Universidade Federal de Ouro Preto - UFOP
Instituto de Ciências Exatas e Biológicas - ICEB
Departamento de Computação - DECOM

Algoritmos Heurísticos e Metaheurísticos para o Problema de Minimização de Pilhas Abertas

Bolsista: Júnior Rhis Lima
Orientador: Prof. Dr. Marco Antonio Moreira de Carvalho – DECOM/UFOP

Nota: Relatório Final referente ao período de 01/08/2014 a 01/08/2015, apresentado à Universidade Federal de Ouro Preto, como parte das exigências do programa de iniciação científica -PIBIC

Local: Ouro Preto - Minas Gerais - Brasil
Data: 11 de setembro de 2015

Resumo

Problemas de corte são problemas combinatórios, encontrados em sua maioria em indústrias e outros sistemas produtivos, onde frequentemente surge o cenário da produção de itens menores a partir do processamento de itens maiores, envolvendo o corte, a estocagem, a preparação de máquinas, o atendimento de demandas específicas, a utilização de recursos e a minimização de desperdícios. Correlatos aos problemas de corte, os problemas de sequenciamento de padrões consistem na fase subsequente do processo produtivo e possuem influência direta das etapas anteriores. Tais problemas são caracterizados pela ordem em que os itens maiores são processados, envolvendo dentre outros, a manipulação, o transporte e também o atendimento de demandas específicas. Tais problemas possuem grande impacto financeiro no processo de produção e têm se tornado mais relevantes a cada dia, ganhando cada vez mais espaço no planejamento industrial em contextos em que metal, papel, vidro ou madeira são utilizados como matéria prima. O presente projeto de pesquisa propõe o desenvolvimento de abordagens heurísticas e metaheurísticas para o Problema de Minimização de Pilhas Abertas, um problema de sequenciamento de padrões de ampla aplicação prática.

Assinatura do orientador(a): Marco Antonio M. Carvalho

Marco Antonio Moreira de Carvalho

Assinatura do bolsista: _____

Júnior Rhis Lima

Sumário

Capítulo 1

Introdução

Com a evolução e a disseminação da tecnologia, cada vez mais os setores industriais confiam o tratamento de questões operacionais a métodos computacionais, os quais têm sido aplicado com sucesso, contribuindo para a otimização de sistemas de produção.

Os problemas de corte são problemas de otimização combinatória, abordados frequentemente devido à sua aplicabilidade prática em contextos industriais. Entre os problemas de corte citam-se problemas de corte de estoque, carregamento de paletes e contêineres, alocação de tarefas e balanceamento de linhas de produção, entre outros. Estes problemas consistem em cortar unidades menores a partir de unidades maiores que possuem formas e medidas bem especificadas, satisfazendo-se algum objetivo, como por exemplo, minimizar as sobras das unidades maiores, dado que dificilmente aproveita-se o todo. As unidades maiores podem ser barras, bobinas, chapas e outros objetos feitos de papel, madeira, vidro ou metal, que devem ser cortados em peças menores.

A disposição de unidades menores (ou peças) dentro de unidades maiores para realização do corte define um plano (ou padrão) de corte. Geralmente em problemas de corte, as peças a serem cortadas possuem grande demanda, implicando no corte de vários objetos em estoque e na repetição de vários padrões de corte. A este tipo de problema denomina-se Problema de Corte de Estoque, onde o objetivo pode ser minimizar o número de unidades maiores (diminuindo assim as sobras ou desperdício) ou minimizar os custos dos objetos cortados. Os custos variam de acordo com os critérios de qualidade estabelecidos.

Após a definição dos padrões de corte (ou simplesmente padrões), passa-se à etapa do corte (ou processamento) dos mesmos. Porém, a ordem na qual os padrões são processados pode afetar a produtividade e os custos relacionados ao processo caso alguns critérios de qualidade sejam desrespeitados. 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 ao redor da máquina que a produziu; cada pilha armazena somente peças de um mesmo tipo. Quando a última peça de um tipo for produzida, a respectiva pilha pode ser removida de perto da máquina que a 3 processou para outro local. Admite-se a existência de uma limitação física para o armazenamento de pilhas ao redor da máquina de produção, de tal forma que não haja espaço suficiente para que uma pilha para cada tipo de peça esteja aberta ao mesmo tempo. Sendo assim, deseja-se minimizar o número de pilhas abertas simultaneamente pelo sequenciamento dos padrões, caracterizando assim o Problema de Minimização de Pilhas Abertas.

1.1 Descrição do Problema

O Problema de Minimização de Pilhas Abertas (ou MOSP, de Minimization of Open Stacks Problem), como descrito por (YUEN, 1995) remonta à um ambiente de produção em que peças com demandas específicas são produzidas por uma única máquina de corte. O processamento dos padrões de corte a partir das quais as peças são cortadas se dá em lotes, ou seja, todas as cópias de um mesmo padrão de corte devem ser cortadas antes que um padrão de corte diferente seja processado. A produção é dividida em estágios, e é considerado que em cada estágio um lote de um determinado padrão de corte diferente é 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 de peça. Quando a primeira peça de um dado tipo tiver de ser produzida, a respectiva pilha é considerada aberta e este estado permanece até que a última peça do mesmo tipo seja produzida, quando então a pilha é considerada fechada, podendo ser removida para outro local para dar continuidade ao processo de produção.

A Figura 1 apresenta um exemplo de instância MOSP, em que são fornecidos dados sobre a composição dos padrões. Note que informações relativas à quantidade de peças de um mesmo tipo na composição de um dos padrões não são relevantes para este problema específico – esta, entre outras informações como limite máximo para a altura das pilhas e diferentes tempos de processamento para diferentes tipos de peças não são consideradas no MOSP. O conjunto de todos os tipos de peças é P, e os tipos de peças são enumerados de 1 a 6, representadas na vertical. O conjunto dos padrões de corte é S, e os padrões são enumerados de p_1 a p_6, representados na horizontal. O símbolo $\otimes$ indica que o padrão p_i contém a peça j. Espaços em branco indicam o contrário. Por exemplo, o padrão p_1 é composto pelas peças 1, 2 e 4, e o padrão p_2 é composto pelas peças 2, 4, 5 e 6.

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

Figura 1.1: Um exemplo de instância do MOSP

A Figura 2 apresenta dois possíveis sequenciamentos do conjunto S do exemplo anterior. Uma pilha é associada a cada peça, e também é representada na horizontal. Na vertical, são representados os estágios de produção, nos quais um lote de determinado padrão é processado. A notação é definida a seguir:

- O símbolo Δ indica a abertura de uma determinada pilha;
- O símbolo □ indica que uma peça do tipo j foi produzida a partir do padrão pi;
- O símbolo ▽ indica o fechamento de uma pilha;
- O símbolo – indica que uma pilha permanece aberta, embora não esteja sendo produzida nenhuma cópia da peça correspondente naquele estágio de processamento, o que caracteriza uma descontinuidade em sua produção;
- As pilhas abertas são contabilizadas a cada estágio, na vertical, de modo que no referido sequenciamento (à esquerda na referida figura), no primeiro estágio, quando o padrão p_3 é processado, duas pilhas são abertas – referentes às peças 3 e 4. No segundo estágio, quando o padrão p_4 é processado, existem 4 pilhas abertas – referentes às peças 1, 3, 4 e 5. Note que apesar de a peça 4 não ser produzida a partir do padrão p_4, sua pilha permanece aberta e por isto deve ser contabilizada. Note também que a pilha referente à peça 3 é fechada após o término deste segundo estágio e também é contabilizada na quantidade de pilhas abertas.

No primeiro sequenciamento 2(a), a lista que contém a ordenação dos padrões $L_{pa} = p_3, p_4, p_5, p_1, p_2, p_6$, o que indica que no primeiro estágio serão processadas as cópias do padrão p_3, no segundo estágio serão processadas as cópias do padrão p_4 e assim sucessivamente até que no último estágio sejam processadas as cópias do padrão p_6.

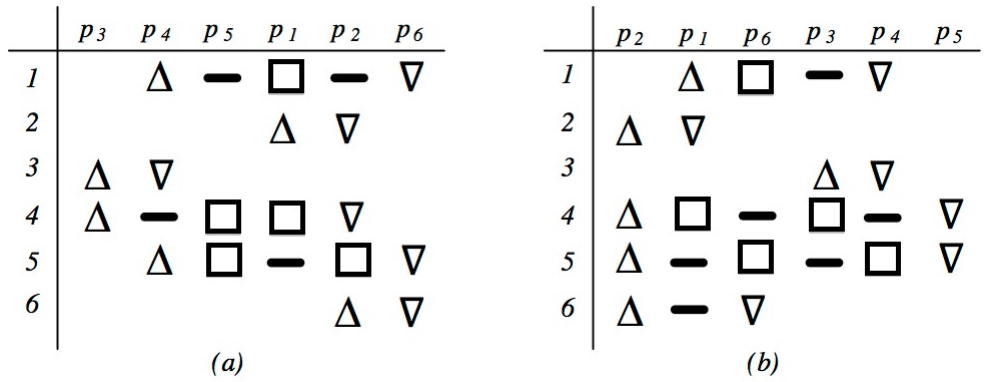

Figura 1.2: Duas possíveis sequências para processamento de padrões.

Neste sequenciamento (à esquerda na referida figura), o número máximo de pilhas abertas é 5; no quinto estágio, após o processamento do padrão p_2, as peças 2 e 4 terão suas pilhas fechadas (linha 2 e colunas 4-5 e linha 4 e colunas 1-5), mas o padrão p_2 também abre a pilha associada à peça 6 (linha 6 e colunas 4-5). O segundo sequenciamento (à direita na referida figura) apresenta uma solução com sequenciamento de padrões diferente, $L_{pa} = p_2, p_1, p_6, p_3, p_4, p_5$, e com número máximo de pilhas abertas igual a 5.

Neste problema, não é suficiente apenas gerar uma agenda de processamento associada ao conjunto S de padrões e consequente fabricação do conjunto P de peças de forma a atender as demandas; também, é necessário atingir a melhor utilização do espaço físico disponível,

agilizando a linha de produção. Admite-se a existência de uma limitação física para o armazenamento de pilhas ao redor da máquina de produção, de tal forma que não haja espaço suficiente para que as pilhas de cada tipo de peça estejam abertas simultaneamente. Sendo assim, 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. Posteriormente, quando as peças referentes a estas pilhas removidas forem produzidas, as mesmas são trazidas de volta para o ambiente de produção, eventualmente devendo outras serem removidas para liberação do espaço necessário.

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. Além disso, alguns tipos de peças são frágeis, como na indústria de vidro, em que o manuseio excessivo pode gerar outros prejuízos. É desejável então que as pilhas de peças sejam removidas uma única vez, depois de fechadas, para diminuir os custos relacionados à produção das peças.

Este é um problema NP-Difícil (LINHARES; YANASSE, 2002) e possui aplicações em diferentes áreas, bem como diferentes problemas de formulação equivalente (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.*

Capítulo 2

Objetivos

1. Elaborar heurísticas consistentes e robustas que possam ser utilizadas no contexto do problema abordado que 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;

2. Avaliar o comportamento heurístico de diferentes formulações para o problema abordado;

3. Pesquisar técnicas para melhoria fina de soluções (polishing) obtidas por heurísticas usando programação linear inteira;

4. Ampliar o conjunto de problemas testes com soluções ótimas já conhecidas através da execução de testes sistemáticos usando formulações já desenvolvidas;

5. Será buscada ainda a aplicação prática dos métodos desenvolvidos em contextos reais, a fim de que também seja constituído um avanço para as indústrias nacionais;

6. Além dos objetivos principais, outros produtos deste projeto de pesquisa serão trabalhos publicados em periódicos e eventos nacionais e internacionais, os quais contribuem para a promoção dos centros de pesquisas nacionais e também da tecnologia.

2.1 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. Implementação computacional de um software para determinação de soluções de alta qualidade para o problema abordado usando ferramentas de Inteligência Computacional, o que inclui a utilização de métodos heurísticos e metaheurísticos;

3. Avaliação do software implementado considerando dados reais e também com problemas teste publicamente disponíveis;

4. Promover o Departamento de Computação e a Universidade Federal de Ouro Preto por meio da divulgação dos resultados obtidos neste projeto de pesquisa.

Capítulo 3

Fundamentos Teóricos e Revisão

O presente trabalho visa elaborar algoritmos eficientes para um problema de grande importância prática na indústria, sendo relevante nas indústrias que realizam o processamento de matérias-primas tais como metal, madeira, vidro e papel para a fabricação de bens de consumo. O problema em questão é bastante genérico, sendo similar a um grande conjunto de problemas de escalonamento.

O trabalho de (LINHARES; 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*).

Visando diminuir o tamanho do problema, métodos de pré-processamento podem ser utilizados. Em (YANASSE; SENNE, 2010), sete métodos de pré-processamento para o MOSP são discutidos, sendo que todos visam reduzir o tamanho dos problemas tratados por meio da decomposição desses problemas ou da eliminação de redundâncias - dados que podem ser desconsiderados ao se resolver o problema por não alterarem sua estrutura básica e, portanto, não interferir em sua solução. Estes métodos envolvem a análise de algumas propriedades do problema e também sua representação em grafos MOSP - nos quais os vértices correspondem às peças presentes em uma instância MOSP, sendo conectados os vértices presentes na composição de uma mesmo padrão de corte. Um exemplo da formação de um grafo MOSP a partir da união de cliques pode ser visualizado na 5.1.

Devido a sua complexidade e equivalência com vários outros problemas, além da aplicação em manufatura, em 2005 o MOSP foi escolhido como tema do *First Constraint Modelling Challenge* (SMITH, B. M., Gent, I. P. (eds), 2005). O modelo vencedor do desafio foi o proposto por (BANDA; STUCKEY, 2007).

Posteriormente, (CHU; STUCKEY, 2009) propuseram uma nova estratégia ao modelo utilizado por (BANDA; STUCKEY, 2007). Na nova estratégia as peças são sequencidas primeiro e logo em seguida ocorre o sequenciamento dos padrões de acordo com a lista de peças formada. Dessa forma, os padrões não são sequenciados diretamente, ao contrário do método anterior. De acordo com os autores, esta estratégia é mais efetiva por fornecer dados que proporcionam um maior controle sobre a abertura e o fechamento das pilhas.

Os trabalhos pioneiros (YUEN, 1991) (YUEN, 1995) contém as heurísticas mais difundidas para o tratamento do MOSP. Nestes trabalhos são apresentadas seis heurísticas rápidas e simples envolvendo pilhas abertas e padrões ainda não sequenciados. Entre estas heurísticas se encontra a *Yuen3*, considerada como a de melhor desempenho durante um período de tempo. A Heurística de Nó de Custo Mínimo (BECCENERI; YANASSE; SOMA, 2004) é considerada como a de melhor desempenho atualmente. Os experimentos comprovam a superioridade dessa heurística em relação a qualidade da solução quando comparada ao método *Yuen3*, que por sua

vez, possui um custo computacional menor que o da Heurística de Nó de Custo Mínimo. A heurística é baseada em grafos MOSP e consiste no sequenciamento das arestas. Cada vértice incidido pelas arestas é considerado *aberto* e o objetivo é sequenciar todas as arestas que incidem sobre tais vértices, evitando ao máximo que novos vértices sejam *abertos*. Os vértices a serem *abertos* são aqueles de menor grau entre os vértices ainda não visitados presentes no grafo MOSP. Os graus dos vértices são atualizados a cada aresta sequenciada, a fim de que se tenha o controle das pilhas que já foram abertas. Posteriormente, a sequência de padrões é obtida por meio do sequenciamento dos vértices, de tal forma que, assim que todos os vértices que representam as peças que compõem um determinado padrão forem *abertos*, tal padrão é sequenciado.

Face à importância do problema, o mesmo foi escolhido como tema de grupos de cooperação acadêmica internacional como o *SCOOP Project* (*Sheet Cutting and Process Optimization for Enterprises*) (NADA, a) e OPTSICOM Project (*Optimization of Complex Systems*) (OPTSICOM Project, 2010,), que reúnem universidades da Alemanha, Espanha, Estados Unidos, Itália, México e Portugal, uma das mais importantes sociedades internacionais de praticantes das áreas de Pesquisa Operacional, Engenharia e Ciência da Administração.

A criação de novos modelos para solução do problema em questão pode ainda contribuir para melhoria da eficiência da indústria nacional de bens de consumo, amplamente discutida como insuficiente no cenário mundial.

Capítulo 4

Materiais e Métodos

A metodologia aplicada no desenvolvimento deste projeto inclui as seguintes etapas:

1. Estudo dos principais métodos de resolução apresentados na literatura como de sucesso para a resolução do problema considerado e correlatos;
2. Revisão de literatura considerando heurísticas para o problema tratado e problemas correlatos;
3. Avaliação de técnicas de inteligência computacional promissoras que não foram ainda consideradas na literatura para o problema abordado;
4. Projeto e análise experimental de um algoritmo construtivo para o problema;
5. Projeto e análise experimental de um algoritmo de busca local para o problema;
6. Implementação da(s) técnica(s) consideradas mais promissoras para a resolução do problema;
7. Avaliação computacional das implementações utilizando as bases de dados do *SCOOP Project* (NADA, a), OPTSICOM (OPTSICOM Project, 2010,), *Constraint Modelling Challenge* (SMITH, B. M., Gent, I. P. (eds), 2005) e também bases de dados disponíveis na literatura ou dados reais de empresas nacionais;

Capítulo 5

Desenvolvimento

O trabalho de (HANSEN; MLADENOVIć, 2001) propõe metaheurísticas simples e eficazes baseadas na busca local para problemas de otimização combinatória, as técnicas utilizadas pelo autor denominadas Variable Neighborhood Descent (VND), Variable Neighborhood Search (VNS), Reduced Variable Neighborhood Search (RVNS) e Variable Neighborhood Decomposition Search (VNDS), tendo as duas primeiras um maior grau de importância e eficiência, em especial a primeira que foi tomada como base na implementação do VND aprimorado desenvolvida neste trabalho.

Neste capítulo serão descritas as implementações utilizados no decorrer deste trabalho assim como todas as etapas presentes nas implementações. Serão descritos termos como representação por grafos, pré-processamento por dominância, BFS, sequenciamento de peças, sequenciamento de padrões, refinamento, Busca Local e *k-opt*, que estão diretamente relacionados com a implementação proposta. Será descrito ainda o VND original, que serviu como base para o VND aprimorado.

5.1 *Variable Neighborhood Descent*

O VND é uma metaheurística baseada no algoritmo de Busca Local do tipo *k-opt*, cujo o funcionamento consiste na análise da vizinhança de uma solução, ou seja, na verificação de soluções oriundas de uma solução inicial que diferem em k elementos quando comparadas a solução original. O VND é composto de três etapas, que se repetem enquanto houver uma melhoria da solução ou até que todas as k vizinhanças sejam analisadas. O algoritmo 1 refere-se ao pseudocódigo do VND.

Algoritmo 1: Pseudocódigo VND

Inicialização: Selecione o conjunto de estruturas de vizinhança $N_k = 1, ..., k_{max}$ que serão usadas na análise. Encontre uma solução inicial x.

```
repita
    defina k ← 1;
    melhoria ← falso;
    repita
        Encontre o melhor vizinho x' de x (x' ∈ N_k(x));
        if x' melhor que x then
            x ← x';
            melhoria ← verdadeiro;
        else
            k ← k + 1;
        end
    até k = k'_max;
até melhoria = falso;
```

A inicialização do algoritmo consiste na seleção das vizinhanças. A vizinhança é definida por meio da busca local *k-opt* cujo funcionamento é descrito na subseção 5.4.1. É gerada então uma solução inicial descrita na seção 5.2 que será analisada no decorrer do processamento.

As próximas etapas do algoritmo consistem basicamente em visitar todo o espaço das vizinhanças definidas verificando se houve uma melhoria da solução, ou seja, se a solução encontrada pela busca local é melhor que a melhor solução encontrada até o momento, atualizando esta solução caso isso ocorra. O algoritmo é executado enquanto houver uma melhoria da solução.

5.2 Solução Inicial

A solução inicial do problema é gerada em etapas. A primeira etapa de geração da solução é o pré-processamento por dominância, nas etapas seguintes é feito o sequenciamento das peças e finalmente ocorre o sequenciamento dos padrões que formam a solução inicial, estratégia proposta por (CHU; STUCKEY, 2009). Os detalhes de cada uma das etapas serão descritos nas seções seguintes.

5.2.1 Representação por grafos

Grafo MOSP conforme descrito por (YANASSE, 1997), é um grafo que representa uma instância do MOSP, neste grafo os vértices representam as peças. Ocorre uma aresta entre dois vértices somente se as peças referentes a esses vértices ocorrem simultaneamente em pelo menos um padrão de corte. Desta forma, como as peças de um mesmo padrão são ligadas entre si, são formados cliques no grafo fazendo com que o grafo MOSP seja composto pela união de cliques de uma instância. A figura 5.1 extraída do trabalho de (CARVALHO; SOMA, 2011) apresenta o exemplo de formação de um grafo MOSP a partir de união de cliques.

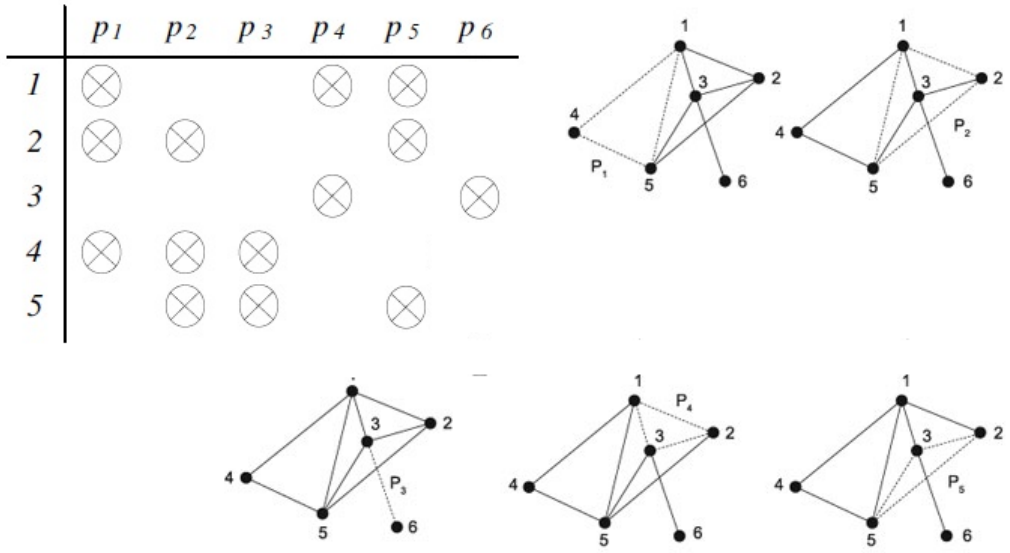

Figura 5.1: Exemplo de formação do grafo MOSP a partir da união de cliques.

Em relação a figura 5.1, primeiramente são definidas as adjacências, responsáveis pela formação da clique, referentes aos vértices(peças) presentes no padrão p_1, onde são ligados os vértices referentes as peças *1*, *4* e *5*. Os padrões restantes formam novos cliques que são ligados entre si por peças que se localizam em mais de um padrão, como por exemplo a peça 1 responsável em ligar os cliques dos padrões P_1, P_2 e P_3, ou seja, os padrões dos quais ela faz parte da composição. Ao final o grafo possui seis vértices referentes as seis peças presentes na instância do problema.

Esta representação é utilizada principalmente em métodos heurísticos para tratamento do MOSP. Como mostram os trabalhos de (ASHIKAGA; SOMA, 2009), (BECCENERI, 1999) e (BECCENERI; YANASSE; SOMA, 2004).

5.2.2 Pré-processamento por Dominância

O Pré-processamento por Dominância consiste na eliminação de redundâncias - dados que podem ser desconsiderados ao se resolver o problema por não alterarem sua estrutura básica (YANASSE; SENNE, 2010). O Pré-processamento por Dominância remove vértices 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 todas as peças de outro padrão, podendo ter peças a mais. O padrão dominado é aquele que está contido em outro padrão. Quando a solução final for gerada, os padrões dominados são sequenciados imediatamente

após seus respectivos padrões dominantes. A figura 5.2 apresenta um exemplo da aplicação do pré-processamento por dominância em uma instância do problema.

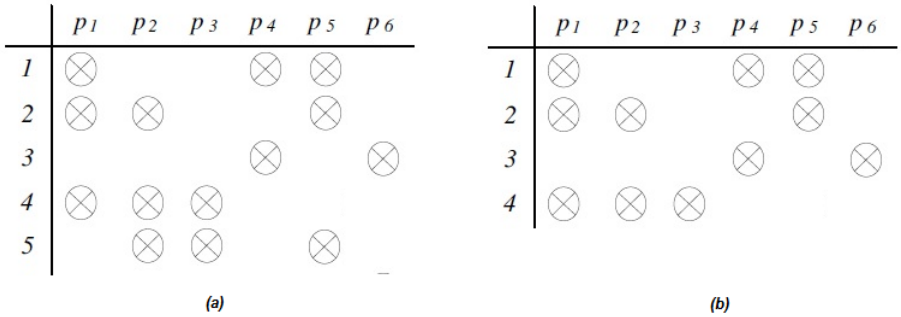

Figura 5.2: Exemplo de redução feita através do pré-processamento por dominância.

Em um primeiro momento, analisando a figura 5.2 (a) verifica-se que a composição do padrão P_5 formado pelas peças *2* e *3*, é um subconjunto da composição de peças do padrão P_2 que possui as peças *1*, *2* e *5*. Dessa forma, o pré-processamento por dominância elimina o padrão P_5, figura 5.2 (b), da instância do problema, que é sequenciado após o padrão P_2 na solução final sem causar qualquer prejuízo a mesma.

5.2.3 *Breadth-First Search*

Busca em largura (ou busca em amplitude, do termo em inglês *Breadth-First Search* - BFS) é um algoritmo de busca em grafos utilizado para realizar uma varredura em todos os vértices de um grafo para fins de busca por vértices específicos. O funcionamento de uma BFS é simples, primeiramente escolhe-se um vértice aleatório e explora todos os vértices vizinhos do vértice escolhido que ainda não tenham sido visitados. Então, para cada um dos vértices vizinhos, explora-se os seus vértices vizinhos e assim sucessivamente, até que o vértice o qual se procura seja encontrado e/ou não exista nenhum outro vértice a ser visitado.

A figura 5.3 ilustra um grafo para o qual é descrito logo em seguida como seria a execução de uma BFS no mesmo.

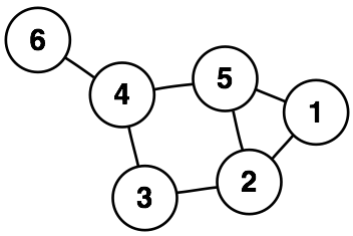

Figura 5.3: Grafo exemplo sobre o qual será executado o algoritmo da BFS.

A primeira etapa do algoritmo consiste na seleção de um vértice inicial, ou vértice de partida, sobre o qual o algoritmo terá início. Neste exemplo o vértice inicial será o 6, mas poderia ser qualquer outro vértice do grafo. A lista de vértices até o momento é composta somente pelo vértice 6. O próximo passo consiste na análise dos vizinhos do vértice selecionado. Para o grafo do exemplo o único vizinho do vértice 6 é o 4, dessa forma ele é imediatamente sequenciado, ou seja, adicionado na lista de vértices. A solução passa a ser formada pelos vértices 6 e 4, S = [6,

4]. Dos vértices presentes na solução somente o vértice 4 possui vizinhos ainda não visitados, os vértices 5 e 3. Dessa forma os vizinhos do vértice 4 serão visitados, neste ponto a ordem em que os vértices serão selecionados depende da implementação da BFS, neste exemplo, por boa prática, os vértices serão sequenciados por ordem crescente do seu valor, diante disso os vértices 3 e 5, respectivamente, serão inseridos na solução. A nova solução será S = [6, 4, 3, 5].

Dos vértices presentes na solução, somente os vértices 3 e 5 possuem vizinhos ainda não selecionados. Sendo que os vértices 3 e 5 possuem o vértice 2 como vizinho em comum. Assim como a escolha dos vértices não selecionados é feita em ordem crescente, a análise da vizinhança será feita dos vértices presentes na solução de menor valor para os de maior valor. Dessa forma, como o vértice 3 é "menor"que o vértice 5, primeiramente será sequenciado o vértice 2 que também é vizinho do vértice 3 e logo em seguida o vértice 1, vizinho do vértice 5. A solução final será S = [6,4,3,5,2,1].

5.2.4 Sequenciamento das Peças

O sequenciamento das peças é feito com auxílio de uma BFS que é executada sobre o grafo MOSP. O sequenciamento dos vértices, referentes as peças presentes na instância do problema, é relativamente simples, primeiramente é sequenciado o vértice com menor grau dentre todos, ou seja, o vértice que apresenta o menor número de vizinhos. Em seguida, todos os seus vizinhos são sequenciados em ordem crescente do grau. Os próximos vértices a serem sequenciados são os vizinhos do próximo vértice presente na lista de vértices. Caso o grafo seja desconexo, o processo se repete com o sequenciamento de um outro vértice com o menor grau dentre os demais vértices que ainda não tenham sido adicionados na solução.

Como exemplo, considere a figura 5.1. A partir do grafo MOSP formado, o primeiro vértice a ser sequenciado será o *6*, os próximos vértices serão os vértices adjacentes ao vértice *6*, como ele possui grau 1, somente o o vértice *3* é adicionado na lista de vértices. O próximo vértice a ser analisado é o *3*, a partir do qual serão sequenciados os vértices *4* (grau 2), *1* (grau 3), *2* (grau 3) e *5* (grau 3), sequenciando em ordem crescente os vértices de mesmo grau. Como todos os vértices foram adicionados o sequenciamento das peças será L_{pe} = [6,3,4,1,2,5]. Caso ainda houvessem vértices a serem sequenciados o próximo vértice analisado seria o *4*, depois o *1* e assim sucessivamente. Se o grafo for desconexo todo o processo se repete para os demais componentes do grafo.

5.2.5 Sequenciamento dos Padrões

O sequenciamento dos padrões é obtido por meio da estratégia elaborada por (BECCENERI, 1999), a técnica baseia-se no sequenciamento dos padrões de acordo com uma lista de peças previamente definida, procedimento descrito na seção 5.2.4. O algoritmo consiste na abertura sucessiva das pilhas(peças) presentes na lista de peças e na verificação da composição dos padrões, de modo que, caso a composição de um padrão seja o conjunto ou subconjunto das pilhas abertas naquele instante, este padrão é imediatamente sequenciado.

Como exemplo, considere a figura 5.1. A partir da composição dos padrões e da lista de peças L_{pe} = [6,3,4,1,2,5] definida na seção 5.2.4, primeiramente será aberta a pilha referente a peça *6*, que é a primeira da lista de peças, como a peça 6 não se encontra sozinha em nenhum padrão a próxima pilha referente a peça *3* será aberta, nesse momento somente o padrão P_3 será sequenciado, por possuir exatamente as peças *3* e *6* em sua composição. As próximas pilhas abertas serão referentes as peças *4*, *1* que não sequenciariam nenhum novo padrão. Ao ser aberta a pilha referente a peça *2* o padrão P_4 cuja composição é *1*, *2* e *5* será sequenciado. Com

a abertura da última pilha referente a peça *5* os padrões restantes *1*, *2* e *5* serão adicionados e o sequenciamento resultante será L_{pa} = [3,4,1,2,5].

5.3 Refinamento

O refinamento de uma solução é uma técnica que consiste na variação do sequenciamento dos padrões de acordo com algum critério, visando reduzir a quantidade de pilhas abertas na solução.

O refinamento proposto neste trabalho tem como critério a análise da similaridade da composição dos padrões em relação as pilhas abertas no gargalo (ou coluna crítica) da solução. O gargalo de uma solução é o padrão que mantém a maior quantidade de pilhas abertas simultaneamente. São consideradas pilhas abertas as peças que compõem o gargalo assim com as descontinuidades presentes em função da localização do padrão na solução. Na figura 1.2 *(a)* o gargalo está localizado no padrão p_2, composto pelas peças *2*, *4*, *5* e *6*, e apresenta uma descontinuidade provocada pela peça *1*, resultando em 5 pilhas abertas.

Verifica-se que a quantidade de pilhas abertas no gargalo é a resposta do problema, e portanto o objetivo do problema torna-se sequenciar os padrões de forma a minimizar a quantidade máxima de pilhas abertas simultaneamente no gargalo do problema. O algoritmo 2 apresenta o pseudocódigo referente ao refinamento proposto.

Algoritmo 2: Pseudocódigo Refinamento

```
inicio
    Entrada: Solução Inicial x
    x' ← x;
    s1 ← quantidade de pilhas abertas em x';
    repita
        Encontre o gargalo de x;
        Liste os padrões que tem relação com o gargalo;
        Ordene os padrões de acordo com a maior semelhança com o gargalo;
        melhoria ← falso;
        repita
            p ← próximo padrão;
            Procura a melhor posição para inserção de p;
            Altera x inserindo p em outra posição;
            s2 ← quantidade de pilhas abertas em x;
            if s2 < s1 then
                s1 ← s2;
                x' ← x;
                melhoria ← verdadeiro;
            else
                x ← x';
            end
        até que haja padrões;
    até melhoria = falso;
fin
```

A entrada do algoritmo consiste na solução inicial x na qual será aplicado o refinamento. A solução é então guardada na variável auxiliar *x'* que guarda a melhor solução corrente, uma

vez que *x* será alterada no decorrer do processamento e é preciso conhecer a melhor solução até o momento, caso a nova solução *x* apresente uma solução pior que *x'*. A variável *s1* guarda a quantidade de pilhas abertas em *x* antes do refinamento e será utilizada na comparação com a quantidade de pilhas abertas em *x* após o refinamento.

A próxima etapa do algoritmo é executada enquanto a solução obtida for melhor que a solução corrente, ou seja, enquanto houver redução das pilhas abertas. Primeiramente é definido o gargalo de *x*. De posse do gargalo, os padrões são listados e ordenados de acordo com o maior grau de semelhança das pilhas que os compõem em relação as pilhas abertas no gargalo. Para cada padrão *p* é feita uma análise em *x* procurando a melhor posição para inserção de *p* de forma a gerar a menor descontinuidade possível em *p* na nova posição. A quantidade de pilhas abertas na nova solução *x* é então guardada em *s2*. Caso *s2* seja menor que *s1*, que é a menor quantidade de pilhas abertas até o momento, *s1* é atualizada com o valor de *s2* e a melhor solução até o momento *x'* é atualizada com *x*, neste caso houve uma melhoria. Quando a melhoria não ocorre a solução corrente *x* é atualizada com a melhor solução até o momento *x'*, e a movimentação do próximo padrão é processada enquanto houverem padrões a serem analisados.

5.4 Busca Local

Em ciência da computação, busca local é um método metaheurístico para resolver problemas de otimização, amplamente aplicado em problemas computacionalmente difíceis. A busca local pode ser utilizada em problemas que necessitem encontrar uma solução dentre um certo número de soluções candidatas. Os algoritmos de busca local analisam cada uma das soluções em um espaço de soluções possíveis (espaço de busca), aplicando as alterações locais, até encontrar uma solução considerada ótima ou suficientemente boa dadas as condições de tempo e complexidade do espaço de busca. A figura 5.4 referente ao panorama do espaço de estados, ou soluções, de um problema.

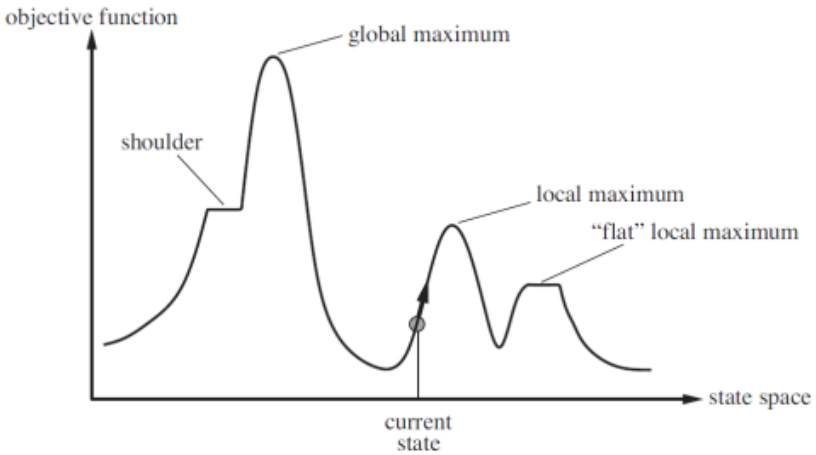

Figura 5.4: Panorama do espaço de soluções.

A ideia básica de um algoritmo de busca local é partir de uma solução e analisar soluções vizinhas, verificando se são melhores que a solução anterior atualizando a solução caso isso ocorra. A partir da análise da figura 5.4 é possível verificar que a próxima solução, do estado de soluções possíveis, a ser analisada pode ser pior, igual ou melhor que a solução corrente, podendo inclusive ser a solução ótima, caso o espaço de busca englobe todas as possíveis soluções do problema e a solução analisada seja um ótimo global.

5.4.1 K-opt

k-opt é um algoritmo de busca local cuja ideia principal consiste na realização de transições entre os elementos pertencentes a solução de um problema, gerando soluções que diferem em exatamente *k* elementos da solução inicial. A exemplo do MOSP, considere um problema em que são dispostos 4 padrões, não importando a composição dos mesmos. A figura 5.5 ilustra as transições decorrentes da aplicação da busca local *2-opt* entre todos os padrões presentes na solução.

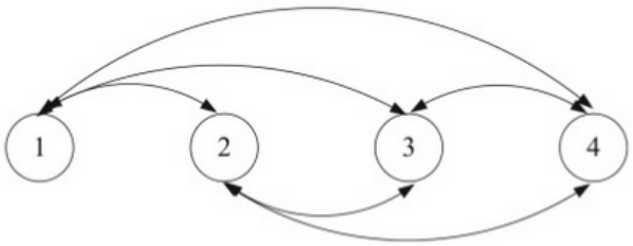

Figura 5.5: Exemplo de transições no *2-opt*.

A análise da figura 5.5 permite a visualização das trocas realizadas entre os padrões presentes na solução original x = [1,2,3,4]. As vizinhanças da solução x a partir da aplicação do *2-opt* seriam:

1. $x1 = 2,1,3,4$
2. $x2 = 3,2,1,4$
3. $x3 = 4,2,3,1$
4. $x4 = 1,3,2,4$
5. $x5 = 1,4,3,2$
6. $x6 = 1,2,4,3$

Portando *6* vizinhanças referentes as combinações tomadas *2* a *2* em relação aos padrões que compõem uma solução.

5.5 VND Aprimorado

O VND Aprimorado é similar ao VND proposto por (HANSEN; MLADENOVIć, 2001), porém algumas variações foram propostas com o intuito de otimizar o desempenho do método em relação à quantidade de pilhas abertas resultante.

A nova metaheurística apresenta uma pequena mudança, em vez de serem analisados os vizinhos da solução x a fim de se escolher o melhor e compará-lo com x, a proposta é aplicar o refinamento, apresentado na seção 5.3, para cada vizinho de x e selecionar o melhor vizinho x' obtido com a aplicação do refinamento. Este novo vizinho x' será comparado com a solução corrente x, que será atualizada caso possua uma quantidade de pilhas abertas superior a x'.

O algoritmo 3 apresenta o pseudocódigo referente ao novo VND com a variação proposta. A inicialização do algoritmo consiste na seleção das vizinhanças. A vizinhança é definida por meio da busca local *k-opt* cujo funcionamento é descrito na subseção 5.4.1. É gerada então uma solução inicial descrita na seção 5.2 que será analisada no decorrer do processamento.

Algoritmo 3: Pseudocódigo VND Aprimorado

```
Inicialização: Selecione o conjunto de estruturas de vizinhança N_k = 1,...,k_max que
serão usadas na análise. Encontre uma solução inicial x.
x' ← x;
sx' ← quantidade pilhas abertas em x';
repita
    k ← 1;
    melhoria ← falso;
    repita
        define s;
        define s';
        repita
            v ← próxima vizinhança de x;
            Aplica o refinamento em v;
            v' ← quantidade pilhas abertas em v;
            if v é a melhor vizinhança then
                s' ← v';
                s ← v;
        até que haja vizinhanças;
        if s' < sx' then
            x' ← s;
            sx' ← s';
            melhoria ← verdadeiro;
        else
            k ← k + 1;
        end
        x ← x';
    até k = k'_max;
até melhoria = falso;
```

O próximo passo do algoritmo consiste em atribuir a solução inicial x à x' e guardar a quantidade de pilhas abertas de x' ou x, nesse caso tanto faz, em sx'. Nesse ponto a variável x' armazena a melhor solução até o momento tendo em vista que nenhum processamento foi realizado.

A próxima etapa do algoritmo é executada enquanto a solução obtida for melhor que a solução corrente. Primeiramente é definida vizinhança a ser analisada que a princípio é a própria solução pois não ocorrem variações quando $k = 1$. O próximo passo consiste em analisar todas as vizinhanças da solução corrente. Nesse ponto são definidas as variáveis *s* responsável em armazenar o melhor vizinho da vizinhança *k* sendo analisada naquele momento após o refinamento e a variável *s'* que armazena a quantidade de pilhas abertas em *s*. Para se determinar melhor vizinhança da solução corrente é aplicado o refinamento descrito na seção 5.3 em cada uma dessas vizinhanças e a melhor vizinhança é salva em *s* assim como a quantidade de pilhas abertas nessa vizinhança é salva em *s'*.

O próximo passo consiste na comparação de *s'* com *sx'*, ou seja, é realizada uma comparação entre a melhor solução obtida na vizinhança após a aplicação do refinamento com a melhor solução até o momento. Caso *s'* seja menor que *sx'*, a melhor solução até o momento *x* recebe a melhor vizinhança *s* e a quantidade de pilhas abertas na melhor solução até o momento *sx'* recebe a quantidade de pilhas abertas na melhor vizinhança *s'*, neste caso houve uma melhoria. Quando a melhoria não ocorre a próxima vizinhança é selecionada para a análise. É importante ressaltar que a análise da próxima vizinhança será em relação à melhor solução até o momento, e desse modo, a solução sob análise *x* recebe *x'* que guarda a melhor solução, podendo ser inclusive a própria solução, considerando o pior caso em que não houver nenhuma melhoria.

A limitação do método proposto está no tempo de execução, que devido a complexidade elevada do algoritmo demanda um tempo de execução elevado quando executado para instâncias grandes do MOSP.

Capítulo 6

Resultados e Discussões

Neste capítulo será descrito o experimento realizado assim como a análise dos resultados obtidos pelo VND Aprimorado proposto neste trabalho. No experimento computacional realizado comparam-se o VND Aprimorado com o VND (HANSEN; MLADENOVIć, 2001), a fim de avaliar o desempenho dos métodos e a qualidade das soluções obtidas.

Os critérios usados na avaliação são a qualidade da solução obtida e o tempo de execução gasto para cada instância do problema.

6.1 Ambiente Computacional

O ambiente computacional utilizado no experimento consiste de um processador Intel Core i7-4500U 1,8 GHz de frequência e 8 GB de memória RAM, sob o sistema operacional Windows 8 64-bit.

6.2 Conjunto de instâncias

No experimento computacional conduzido utilizou-se o conjunto de instâncias *benchmark* proposto no *First Constraint Modeling Challenge* (SMITH, B. M., Gent, I. P. (eds), 2005). Foram selecionadas 4092 instâncias agrupadas por características comuns.

6.3 Experimento com instâncias *benchmark*

A tabela 6.1 apresenta os valores médios para os resultados referentes a grupos de instâncias que variam entre 10 e 40 padrões/peças e diferentes proporções entre padrões e peças. O nome de cada grupo segue o formato *Nome_m_n_p*, em que m denota o número de peças, n denota o número de padrões e p denota a quantidade de instâncias do grupo. A solução ótima é apresentada na segunda coluna da tabela.

Para cada um dos métodos é apresentado o tempo de execução médio gasto em milisegundos e *gap* (ou distância) entre duas soluções em porcentagem calculada de acordo com a equação abaixo, considerando A como referência à quantidade de pilhas abertas no resultado ótimo e B sendo a quantidade de pilhas abertas na solução obtida pelo método. Em outras palavras, *gap* mede a distância de uma solução encontrada à solução ótima.

$$gap_{B,A} = \text{mod}(100 \times \frac{solucaoB - solucaoA}{solucaoA})$$

6.4 Avaliação dos Resultados

A partir da análise da tabela 6.1 é possível verificar a eficiência da nova implementação proposta. Com exceção do conjunto *wbp*_10_10_31 em que o *gap* do VND (HANSEN; MLADENOVIć, 2001) foi igual ao *gap* do VND Aprimorado, sendo ambas soluções encontradas iguais a solução ótima, pois o *gap* foi 0. Nos demais conjuntos de instâncias o *gap* obtido pelo VND Aprimorado foi menor, e portanto, as soluções encontradas se aproximaram mais das ótimas ou encontraram soluções ótimas. Verifica-se ainda que o tempo de execução do VND Aprimorado foi muito superior ao do VND (HANSEN; MLADENOVIć, 2001), em consequência do refinamento aplicado a todos os vizinhos de todas as vizinhanças analisadas, o que é aceitável levando em consideração a qualidade da solução obtida, em que apenas 125 instâncias das 4092 instâncias testadas não obtiveram resultados ótimos.

Os maiores tempos de execução são encontrados nas instâncias maiores, ou seja, aquelas com uma maior quantidades de padrões. O alto tempo de execução do VND Aprimorado nessas instâncias é explicado pelo fato de que quanto maior o número de padrões, maior será a quantidade de vizinhanças a serem analisadas e consequentemente maior será a quantidade de soluções nas quais o refinamento deverá ser aplicado.

Tabela 6.1: Comparação entre os métodos considerando as instâncias agrupadas do *First Constraint Modeling Challenge* (SMITH, B. M., Gent, I. P. (eds), 2005)

Grupo	Solução Ótima	VND		VND Aprimorado	
		Tempo (ms)	*gap*	Tempo (ms)	*gap*
*problem*_10_10_380	2717	0,86	0,66	6,75	0,33
*problem*_10_20_246	1868	5,3	0,42	90,96	0,1
*problem*_15_15_346	4018	12,55	1,81	489,60	0,17
*problem*_15_30_84	1044	217,57	2,20	16580,44	0,0
*problem*_20_10_459	6913	4,25	1,39	138,17	0,38
*problem*_20_20_122	1994	118,3	1,85	10299,06	0,0
*problem*_30_10_500	11675	7,03	1,37	371,14	0,24
*problem*_30_15_162	3973	52,73	2,11	5556,52	0,08
*problem*_30_30_51	1345	2230,73	2,9	520810,41	0,07
*problem*_40_20_68	2322	381,94	2,33	86660,09	0,04
*wbo*_10_10_31	159	2,87	3,14	21,84	1,25
*wbo*_10_20_40	294	84,93	2,72	1685,53	0,34
*wbo*_10_30_24	168	413,96	4,17	11806	0,0
*wbo*_15_15_60	561	30,73	3,39	750,3	0,0
*wbo*_15_30_48	515	1228,04	4,66	69582,46	0,0
*wbo*_20_10_70	903	5,49	2,77	118,49	0,22
*wbo*_20_20_82	1072	190,76	5,5	10328,02	0,17
*wbo*_30_10_100	2005	7,13	2,59	268,04	0,1
*wbo*_30_15_120	2515	52,93	4,21	4066,17	0,08
*wbo*_30_30_120	2559	2247,99	5,67	323720,55	0,47
*wbop*_10_10_34	210	2,44	1,9	20,79	0,48
*wbop*_10_20_24	163	73,88	4,29	1569,46	0,0
*wbop*_10_30_22	162	510,27	4,32	14861,05	0,0
*wbop*_15_15_52	502	26,38	2,39	684,73	0,2
*wbop*_15_30_38	399	1062,11	5,76	65541,66	0,5
*wbop*_20_10_39	551	4,95	3,27	111,03	0,18
*wbop*_20_20_71	958	160,22	3,76	8846,62	0,0
*wbop*_30_10_39	869	6,18	0,92	229,82	0,0
*wbop*_30_15_55	1193	55,78	3,44	3636,45	0,08
*wbop*_30_30_100	2139	2136,76	5,42	293851,37	0,37
*wbp*_10_10_31	201	0,74	0,0	6,94	0,0
*wbp*_10_20_38	290	8,79	1,38	199,42	0,0
*wbp*_10_30_37	301	15,11	1,66	417,92	0,0
*wbp*_15_15_48	483	10,35	2,28	365,65	0,83
*wbp*_15_30_67	776	210,85	3,87	15504,91	0,0
*wbp*_20_10_35	505	3,97	1,98	121,83	0,4
*wbp*_20_20_63	847	86,94	4,37	7431,4	0,35
*wbp*_30_10_38	867	6,24	1,5	311,19	0,12
*wbp*_30_15_55	1229	43,36	3,82	4368,02	0,08
*wbp*_30_30_93	2015	1541,75	4,62	372035,53	0,15

Capítulo 7

Conclusões e Trabalhos Futuros

Foi proposta uma nova implementação do VND apresentado em (HANSEN; MLADENOVIć, 2001) para solução do problema de minimização de pilhas abertas, um problema de sequenciamento de padrões com aplicação industrial direta e também forte motivação teórica.

No experimento envolvendo um grande conjunto de instâncias da literatura, comparou-se o VND Aprimorado proposto neste trabalho com o VND proposto por (HANSEN; MLADENOVIć, 2001). Os resultados obtidos pela nova implementação foram muito superiores as da implementação original, o que já era esperado devido ao aumento da complexidade do método proposto, decorrente da aplicação de uma fase adicional, em relação ao método original.

O VND Aprimorado conseguiu melhores resultados que os obtidos pelo VND (HANSEN; MLADENOVIć, 2001). O tempo de execução, por outro lado, deixou a desejar, mas a qualidade da solução obtida, que apresentou *gaps* muito baixos, faz com que o VND Aprimorado seja interessante quando o objetivo sejam boas soluções independentemente do tempo gasto para obtê-las.

A redução da complexidade do refinamento aplicado ao método proposto, mantendo a eficácia ou melhorando, será objeto de estudo em trabalhos futuros, a fim de melhorar o tempo de execução do VND Aprimorado, principalmente para as instâncias maiores do problema. Experimentos com o VNS (HANSEN; MLADENOVIć, 2001) assim como propostas de implementação abordando esta metaheurística também serão objeto de futuros trabalhos.

Referências Bibliográficas

JOSé Carlos Becceneri.

ASHIKAGA, F. M.; SOMA, N. Y. A heuristic for the minimization of open stacks problem. **Pesquisa Operacional**, scielo, p. 439 – 450, 08 2009. ISSN 0101-7438.

BANDA, M. G. de la; STUCKEY, P. J. Dynamic programming to minimize the maximum number of open stacks. **INFORMS Journal on Computing**, v. 19, n. 4, p. 607–617, 2007.

BECCENERI, J. C. 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, 1999.

BECCENERI, J. C.; YANASSE, H. H.; SOMA, N. Y. A method for solving the minimization of the maximum number of open stacks problem within a cutting process. **Computers** & **Operations Research**, v. 31, n. 14, p. 2315 – 2332, 2004. ISSN 0305-0548.

CARVALHO, M. A. M. de; SOMA, N. Y. Métodos simplificados para o problema de minimização de pilhas abertas. **Gestão & Produção**, Universidade Federal de São Carlos, v. 18, n. 2, p. 299–310, 2011.

CHU, G.; STUCKEY, P. J. Minimizing the maximum number of open stacks by customer search. In: **Proceedings of the 15th International Conference on Principles and Practice of Constraint Programming**. Berlin, Heidelberg: Springer-Verlag, 2009. (CP'09), p. 242–257. ISBN 3-642-04243-0, 978-3-642-04243-0.

HANSEN, P.; MLADENOVIć, N. Variable neighborhood search: Principles and applications. **European Journal of Operational Research**, v. 130, n. 3, p. 449 – 467, 2001. ISSN 0377-2217.

LINHARES, A.; YANASSE, H. H. Connections between cutting-pattern sequencing, vlsi design, and flexible machines. **Comput. Oper. Res.**, Elsevier Science Ltd., Oxford, UK, UK, v. 29, n. 12, p. 1759–1772, out. 2002. ISSN 0305-0548.

MöHRING, R. H. **Graph Problems Related to Gate Matrix Layout and PLA Folding**. [S.l.]: Springer Vienna, 1990.

OPTSICOM Project, 2010. **Optimization of Complex Systems**. Disponível em: <http://www.optsicom.es/>. Acessado em 07 de Novembro de 2013.

SMITH, B. M., Gent, I. P. (eds). Proceedings of IJCAI'05 — *Constraint Modelling Challenge*. 2005.

YANASSE, H. On a pattern sequencing problem to minimize the maximum number of open stacks. **European Journal of Operational Research**, Elsevier, v. 100, n. 3, p. 454–463, ago. 1997. ISSN 03772217.

YANASSE, H. H.; SENNE, E. L. F. The minimization of open stacks problem: A review of some properties and their use in pre-processing operations. **European Journal of Operational Research**, v. 203, n. 3, p. 559 – 567, 2010. ISSN 0377-2217.

YUEN, B. J. Heuristics for sequencing cutting patterns. **European Journal of Operational Research**, v. 55, n. 2, p. 183 – 190, 1991. ISSN 0377-2217.

YUEN, B. J. Improved heuristics for sequencing cutting patterns. **European Journal of Operational Research**, v. 87, n. 1, p. 57 – 64, 1995. ISSN 0377-2217.