplo, o padrão p_1 é composto pelas peças 1, 2 e 4, o padrão p_2 é composto pelas peças 2, 4, 5 e 6 e assim por diante.

Tabela 3.1: Instância MOSP.

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

Uma solução para o MOSP é uma permutação π das colunas da matriz M, gerando uma matriz permutação Q^{π} que 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. Nesta tabela, uma pilha é associada a cada peça, também representadas na horizontal. Na vertical, são representados cada estágio de produção, nos quais um lote de determinado padrão de corte é processado. As pilhas abertas são contabilizadas a cada estágio, na vertical.

Caso uma peça cuja pilha correspondente estiver aberta não for produzida em um determinado estágio, esta pilha será contabilizada como aberta se ainda houver peças desse tipo a serem processadas. Esta configuração caracteriza a propriedade dos *uns consecutivos* (ou *1s consecutivos*), que indica que uma pilha permanece aberta mesmo que a peça correspondente não seja produzida naquele estágio. Em outras palavras, a pilha não está fechada porque ainda há peças a serem produzidas e, portanto, não pode ser removida do ambiente de produção. Por exemplo, considere a Tabela 3.2(a), no terceiro e quarto estágios (nos quais os padrões p_3 e p_6 são processados) a peça 1, não é produzida, uma vez que os respectivos padrões não possuem essa peça, porém, em decorrência da configuração dos estágios de produção, a pilha referente a peça 1 permanece aberta até o último estágio, em que o padrão p_5 é produzido. Desse modo, em qualquer estágio intermediário entre a abertura de uma pilha e o seu fechamento a pilha referente a peça é considerada sempre aberta. Os 1s consecutivos estão representados com o valor 1 em negrito na Tabela 3.2.

Tabela 3.2: Dois possíveis sequenciamentos para o processamento dos padrões.

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

(a)

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

(b)

A matriz Q^π representando os estágios de produção obtidos a partir da permutação π dos padrões tem seus elementos q^π_{ij} definidos de acordo com a Equação 3.1, em que π representa a permutação dos padrões que define a sequência em que os mesmos serão cortados. Desse modo $\pi[n]$ indica a ordem de processamento do n-ésimo padrão. Observe que a matriz é definida com base na propriedade dos 1s consecutivos, de modo que toda entrada de valor 0 existente na matriz M compreendida entre duas entradas de valor 1 em uma mesma linha é preenchida com o valor 1 na matriz Q^π.

$$q^\pi_{ij} = \begin{cases} 1, & \text{se } \exists x, \exists y \mid \pi[x] \leq j \leq \pi[y] \text{ e } m_{ix} = m_{iy} = 1 \\ 0, & \text{caso contrário} \end{cases} \tag{3.1}$$

Tendo definido a matriz Q^π, a quantidade máxima de pilhas abertas simultaneamente é calculada pela Função 3.2, em que a quantidade máxima de pilhas abertas será dada pela coluna de Q^π (ou estágio de processamento) que possuir a maior quantidade de 1s, sejam consecutivos ou não.

$$Z^\pi_{MOSP}(Q^\pi) = \max_{j \in P} \sum_{i=1}^{|C|} q^\pi_{ij} \tag{3.2}$$

A partir da análise da Função 3.2 é possível definir a função objetivo do Problema de Minimização de Pilhas Abertas conforme mostrado na Função 3.3, que visa minimizar o número máximo de 1s em qualquer coluna de Q^π.

$$\min_{\pi \in \Pi} \; Z^\pi_{MOSP}(M) \tag{3.3}$$

No primeiro sequenciamento, definido na Tabela 3.2(a), temos $\pi = [p_2, p_1, p_3, p_6, p_4, p_5]$, o que indica que no primeiro estágio serão processadas as cópias do padrão p_2, no segundo estágio serão processadas as cópias do padrão p_1 e assim sucessivamente até que no último estágio sejam processadas as cópias do padrão p_5. Neste sequenciamento, o número máximo de pilhas abertas é 6: no terceiro estágio as pilhas referentes à todas as peças estarão abertas simultaneamente. O segundo sequenciamento, na Tabela 3.2(b), apresenta uma solução com um sequenciamento de padrões diferente, $\pi = [p_3, p_4, p_5, p_1, p_2, p_6]$, com um número máximo de pilhas abertas igual a 4, o que ocorre no segundo, terceiro, quarto e quinto estágios da produção.

3.2 Pré-Processamento por Dominância Entre Padrões

Em uma instância MOSP, o pré-processamento por dominância entre padrões consiste na eliminação de redundâncias – dados que podem ser desconsiderados ao se resolver o problema por não alterarem sua estrutura básica (Yanasse e Senne, 2010). O pré-processamento por

dominância remove padrões que possuem em sua composição todas as peças existentes em algum outro padrão do problema. Denomina-se padrão *dominante* aquele que possui em sua composição um subconjunto de peças que engloba todas as peças de um outro padrão. O padrão *dominado* é aquele que está contido em outro padrão.

Antes da aplicação de um método para solução do MOSP, os padrões dominados podem ser removidos da entrada, diminuindo assim o tamanho do problema tratado. Quando a solução final for gerada, os padrões dominados são sequenciados imediatamente após seus respectivos padrões dominantes em π, sem perda de otimalidade, conforme (Yanasse e Senne, 2010). Embora a verificação de dominância entre os padrões seja executada em tempo cúbico, uma vez que no pior caso cada padrão é comparado a todos os outros, esta estratégia é interessante pois o processamento nas próximas etapas do algoritmo é mais pesado, sendo a quantidade de padrões processados diretamente proporcional ao tempo de execução do mesmo. A Tabela 3.3 apresenta a instância da Tabela 3.1 antes (a) e após (b) a 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	0	0	1	1	0
2	1	1	1	0	0	0
3	0	0	1	1	0	0
4	1	1	1	0	1	0
5	0	**1**	0	0	1	**1**
6	0	**1**	0	0	0	**1**

(a)

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

(b)

Em um primeiro momento, analisando Tabela 3.3(a) verifica-se que a composição do padrão p_6, formado pelas peças 5 e 6 (ressaltadas em negrito) é um subconjunto das peças que compõem o padrão p_2, formado pelas peças 2, 4, 5 e 6. Dessa forma, o pré-processamento por dominância entre padrões elimina o padrão p_6, resultando em uma instância MOSP reduzida, conforme a Tabela 3.3(b), sem causar qualquer prejuízo a solução do problema.

3.3 Descida em Vizinhança Variável

Métodos projetados para aprimoramento de soluções de problemas combinatórios frequentemente empregam a utilização de algoritmos de *busca* local: dada uma solução inicial gerada por outro método, um *movimento* consiste em um tipo de operação que altera pontualmente a solução original, transformando-a em uma solução diferente. A nova solução obtida é dita *vizinha* da solução original. Desta forma, a aplicação de um movimento particular define uma estrutura de *vizinhança*, ou seja, um conjunto de soluções que podem ser geradas a partir de uma solução original pela aplicação do referido movimento. Os algoritmos de busca local, ao

explorarem uma estrutura de vizinhança, exploram uma região específica do espaço de busca. A solução obtida pela busca local em uma estrutura de vizinhança (dependendo da qualidade da mesma) pode ser um ótimo local, ou mesmo, um ótimo global.

Hansen e Mladenović (2001) propõem dois métodos metaheurísticos simples (e outras variações) de aplicação geral, baseadas em busca local em vizinhanças variáveis. A primeira delas é a *Descida em Vizinhança Variável* (ou *Variable Neighborhood Descent* – VND). Segundo os autores, o VND se baseia em três princípios:

1. Um ótimo local com relação a uma estrutura de vizinhança não é necessariamente um ótimo local relativo à outra estrutura de vizinhança;
2. Um ótimo global é um ótimo local com relação a todas as estruturas de vizinhança; e
3. Para muitos problemas, ótimos locais com relação a uma ou mais estruturas de vizinhança são relativamente próximos.

Este método consiste em definir um conjunto N_k de k_{max} diferentes estruturas de vizinhança (definidas por movimentos específicos) e determinar o ótimo local em cada uma destas vizinhanças, iteradamente. A cada iteração, uma vizinhança k específica é explorada pela aplicação sucessiva do movimento associado à solução π e, a cada ótimo local π' encontrado, a solução atual π é atualizada. Quando não há melhoria possível, a próxima vizinhança passa a ser explorada e o método se repete até que todas as k_{max} vizinhanças sejam analisadas. O Algoritmo 1 apresenta um pseudocódigo genérico para o VND.

Algoritmo 1: Pseudocódigo do método de Descida em Vizinhança Variável.

```
1  Inicialização: Selecione o conjunto de estruturas de vizinhança N_k = 1, ..., k_max.
2  Determine uma solução inicial π;
3  k ← 1;
4  repita
5  |  Encontre um melhor vizinho π' de π (π' ∈ N_k(π));
6  |  se π' melhor que π então
7  |  |  π ← π';
8  |  fim
9  |  senão
10 |  |  k ← k + 1;
11 |  fim
12 até k = k_max;
```

Alguns dos componentes do VND utilizados neste trabalho, bem como o método de geração de uma solução inicial, foram retirados da literatura. As seções a seguir descrevem brevemente estes componentes e no Capítulo 4 os métodos propostos são descritos em maior detalhe.

3.4 Uma Heurística Baseada em Busca em Grafos

O método heurístico proposto por (Carvalho e Soma, 2014) modela o MOSP por meio de grafos, conforme descrito por (Yanasse, 1997a), denominados grafos MOSP. Nestes grafos, os vértices representam as peças e há uma aresta entre dois vértices se e somente se as peças correspondentes a estes vértices forem cortadas juntas em pelo menos um mesmo padrão de corte. Desta forma, como as peças de um mesmo padrão de corte são adjacentes entre si, as mesmas induzem cliques no grafo MOSP resultante. A Figura 3.1 representa um grafo MOSP formado a partir da união de cliques. Esta estratégia permite que o problema seja resolvido considerando em um primeiro momento as peças que compõem os padrões e posteriormente o sequenciamento dos mesmos.

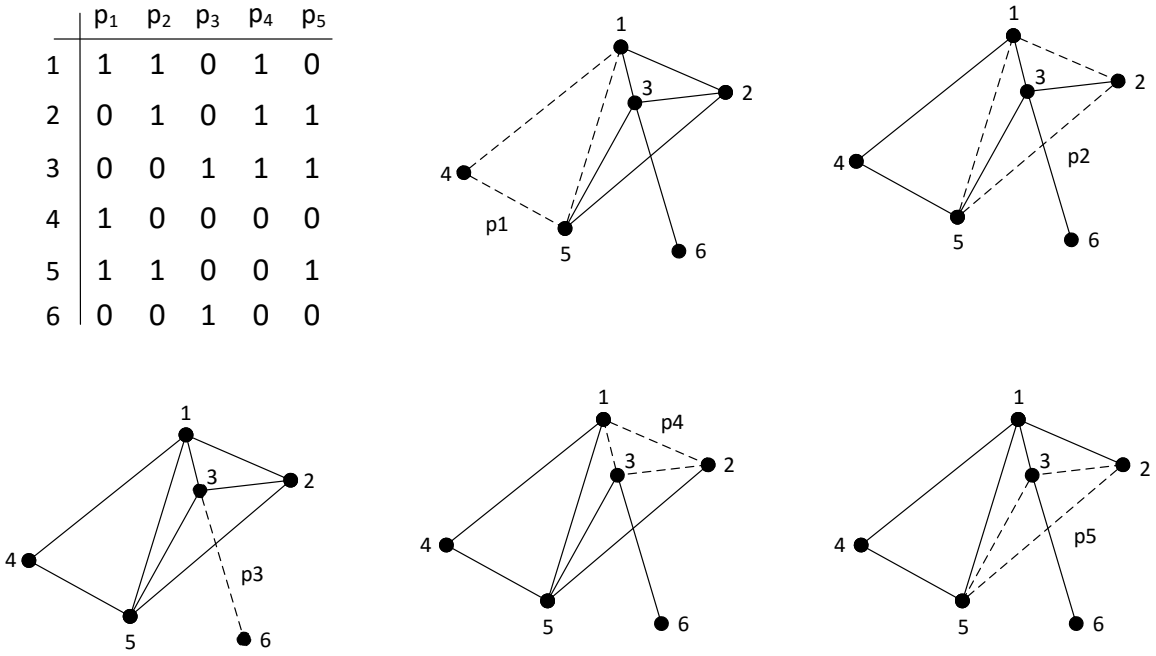

Figura 3.1: Grafo MOSP formado a partir da união de cliques.

Antes de determinar a sequência π de padrões, determina-se a sequência ϕ de peças, de maneira a refletir o relacionamento entre as mesmas e tentar identificar quais pilhas devem estar abertas simultaneamente. Para este fim, aplica-se a conhecida *Busca em Largura* (ou *Breadth-First Search* – BFS) no grafo MOSP. A BFS gera uma sequência de vértices, referentes à ordem de exploração dos mesmos, utilizando-se como critério inicialmente escolher o vértice com menor grau dentre todos, e posteriormente, todos os demais em ordem crescente do grau. Caso o grafo seja desconexo, o processo se repete com o sequenciamento de um outro vértice com o menor grau dentre os vértices que ainda não tenham sido explorados.

É descrito o passo à passo da aplicação da BFS para o grafo apresentado na Figura 3.1. A primeira etapa do algoritmo consiste na seleção de um vértice de partida, sobre o qual

o algoritmo terá início. Neste exemplo, o vértice 6 será o inicial, de acordo com o critério adotado. O vértice 6 é então adicionado à lista ϕ. O próximo passo consiste na análise dos vértices adjacentes ao vértice 6 ainda não presentes na lista ϕ, ou seja, o vértice 3, que então é adicionado à lista ϕ após o vértice 6. Após a análise dos vizinhos do vértice 6, os vizinhos do próximo vértice da lista (vértice 3) serão analisados. Os vizinhos são os vértices 2, 1 e 5, inseridos nesta ordem (ou seja, grau crescente) ao final de ϕ. Neste ponto, temos $\phi = [6, 3, 2, 1, 5]$. O vértice 4 é o último remanescente e, na análise da vizinhança do vértice 1, é sequenciado ao final de ϕ, resultando em $\phi = [6, 3, 2, 1, 5, 4]$. A BFS continua, entretanto, todos os vértices já foram sequenciados.

O sequenciamento dos padrões é obtido por meio do método elaborado por Becceneri (1999). Este método simula a abertura sucessiva das pilhas referentes às peças presentes em ϕ e verifica a composição dos padrões do problema, de modo que, caso a composição de um padrão seja um subconjunto das pilhas abertas em determinado instante, este padrão é imediatamente inserido ao final da solução π. O Algoritmo 2 apresenta o pseudocódigo para o sequenciamento dos padrões. A entrada consiste na lista ϕ de peças obtida a partir do sequenciamento das peças. A função $c(p_i)$ retorna quais peças compõem o padrão p_i, e, a cada instante, o conjunto A de peças, representa as pilhas que vão sendo abertas a cada iteração do algoritmo.

Algoritmo 2: Pseudocódigo do sequenciamento de padrões (Becceneri, 1999).

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

Como exemplo, considere o grafo MOSP representado na Figura 3.1, a partir da lista ϕ de peças definida no processo de sequenciamento das peças descrito anteriormente, em um primeiro momento é aberta a pilha referente a peça 6, como nenhum padrão possui em sua composição somente a peça 6, a próxima pilha, referente a peça 3 é aberta. Neste momento, a composição do padrão p_3 é exatamente o conjunto das pilhas abertas e, portanto, p_6 é inserido em π. A próxima pilha a ser aberta é referente à peça 2 e não há padrões que atendam ao critério de sequenciamento. A pilha que representa a peça 1 é então aberta, neste momento o padrão p_4 é então adicionado à solução π. A próxima pilha, referente a peça 5 é aberta e neste momento os padrões p_2 e p_5 sequenciados em ordem lexicográfica crescente, dado que

houve um empate. A última pilha a ser aberta referente a peça 4 faz com que o padrão p_1 restante seja sequenciado e a solução final seja $\pi = [p_3, p_4, p_2, p_5, p_1]$.

3.5 Método *k-swap*

O *k-swap* é considerado uma generalização do método original *swap* – em que dois elementos são trocados de posição – projetado para o mesmo fim. Este método também é aplicado com frequência como método de busca local ou método de perturbação de soluções, incorporado em metaheurísticas.

O princípio do *k-swap* consiste na realização de trocas de posição entre k elementos pertencentes à solução de um problema combinatório, gerando soluções que diferem em exatamente k elementos da configuração inicial. Por exemplo, considere um problema em que são dispostos 4 elementos. A Figura 3.2 ilustra as transições decorrentes da aplicação do *swap* entre todos os pares de elementos dada uma configuração inicial $[1, 2, 3, 4]$. A cada instante, dois elementos trocam de posição – as setas indicam os elementos envolvidos na troca de posições. A aplicação do método produz uma estrutura de vizinhança de seis elementos, referentes as combinações dos elementos tomados 2 a 2.

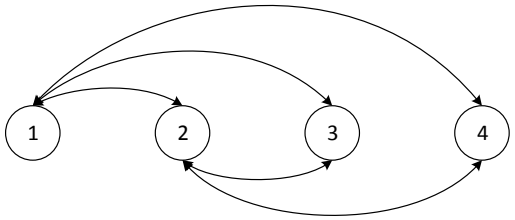

Figura 3.2: Exemplo de transições no método *k-swap*, no caso em que k=2 (*swap* clássico).

3.6 Descida Mais Rápida

O método de *Descida Mais Rápida* trata-se de um método de busca local baseado no conceito de melhoria iterativa de uma solução inicial. Um método de busca local é aplicado a uma solução iterativamente, atualizando-a sempre que uma melhoria for atingida. Quando um ótimo local for atingido (ou seja, quando não houver melhoria possível), o método se encerra. Este é um método de minimização, frequentemente utilizado para encontrar mínimos locais.

O Algoritmo 3 apresenta o pseudo-código genérico para o método de descida mais rápida em que $N(\pi)$ denota a vizinhança da solução corrente sob análise.

Algoritmo 3: Pseudocódigo do método Descida Mais Rápida.

```
1 Inicialização: Determine uma solução inicial π.
2 melhoria ← falso;
3 repita
4 |   Determine um vizinho π' de π (π' ∈ N(π));
5 |   se π' melhor que π então
6 |   |   π ← π';
7 |   |   melhoria ← verdadeiro;
8 |   fim
9 até melhoria = falso;
```

No Algoritmo 3, em um primeiro momento é encontrada uma solução inicial. Neste momento a solução corrente passa a ser a solução inicial e a análise sobre sua vizinhança tem início. Caso alguma solução vizinha seja melhor que a solução corrente o processo se repete para a nova solução corrente enquanto houver melhoria. Desse modo, o algoritmo só termina quando após a análise de toda uma vizinhança não for encontrada sequer uma solução melhor que a solução corrente.

Capítulo 4

Desenvolvimento

Neste capítulo, são apresentados os métodos propostos, em detalhes. A proposta deste trabalho é a aplicação de implementações dos métodos Busca Local Iterada e Descida Mais Rápida para resolução do MOSP. Salvo melhor juízo, esta é a primeira aplicação reportada de Descida em Vizinhança Variável ao MOSP. Ambas as implementações propostas são detalhadas nas seções a seguir.

4.1 Um Novo Método de Busca Local

O princípio do novo método de busca local, dada uma solução inicial π se dá como descrito a seguir. A solução π é avaliada quanto ao número de pilhas abertas e identifica-se o conjunto G de pilhas relacionadas com o número máximo de pilhas abertas, ou seja, quais pilhas representam o *gargalo* do problema. Posteriormente, identifica-se quais padrões possuem em sua composição interseção com os elementos de G. Estes padrões são removidos da solução π e inseridos novamente na melhor posição possível. A ordem de escolha dos padrões para reinserção é determinada pela similaridade da composição dos mesmos em relação aos elementos de G, de maneira decrescente.

O Algoritmo 4 apresenta o pseudocódigo do algoritmo de busca local utilizado. A entrada do algoritmo consiste de uma solução vizinha π' que será modificada no processo. O primeiro passo é encontrar o gargalo de π'. Identificado o estágio em que esse gargalo ocorre, é realizada uma verificação dos padrões de modo a determinar aqueles que possuem em sua composição alguma peça presente no gargalo da solução. A lista L com os i padrões com alguma semelhança com o gargalo é ordenada de forma decrescente da similaridade de peças em relação ao mesmo. Cada padrão x é então reinserido na solução vizinha π' de modo a gerar a menor quantidade de 1s consecutivos possível. Caso ocorra melhoria no valor da nova solução, o procedimento é aplicado novamente na solução atual. A busca local é aplicada enquanto houver melhoria na solução corrente, ou seja, ao atingir um ótimo local, o procedimento é encerrado.

Algoritmo 4: Pseudocódigo da Busca Local

Entrada: Solução π

repita
  Encontre o gargalo de π';
  Liste os padrões que tem relação com o gargalo;
  Ordene os padrões de acordo com a maior semelhança com o gargalo;
  $melhoria \leftarrow falso$;
  para cada $x \in L$ **faça**
    Reinsira x em π';
    se π' *foi melhorada* **então**
      $melhoria \leftarrow verdadeiro$;
    fim
  fim
até $melhoria = falso$;

Por exemplo, na Tabela 3.2(a) o número máximo de pilhas abertas ocorre no terceiro estágio, quando o padrão p_3 é processado. O conjunto G é formado pela peças 1, 2, 3, 4, 5 e 6, que representam o gargalo do problema, e, consequentemente, os padrões p_2, p_1, p_5, p_4 e p_6 seriam removidos e inseridos novamente na solução, nesta ordem.

4.2 Descida em Vizinhança Variável

A implementação do método VND proposta neste trabalho utiliza a heurística construtiva descrita na Seção 3.4 para geração de uma solução inicial. A geração de uma solução inicial e as estruturas de vizinhança, bem como um método adicional de busca local são descritos nas seções a seguir.

O segundo método proposto neste trabalho, denominado VND Aprimorado, apresenta uma pequena modificação em relação ao VND original. Esta modificação consiste em, ao invés de se escolher o melhor vizinho π' e compará-lo à solução corrente π, a aplicar uma segunda busca local, apresentada na Seção 4.1, a cada vizinho π' de π e então selecionar o melhor vizinho π', que será comparado à solução corrente π.

A eficiência do VND reside na definição de estruturas de vizinhança que explorem diferentes regiões promissoras do espaço de soluções, em busca de soluções ótimas ou subótimas globais. Na implementação de VND proposta, o algoritmo *k-swap*, descrito na Seção 3.5, foi utilizado para gerar quatro diferentes estruturas de vizinhança.

Visando reduzir o espaço de busca e o tempo execução do VND, empregou-se uma estratégia de agrupamento de padrões consecutivos presentes em uma solução. Os grupos formados serão compostos por no máximo n padrões distintos. Por exemplo, considere uma solução

inicial $\pi = [1, 3, 5, 4, 2]$. Para $n \leq 2$, os grupos seriam $[1, 3]$, $[5, 4]$ e $[2]$, ou seja, o *k-swap* passaria a ter 3 elementos e não 5, e os grupos seriam trocados entre si ao invés de cada um dos padrões. Em outras palavras, é realizado um *k-swap* sobre grupos de padrões, e não sobre padrões individuais. Após experimentos preliminares, optou-se por utilizar $n \leq 2$ e selecionar os pares para troca em ordem aleatória, embora todos os pares sejam testados.

Para gerar diferentes estruturas de vizinhança utilizando o algoritmo *k-swap*, variou-se o valor de k entre 2 e 5, nesta ordem. À medida em que o valor de k aumenta, maior é a exploração do espaço de busca. Este método foi escolhido para gerar as estruturas de vizinhança porque o método para geração da solução inicial fornece soluções de boa qualidade, das quais optou-se por não alterar fortemente a estrutura. Conforme reportado na Seção 5, bons resultados foram obtidos com este método.

O Algoritmo 5 apresenta o pseudocódigo referente ao VND Aprimorado. O funcionamento do algoritmo é semelhante ao VND original. A proposta de alteração encontra-se nas linhas 5, 8, 9, 12 e 13 do referido algoritmo, onde as vizinhanças da solução corrente a serem analisadas passam a ser divididas em grupos de Δ padrões, sendo este um parâmetro passado para o procedimento. Desse modo, cada solução vizinha π' é formada a partir de trocas do tipo *k-swap* entre os grupos de padrões que compõem a solução corrente, e, para cada solução vizinha π' é realizada uma nova busca local. Na primeira solução π'' melhor do que a solução π atualiza-se a solução corrente, agrupa-se os padrões da nova solução e reinicia-se a busca na mesma vizinhança a partir desta nova solução corrente. O VND é executado enquanto houver melhoria na solução corrente, ou seja, ao atingir um ótimo local em todas as vizinhanças, o procedimento é encerrado.

Algoritmo 5: Pseudocódigo do VND Aprimorado

Entrada: Tamanho dos grupos Δ.

1 **Inicialização:** Selecione o conjunto de estruturas de vizinhança $N_{k-swap} = 1, ..., k_{max}$. Determine uma solução inicial π.
2 **repita**
3 $k \leftarrow 1$;
4 *melhoria* $\leftarrow$ *falso;*
5 Agrupe os elementos de π em conjuntos de tamanho Δ;
6 **repita**
7 **para cada** *vizinho* π' *de* π ($\pi' \in N_{k-swap}(\pi)$) **faça**
8 $\pi'' \leftarrow BuscaLocal(\pi')$;
9 **se** π'' *melhor que* π **então**
10 $\pi \leftarrow \pi''$;
11 *melhoria* $\leftarrow$ *verdadeiro;*
12 Agrupe os elementos de π em conjuntos de tamanho Δ;
13 retorne à linha 6;
14 **fim**
15 **fim**
16 **se** $melhoria = falso$ **então**
17 $k \leftarrow k + 1$;
18 **fim**
19 **até** $k = k_{max}$;
20 **até** $melhoria = falso$;

Mesmo após a adoção da estratégia de agrupamento na geração das vizinhanças utilizando o *k-swap*, a limitação do método proposto se encontra no tempo de execução, que devido a aplicação da busca local para cada vizinho π', demanda um alto tempo de execução quando aplicado em instâncias grandes do MOSP. Este fator levou a proposta de uma segunda implementação, baseada no método de Descida Mais Rápida apresentado na Seção 3.6.

4.3 Descida Mais Rápida

O terceiro método proposto neste trabalho é do tipo *Descida Mais Rápida* (ou *Steepest Descent* – SD), descrito na Seção 3.6. Este método foi proposto como uma simplificação do VND descrito anteriormente, em que a vizinhança explorada é induzida somente pelo método de *swap* clássico – ou *2-swap* – em contraposição ao *k-swap* utilizado no VND proposto. Outra diferença se dá pela não utilização da estratégia de agrupamentos de padrões como proposta para o VND, tendo esta sido substituída por uma estratégia de *janela deslizante*.

Dada uma solução π', uma *janela* é uma subsequência de n padrões consecutivos ($1 \leq n \leq |P|$). Inicialmente, esta subsequência consiste nos primeiros n padrões em π' ($\pi'[1..n]$). Esta janela é então deslizada um estágio para a direita ($\pi'[2..n+1]$) e assim sucessivamente até que todos os padrões em π' tenham sido contemplados. Por exemplo, considerando a solução $\pi' = [1, 3, 5, 4, 2]$ e n=2, as janelas seriam [1, 3], [3, 5], [5, 4] e [4, 2]. O método de *2-swap* é aplicado sobre duas janelas diferentes, escolhidas aleatoriamente, como busca local. Após experimentos preliminares, optou-se por utilizar $n = 2$.

A cada solução gerada pelo método de *2-swap*, aplicou-se o método de busca local descrito na Seção 4.1, no intuito de aprimorá-las. Caso a solução resultante possua melhor valor do que a solução corrente, a mesma é atualizada. O método de descida mais rápida é executando enquanto houver melhoria nas soluções obtidas.

O Algoritmo 6 apresenta o pseudocódigo para o método de descida rápida proposto, em que $N_{2\text{-}swap}$ denota a vizinhança induzida pelo método de *2-swap* sobre todas as janelas de padrões possíveis, em ordem aleatória. O parâmetro Δ indica a quantidade de padrões em cada janela deslizante. A partir das janelas de padrões, é realizada a geração das vizinhos da solução corrente através do método de *2-swap*, ou seja, janelas de padrões são trocadas de lugar a fim de gerar novos vizinhos.

Algoritmo 6: Pseudocódigo do método Descida Mais Rápida.

```
Entrada: Tamanho das janelas Δ.
Inicialização: Determine uma solução inicial π.
melhoria ← falso;
Gerar todas as janelas de tamanho Δ com os elementos de π;
repita
    para cada vizinho π' de π (π' ∈ N_2-swap(π)) faça
        π'' ← BuscaLocal(π');
        se π'' melhor que π então
            π ← π'';
            melhoria ← verdadeiro;
            Gerar todas as janelas de tamanho Δ com os elementos de π;
        fim
    fim
até melhoria = falso;
```

Embora esta estratégia de janela deslizante se assemelhe ao agrupamento de pares de padrões consecutivos utilizado pelo VND, tratam-se de estratégias diferentes. A utilização de janelas deslizantes permite que um mesmo padrão figure em diferentes janelas, aumentando o espaço de busca. Ainda assim, isto permite que a estrutura da solução inicial seja modificada apenas pontualmente. Conforme descrito anteriormente, a busca em grafos utilizada produz

boas soluções iniciais e optou-se por não alterá-las significativamente.

Capítulo 5

Experimentos

Os experimentos computacionais foram realizados em um computador com processador *Intel i5 Quad Core* de 3.2 GHz com 16 GB RAM sob o sistema operacional Ubuntu 12.4.1. Os códigos dos métodos propostos foram escritos em C++, compilados com g++ 4.4.1 e a opção de otimização -O3. Comparam-se a quantidade de pilhas abertas obtidas pelos métodos propostos e o tempo de execução médio. Dada a utilização de componentes de aleatoriedade pelos algoritmos, foram realizadas 20 execuções independentes para cada uma das instâncias, considerando-se os resultados médios.

Cinco conjuntos de instâncias foram utilizados, divididos em diferentes seções. Nas tabelas seguintes, a coluna *Instância* refere-se ao nome das instâncias, $|P|$ indica a quantidade de padrões, $|C|$ representa a quantidade de peças e OPT indica o valor da solução ótima. Os valores de solução obtidos são apresentados na coluna *VND* e *SD*, respectivamente para os métodos Descida em Vizinhança Variável e Descida Mais Rápida. Os tempos médios de execução (expressos em segundos) são apresentados na coluna T e o valor da melhor solução obtida é apresentado na coluna S^*. Nos casos em que as soluções ótimas foram atingidas, os valores estão destacados em negrito. Por fim, a coluna σ indica o desvio padrão entre as soluções encontradas para a instância após as 20 execuções individuais.

Onde mencionado, o *gap* (ou distância percentual) é calculado como.

$$gap = 100 \times \frac{S^* - OPT}{OPT} \tag{5.1}$$

5.1 Instâncias Reais MOSP

O primeiro conjunto de instâncias, fornecido pelo *SCOOP Consortium* (disponível em `http://www.scoop-project.net`), contém 187 instâncias MOSP reais de duas empresas moveleiras européias. Entretanto, a maioria destas instâncias são muito pequenas (por exemplo, 2 linhas e colunas), possuindo soluções triviais. Deste conjunto, 24 instâncias com dimensões significativas, variando de 10 linhas e 14 colunas a 49 linhas e 134 colunas, foram seleciona-

das para serem usadas nos experimentos. A Tabela 5.1 apresenta os valores para tempo de execução médio e melhor solução encontrada para cada instância utilizada nos testes.

Tabela 5.1: Instâncias SCOOP disponíveis em: `http://www.scoop-project.net`. O valor em negrito indica soluções ótimas.

Instância	$\|P\|$	$\|C\|$	OPT	VND			SD		
				T	S^*	σ	T	S^*	σ
A_FA+AA-1	37	107	12	1,02	**12**	0,32	0,28	**12**	0,79
A_FA+AA-11	28	99	11	0,11	**11**	0,45	0,08	**11**	0,65
A_FA+AA-12	20	75	9	0,01	**9**	0,45	0,02	**9**	0,00
A_FA+AA-13	37	134	17	1,69	18	0,50	0,40	18	0,60
A_FA+AA-15	18	68	9	0,01	**9**	0,23	0,01	10	0,00
A_FA+AA-2	19	75	11	0,00	**11**	0,50	0,01	**11**	0,37
A_FA+AA-6	21	79	13	0,02	**13**	0,51	0,04	**13**	0,47
A_FA+AA-8	28	82	11	0,12	12	0,48	0,08	12	0,51
A_AP-9.d-3	16	20	6	0,00	**6**	0,23	0,00	**6**	0,00
A_AP-9.d-10	13	20	6	0,00	**6**	0,51	0,00	**6**	0,47
A_AP-9.d-11	21	27	6	0,00	**6**	0,00	0,00	**6**	0,00
A_AP-9.d-6	20	31	5	0,00	**5**	0,00	0,00	**5**	0,00
B_12F18-11	15	21	6	0,00	**6**	0,51	0,00	**6**	0,00
B_12M18-12	22	31	6	0,00	**6**	0,45	0,00	**6**	0,50
B_18AB1-32	11	15	6	0,00	7	0,00	0,00	**6**	0,00
B_18CR1-33	18	20	4	0,00	**4**	0,00	0,00	**4**	0,00
B_22X18-50	11	14	10	0,00	**10**	0,00	0,00	**10**	0,00
B_23B25-52	21	29	5	0,00	**5**	0,00	0,00	**5**	0,00
B_39Q18-82	10	14	5	0,00	6	0,00	0,00	**5**	0,22
B_42F22-93	10	18	5	0,00	**5**	0,00	0,00	**5**	0,00
B_CARLET-137	12	14	5	0,00	6	0,00	0,00	**5**	0,41
B_CUC28A-138	26	37	6	0,00	**6**	0,51	0,00	**6**	0,50
B_GTM18A-139	20	24	5	0,00	**5**	0,23	0,00	**5**	0,00
B_REVAL-145	49	60	7	0,97	**7**	0,45	0,16	**7**	0,31
Média			7,75	0,16	7,96	0,26	0,05	7,88	0,24

Ambos os métodos obtiveram boas soluções para este conjunto de instâncias em muito baixo tempo de execução, porém, o método de Descida Mais Rápida apresentou melhores indicadores. Os *gaps* foram 2,69% e 1,61% respectivamente para o VND e o método de Descida Mais Rápida. Além disto, o primeiro método foi capaz de obter 79,17% das soluções ótimas, ao passo que o segundo obteve 87,5%, ou seja, apenas para as instâncias *A_FA+AA* 8, 13 e 15 a melhor solução não foi atingida pelo método de Descida Mais Rápida. O VND não foi capaz de obter as soluções ótimas para cinco instâncias. Nos casos em que a solução ótima não foi obtida, uma pilha a mais foi aberta por instância pelos dois métodos. Ambos os métodos geraram soluções cujos valores resultaram em baixo desvio padrão, demonstrando robustez.

5.2 Instâncias Reais VLSI

O segundo conjunto contém 25 instâncias reais para o problema de Leiaute de Matrizes de Portas (um problema de formulação equivalente ao MOSP, conforme mencionado na Seção 1), oriundas de empresas asiáticas, introduzidas por (Hu e Chen, 1990). Os resultados são apresentados na Tabela 5.2.

Tabela 5.2: Instâncias reais VLSI. Valores em negrito indicam soluções ótimas.

Instância	$\|P\|$	$\|C\|$	OPT	*VND*			*SD*		
				T	S^*	σ	T	S^*	σ
v1	8	6	3	0,00	4	0,00	0,00	4	0,00
v4000	17	10	5	0,00	**5**	0,00	0,00	**5**	0,00
v4050	16	13	5	0,00	**5**	0,49	0,00	**5**	0,00
v4090	27	23	10	0,00	**10**	0,00	0,00	**10**	0,00
v4470	47	37	9	1,32	**9**	0,22	0,00	10	0,59
vc1	25	15	9	0,00	**9**	0,47	0,00	**9**	0,00
vw1	7	5	4	0,00	**4**	0,00	0,00	**4**	0,00
vw2	8	8	5	0,00	**5**	0,00	0,00	**5**	0,00
w1	21	18	4	0,00	**4**	0,44	0,00	**4**	0,37
w2	33	48	14	0,00	**14**	0,22	0,02	**14**	0,52
w3	70	84	18	20,87	**18**	0,57	2,40	**18**	0,46
w4	141	202	27	4070,19	**27**	0,52	35,50	28	2,19
wan	7	9	6	0,00	**6**	0,00	0,00	**6**	0,00
wli	10	11	4	0,00	**4**	0,00	0,00	**4**	0,00
wsn	25	17	8	0,00	**8**	0,00	0,00	**8**	0,00
x0	48	40	11	1,11	**11**	0,37	0,35	**11**	0,49
x1	10	5	5	0,00	**5**	0,00	0,00	**5**	0,00
x2	15	6	6	0,00	**6**	0,00	0,00	**6**	0,00
x3	21	7	7	0,00	**7**	0,00	0,01	**7**	0,00
x4	8	12	2	0,00	**2**	0,00	0,00	**2**	0,00
x5	16	24	2	0,00	**2**	0,00	0,00	**2**	0,00
x6	32	49	2	0,00	**2**	0,00	0,00	**2**	0,00
x7	8	7	4	0,00	**4**	0,00	0,00	**4**	0,00
x8	16	11	4	0,00	**4**	0,00	0,00	**4**	0,00
x9	32	19	4	0,06	**4**	0,00	0,03	**4**	0,00
Média			7,12	163,74	7,16	0,13	1,53	7,24	0,18

Neste segundo conjunto de instâncias, o VND apresentou o melhor desempenho, obtendo 96% das soluções ótimas e um *gap* de apenas 0,56%. O método de Descida Mais Rápida obteve 88% das soluções ótimas e *gap* de 1,69%. Nota-se em ambos os métodos um crescimento no tempo de execução, especificamente em instâncias de maiores dimensões, como $w3$ e $w4$ (ultrapassando uma hora). Este fator levou à proposta de implementação do método de Descida Mais Rápida. Novamente, o desvio padrão dos valores de solução obtidos em 20 execuções independentes para cada instância é baixo em ambos os métodos.

5.3 Instâncias MOSP (Faggioli e Bentivoglio, 1998)

O terceiro conjunto, proposto por (Faggioli e Bentivoglio, 1998) contém 300 instâncias MOSP artificiais. Os problemas foram agrupados em subconjuntos de 10 instâncias de acordo com o número de padrões e peças. A Tabela 5.3 apresenta os resultados obtidos para todas as instâncias de cada grupo. A coluna σ apresenta os desvios padrão médios para cada grupo de instâncias obtidos a partir das 20 execuções.

Tabela 5.3: Instâncias (Faggioli e Bentivoglio, 1998). Valores em negrito indicam soluções ótimas.

$\|P\|$	$\|C\|$	OPT	*VND*			*SD*		
			T	S^*	σ	T	S^*	σ
	10	55	0,00	58	0,00	0,00	**55**	0,00
10	20	62	0,00	65	0,36	0,00	**62**	0,29
10	30	61	0,00	66	0,41	0,00	63	0,18
10	40	77	0,00	78	0,24	0,00	78	0,06
10	50	82	0,00	**82**	0,25	0,00	**82**	0,18
15	10	66	0,00	**66**	0,17	0,00	**66**	0,03
15	20	72	0,00	74	0,41	0,00	74	0,26
15	30	73	0,00	78	0,36	0,00	76	0,34
15	40	72	0,00	73	0,33	0,01	**72**	0,30
15	50	74	0,00	**74**	0,32	0,00	75	0,26
20	10	75	0,00	**75**	0,16	0,00	**75**	0,00
20	20	85	0,00	**85**	0,26	0,01	**85**	0,15
20	30	88	0,01	90	0,41	0,02	90	0,33
20	40	85	0,01	88	0,43	0,02	86	0,43
20	50	79	0,01	82	0,35	0,02	81	0,35
25	10	80	0,00	**80**	0,00	0,00	**80**	0,00
25	20	98	0,02	**98**	0,32	0,02	**98**	0,39
25	30	105	0,04	108	0,42	0,04	106	0,44
25	40	103	0,04	106	0,33	0,05	104	0,45
25	50	100	0,05	102	0,38	0,05	102	0,42
30	10	78	0,00	**78**	0,00	0,00	**78**	0,00
30	20	111	0,05	112	0,14	0,05	112	0,26
30	30	122	0,14	**122**	0,33	0,08	123	0,43
30	40	121	0,19	**121**	0,38	0,10	**121**	0,48
30	50	112	0,19	117	0,28	0,11	114	0,47
40	10	84	0,00	**84**	0,00	0,01	**84**	0,00
40	20	130	0,31	**130**	0,15	0,12	132	0,18
40	30	145	0,88	147	0,21	0,24	147	0,39
40	40	149	1,23	155	0,14	0,32	151	0,50
40	50	146	1,40	151	0,35	0,35	153	0,47
MÉDIA		93,00	0,15	94,83	0,26	0,05	94,17	0,27

Neste conjunto de instâncias com dimensões médias, novamente o método de Descida Mais Rápida apresenta leve superioridade sobre o VND, obtendo 88,67% das soluções ótimas e *gap* de 1,25%. O tempo de execução se manteve abaixo de 0,4 segundo. O VND obteve 82,00% das soluções ótimas e *gap* de 1,97%, e o tempo de execução se manteve abaixo de 1,5 segundo. Como esperado, os tempos de execução do VND crescem mais rapidamente do que os do método de Descida Mais Rápida. Após a análise dos desvios padrão médios em cada conjunto de instâncias, pode-se dizer que o o valor obtido para cada instância é baixo em ambos os métodos, demonstrando robustez.

5.4 Instâncias do *First Constraint Modeling Challenge* (Smith e Gent, 2005)

O quarto conjunto contém 46 instâncias MOSP propostas para o *First Constraint Modeling Challenge* (Smith e Gent, 2005). Este subconjunto foi selecionado levando em conta as dimensões das instâncias e a ausência de estruturas especiais que podem facilitar a solução, tal como analisado por (Yanasse e Senne, 2010). Os resultados são apresentados na Tabela 5.4, que possui uma coluna adicional n que identifica a quantidade de instâncias em cada grupo. A coluna σ apresenta os desvios padrão médios para cada grupo de instâncias obtidos a partir das 20 execuções.

Tabela 5.4: Resultados para as instâncias selecionadas do *First Constraint Modeling Challenge* (Smith e Gent, 2005). Valores em negrito indicam soluções ótimas.

Instância	$\|P\|$	$\|C\|$	n	OPT	*VND*			*SD*		
					T	S^*	σ	T	S^*	σ
nwrsSmaller	20	10	2	7	0,00	**7**	0,11	0,00	**7**	0,00
nwrsSmaller	25	15	2	14	0,00	**14**	0,25	0,00	**14**	0,22
nrwsLarger	30	20	2	24	0,01	**24**	0,00	0,03	**24**	0,00
nrwsLarger	60	25	2	26	0,57	**26**	0,24	0,19	**26**	0,26
GP1	50	50	4	155	6,79	157	0,09	1,46	156	0,15
GP2	100	100	4	305	4444,24	311	0,24	75,20	308	0,15
Shaw	20	20	25	342	0,01	**342**	0,21	0,01	**342**	0,17
Miller	40	20	1	13	0,67	**13**	0,37	0,19	**13**	0,00
SP4-1	25	25	1	9	0,00	**9**	0,00	0,01	**9**	0,22
SP4-2	50	50	1	19	2,54	20	0,00	0,52	20	0,50
SP4-3	75	75	1	34	177,93	**34**	0,00	5,89	35	0,59
SP4-4	100	100	1	53	1671,34	**53**	0,22	26,31	54	1,05
MÉDIA				21,76	427,35	21,96	0,18	7,40	21,91	0,19

Os métodos VND e Descida Mais Rápida apresentam valores semelhantes para *gap* (0,90% e 0,70%, respectivamente) e percentual de soluções ótimas obtidas (86,96% e 84,78%, respectivamente). O *gap* baixo indica a robustez dos métodos em relação a este conjunto, e, analisado

em conjunto com a taxa de resultados ótimos obtidos, indica que o erro dos métodos em relação às soluções ótimas foi baixo. O tempo de execução do método de Descida Mais Rápida manteve-se abaixo de 75 segundos, aumentando consideravelmente para instâncias com 100 padrões e peças. Para estas mesmas instâncias, o tempo de execução do VND ultrapassou uma hora. Exceto para a instância de maior dimensão, novamente os desvios padrão indicam que os métodos propostos são robustos.

5.5 Instâncias MOSP (Chu e Stuckey, 2009)

O quinto conjunto consiste em 200 instâncias MOSP de maiores dimensões, geradas aleatoriamente por (Chu e Stuckey, 2009). A Tabela 5.5 apresenta os resultados. Novamente, os problemas foram agrupados, em subconjuntos de 25 instâncias de acordo com o número de padrões e peças. Neste conjunto, devido a dimensão das instâncias, o VND foi executado com parâmetro $K\_MAX = 4$. A coluna σ apresenta os desvios padrão médios para cada grupo de instâncias obtidos a partir das 20 execuções.

Tabela 5.5: Instâncias *Random* (Chu e Stuckey, 2009). O valor em negrito indica solução ótima.

$\|P\|$	$\|C\|$	OPT	VND			SD		
			T	S^*	σ	T	S^*	σ
100	100	1455	339,66	1480	0,35	26,60	1484	0,83
50	100	1027	5,69	1044	0,33	2,03	1036	0,79
125	125	1755	1563,22	1783	0,37	76,78	1799	1,02
30	30	490	0,07	491	0,23	0,08	492	0,24
40	40	637	0,60	639	0,31	0,35	641	0,39
100	50	961	82,99	966	0,19	11,06	976	0,32
50	50	785	2,82	792	0,32	1,08	794	0,46
75	75	1121	50,42	1135	0,30	6,89	1138	0,62
MÉDIA		1028,88	255,68	1041,25	0,30	15,61	1045,00	0,58

Neste último conjunto de instâncias, os métodos VND e Descida Mais Rápida apresentam os piores valores para o percentual de soluções ótimas obtidas: 58,50% e 53,50% respectivamente. Entretanto, os valores de *gap* continuam baixos: 1,20% e 1,57% respectivamente. Estes dados demonstram que ambos os métodos continuam com soluções próximas do ótimo, embora as soluções ótimas sejam obtidas somente em aproximadamente metade dos casos. Ao contrário do esperado, os tempos de execução não cresceram rapidamente como no segundo e quarto conjuntos de instâncias, mesmo quando foram tratados problemas com 100 e 125 peças e padrões, algumas das maiores dimensões consideradas nos experimentos. Nestes casos, o tempo de execução do VND esteve entre 5,6 minutos e aproximadamente 26 minutos, ao passo que o tempo de execução do método de Descida Rápida esteve entre 26 segundos e aproximadamente 1,2 minuto. Mais uma vez, a variação entre os valores de solução é baixa.

Este é um indício que os métodos mantêm o erro em relação às soluções ótimas sob controle, demonstrando robustez.

Capítulo 6

Conclusões

O Problema de Minimização de Pilhas Abertas (ou MOSP) é um problema NP-Difícil de sequenciamento de padrões de corte com ampla aplicação industrial que visa otimizar a utilização de estoque intermediário e manipulação de produtos fabricados dentro do ambiente de fabricação.

Este trabalho propõe dois métodos para solução do MOSP. O primeiro consiste na metaheurística Descida em Vizinhança Variável, cujas estruturas de vizinhança foram induzidas pelo método *k-swap* em grupos de padrões; o segundo consiste em um método de descida rápida, cuja busca local é realizada pelo método de *2-swap* clássico em conjunto com uma estratégia de janela deslizante. Ambos os métodos foram executados sobre uma solução inicial gerada por uma adaptação de uma heurística da literatura. Salvo melhor juízo, esta é a primeira aplicação reportada de VND para solução do MOSP.

Os experimentos computacionais envolveram aproximadamente 600 instâncias de cinco diferentes conjuntos da literatura, incluindo instâncias reais e artificiais. Os resultados obtidos foram comparados com as soluções ótimas. De acordo com os dados reportados, não houve método dominante, havendo alternância de melhor desempenho nos três primeiros conjuntos de instâncias e não havendo diferença significativa nos dois últimos. Entretanto, há de se destacar (*i*) o baixo tempo de execução do método de descida rápida – máximo de 1,2 minuto – em contraposição ao tempo de execução do VND, que ultrapassa 65 minutos para algumas instâncias; (*ii*) as consideráveis taxas de soluções ótimas obtidas nos quatro primeiros conjuntos de instâncias – mínimo de 79,17% e 84,78% respectivamente para o VND e o método de descida rápida; (*iii*) os baixos valores de *gap* para ambos os métodos em todos os conjuntos de instâncias– máximo de 2,69% para o VND e 1,61% para o método de descida rápida – o que indica a robustez dos métodos ao obterem soluções próximas aos valores ótimos.

Os trabalhos futuros serão concentrados em aprimorar as estruturas de vizinhança do VND, experimentando diferentes métodos e também na implementação do métodos Busca em Vizinhança Variável, considerado um aprimoramento do VND. Adicionalmente, o método descida rápida pode ser inserido em métodos *branch and bound* da literatura, pois fornece um

limite superior razoável para o problema em pouco tempo, além de poder ser embutido em uma heurística híbrida, ou *matheuristic*.

Referências Bibliográficas

Becceneri, J. C. (1999). O problema de sequenciamento de padrões para a minimização do número máximo de pilhas abertas em ambientes de cortes industriais. *European Journal of Operational Research*, p. 145.

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.

Carvalho, M. A. M. d. e Soma, N. Y. (2014). A breadth-first search applied to the minimization of open stacks. *Journal of the Operational Research Society*. (to appear).

Chu, G. e Stuckey, P. J. (2009). Minimizing the maximum number of open stacks by customer search. In *Proceedings...*, volume 5732 of *L*ecture Notes in Computer Science, pp. 242–257, Berlin. INTERNATIONAL CONFERENCE ON PRINCIPLES AND PRACTICE OF CONSTRAINT PROGRAMMING, Springer.

de la Banda, M. G. e Stuckey, P. J. (2007). Dynamic programming to minimize the maximum number of open stacks. *INFORMS Journal on Computing*, 19(4):607–617.

Faggioli, E. e Bentivoglio, C. A. (1998). Heuristic and exact methods for the cutting sequencing problem. *European Journal of Operational Research*, 110(3):564–575.

Fink, A. e Voß, S. (1999). Applications of modern heuristic search methods to pattern sequencing problems. *Computers & Operations Research*, 26(1):17–34.

Gonçalves, J. F.; Resende, M. G. C. e Costa, M. D. (2016). A biased random-key genetic algorithm for the minimization of open stacks problem. *International Transactions in Operational Research*, 23(1-2):25–46.

Hansen, P. e Mladenović, N. (2001). Variable neighborhood search: Principles and applications. *European Journal of Operational Research*, 130(3):449 – 467.

Hu, Y. H. e Chen, S.-J. (1990). Gm plan: a gate matrix layout algorithm based on artificial intelligence planning techniques. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 9(8):836–845.

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

Möhring, R. H. (1990). *Graph Problems Related to Gate Matrix Layout and PLA Folding.* Springer Vienna.

Smith, B. e Gent, I. (2005). Constraint modeling challenge. In Smith, B. e Gent, I., editores, *T*he fifth workshop on modelling and solving problems with constraints, Edinburgh, Scotland. IJCAI.

Yanasse, H. (1996). Minimization of open orders-polynomial algorithms for some special cases. *Pesquisa Operacional*, 16(1):1–26.

Yanasse, H. e Limeira, M. (2004). Refinements on an enumeration scheme for solving a pattern sequencing problem. *International Transactions in Operational Research*, 11(3):277–292.

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

Yanasse, H. H. (1997b). A transformation for solving a pattern sequencing in the wood cut industry. *Pesquisa Operacional*, 17(1):57–70.

Yanasse, H. H. e Senne, E. L. F. (2010). The minimization of open stacks problem: A review of some properties and their use in pre-processing operations. *European Journal of Operational Research*, 203(3):559–567.

Yuen, B. J. (1991). Heuristics for sequencing cutting patterns. *European Journal of Operational Research*, 55(2):183–190.

Yuen, B. J. (1995). Improved heuristics for sequencing cutting patterns. *European Journal of Operational Research*, 87(1):57 – 64.

Yuen, B. J. e Richardson, K. V. (1995). Establishing the optimality of sequencing heuristics for cutting stock problems. *European Journal of Operational Research*, 84(3):590–598.