LUCAS GONÇALVES ABREU

Orientador: Marco Antonio Moreira de Carvalho

UMA HEURÍSTICA APLICADA AO PROBLEMA DE DETERMINAÇÃO DE LEIAUTE DE MATRIZES DE PORTAS

Ouro Preto

Agosto de 2016

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

UMA HEURÍSTICA APLICADA AO PROBLEMA DE DETERMINAÇÃO DE LEIAUTE DE MATRIZES DE PORTAS

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.

LUCAS GONÇALVES ABREU

Ouro Preto
Agosto de 2016

UNIVERSIDADE FEDERAL DE OURO PRETO

FOLHA DE APROVAÇÃO

Uma Heurística Aplicada ao problema de Determinação de Leiaute de Matrizes de Portas

LUCAS GONÇALVES ABREU

Monografia defendida e aprovada pela banca examinadora constituída por:

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

Dra. Andrea Gomes Campos Bianchi
Universidade Federal de Ouro Preto

Msc. Reinaldo Silva Fortes
Universidade Federal de Ouro Preto

Ouro Preto, Agosto de 2016

Resumo

Este trabalho aborda o problema de Determinação de Leiaute de Matrizes de Portas (*Gate Matrix Layout Problem*). Este problema combinatório consiste em encontrar uma disposição física das portas de um circuito eletrônico que minimize a quantidade de trilhas necessárias para a implementação do mesmo, diminuindo a área e o custo de fabricação relacionados. Métodos de solução deste problema possuem aplicabilidade prática direta nas áreas de Informática, Telecomunicações, Controles de Processos Industriais, Automação dos Serviços Bancários e Comerciais e também na fabricação de bens de consumo. A modelagem subjacente é bastante genérica e também pode ser utilizada para modelar com sucesso um grande conjunto de problemas de escalonamento em Pesquisa Operacional. Neste trabalho, o problema de Determinação de Leiaute de Matrizes de Portas é modelado como um problema de busca em grafos e são apresentadas implementações de buscas locais e uma heurística para sua solução. Para avaliar as abordagens apresentadas foram utilizados quatro conjuntos de instâncias disponíveis na literatura, sendo estas artificiais e reais. Na maioria dos testes realizados a heurística proposta apresentou resultados satisfatórios. Por outro lado, há a necessidade de novos estudos sobre os métodos apresentados, a fim de que os mesmos se tornem mais eficientes.

Abstract

This work addresses the Gate Matrix Layout Problem. This combinatorial problem consists in finding a physical arrangement of the gates of an electronic circuit that minimizes the amount of necessary tracks in the printed circuit, thus, reducing the area and cost of manufacturing the product. Solution methods for this problem have direct practical applicability in the areas of Information Technology, Telecommunications, Industrial Process Control, Banking and Commercial Automation and also in the production of consumer goods. The underlying modeling adopted is rather generic and it can also be used to successfully model a large set of scheduling problems in operations research. In this work the Gate Matrix Layout Problem is model as a search problem in graphs and implementations of four different local search procedures and a heuristic for its solution are present. To asses the quality of the implemented methods, four sets of real-world and artificial instances from the literature were considered. In most tests the heuristic presented satisfactory results. On the other hand, new studies about the presented methods are necessary, so that they become more efficient.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Acrônimos

GMLP *Gate Matrix Layout Problem*

IGAP *Interval Graph Augmentation Problem*

SCOOP *Sheet Cutting and Process Optimization for Enterprises*

OPTSICOM *Optimization of Complex Systems*

CMOS *Complementary metal-oxide-semiconductor*

BRKGA *Biased Random-Key Genetic Algorithm*

MOSP *Minimization of Open Stacks Problem*

Capítulo 1

Introdução

Os circuitos eletrônicos integrados são circuitos complexos fabricados em uma pequena fatia de silício, podendo possuir de 10 até 100.000 dispositivos ativos (transístores e diodos) Tooley (2006). Um exemplo de circuitos integrados digitais são as portas lógicas Enderlein (1994).

Para elaborar circuitos lógicos combinatórios pode ser utilizado um dispositivo lógico programável chamado de Matriz de Portas. Este dispositivo dispõe de uma dimensão de portas *AND* que podem ser ligadas a uma outra dimensão de portas *OR*, ambas programáveis.

Em uma representação por uma matriz lógica programável de um circuito, as *portas* correspondem a nós e um subconjunto de nós, conectados por um fio, corresponde a uma *rede*. As linhas verticais representam as portas e as horizontais as redes, que indicam como as portas serão conectadas umas às outras. No circuito físico, as redes são implementadas por meio de *trilhas* impressas.

A Figura 1.1(a) mostra um exemplo de matriz com 6 portas (enumeradas de 1 a 6) e 5 redes (nomeadas de A a E).

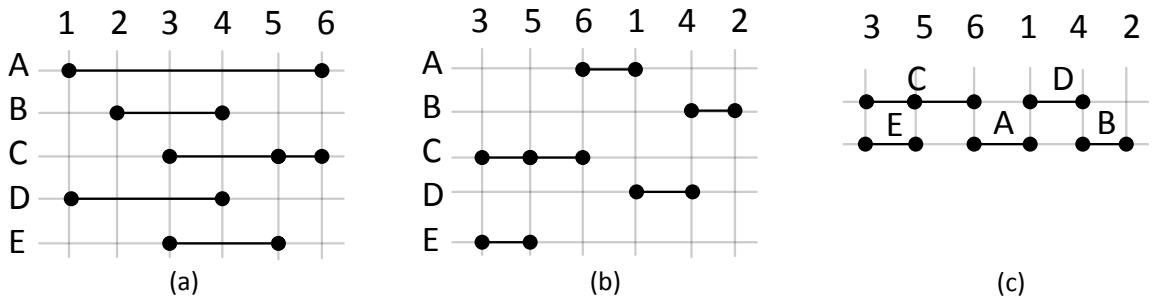

Figura 1.1: : Exemplo de matriz de portas (a) original, (b) permutada e (c) compactada.

A rede A tem conexão com as portas 1 e 6, a rede B tem conexão com 2 e 4 e assim por diante. Uma característica de produção de circuito eletrônico é que a permutação das portas não altera a lógica do circuito. O processo de permutação consiste em alterar a ordem das

portas de modo a diminuir o tamanho das redes. A permutação das portas pode diminuir o número de trilhas, diminuindo consequentemente a área necessária para implementar o circuito. Na referida figura, não é possível compactar a matriz de portas, pois os fios não podem se cruzar ou se sobrepor, exigindo uma área maior para implementação do circuito. A Figura 1.1(b) mostra o mesmo circuito com uma permutação diferente das portas, com esta permutação o tamanho das redes diminui, possibilitando o processo de compactação. A Figura 1.1(c) mostra a melhor compactação possível para a permutação apresentada, este processo consiste em realocar as redes de modo que utilize um numero menor de trilhas sem que ocorra sobreposição das redes. Esta compactação possibilita a produção de circuitos similares com menor área, podendo imprimir mais circuitos por wafer, diminuindo assim o custo de produção.

O Problema de determinação do Leiaute de Matriz de Portas *Gate Matrix Layout Problem* (GMLP) (Wing, Huang e Wang, 1985) é um problema que consiste em encontrar uma permutação de portas que minimize a quantidade de trilhas necessárias para a implementação do circuito, diminuindo a área e o custo de fabricação do mesmo.

1.1 Justificativa

Esta monografia estuda um problema de grande importância prática na indústria microeletrônica, o problema de Determinação de Leiaute de Matrizes de Portas, sendo relevante nas áreas de Informática, Telecomunicações, Controles de Processos Industriais, Automação dos Serviços Bancários e Comerciais e também na fabricação de bens de consumo, com aplicações práticas diretas na engenharia e indústria. O problema em questão é bastante genérico e também modela com sucesso um grande conjunto de problemas de escalonamento em Pesquisa Operacional.

Face à importância do problema, o mesmo foi escolhido como tema de grupos de cooperação acadêmica internacional como o *Sheet Cutting and Process Optimization for Enterprises* (SCOOP) e *Optimization of Complex Systems* (OPTSICOM), que reúnem universidades da Alemanha Espanha, Estados Unidos, Alemanha, Itália, México e Portugal, incluindo algumas 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.

Atualmente, a primeira fábrica de semicondutores (elemento primordial na indústria eletrônica e confecção de seus componentes) do hemisfério sul do globo está sediada no estado de Minas Gerais. Este projeto de pesquisa possui o potencial para gerar a inovação aplicável a este novo nicho industrial, em consonância com as políticas governamentais de investimento em pesquisa e desenvolvimento, e também reforçar os elos de cooperação da Universidade

Federal de Ouro Preto com a indústria.

O GMLP foi provado ser NP-difícil por uma redução do problema de aumento grafos de intervalo *Interval Graph Augmentation Problem* (IGAP) por Kashiwabara e Fujisawa (1979) (apud Linhares e Yanasse, 2002) e possui aplicações em diferentes áreas, bem como diferentes problemas de formulação equivalente (Linhares e Yanasse, 2002).

1.2 Objetivos

O objetivo deste trabalho, é elaborar uma heuristica consistente para ser utilizada nos problemas de leiaute de matrizes de portas que permita a obtenção rápida de soluções próximas da solução ótima.

1.2.1 Objetivos Específicos

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

2. Implementar um *software* para determinação do leiaute de matrizes de portas usando ferramentas de *Inteligência Computacional*, o que inclui a utilização de métodos heurísticos;

3. Avaliar o *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.

1.3 Organização do Trabalho

O restante deste texto encontra-se organizado em 6 Capítulos sendo eles: Revisão da Literatura, Fundamentação Teórica sobre o GMLP, Metodologia, Desenvolvimento, Experimentos Computacionais e Conclusões.

Capítulo 2

Revisão da Literatura

Wing et al. (1985) foi o primeiro trabalho a utilizar métodos heurísticos para resolver o GMLP. O GMLP foi modelado usando grafos de intervalos, onde as portas em comum de diferentes trilhas são consideradas interseções. Essas trilhas com portas em comum, para diminuir a área do circuito, podem ser implementadas como uma única trilha. Como testes, foi possível determinar leiautes para circuitos *Complementary metal-oxide-semiconductor* (CMOS) de 180 e 306 transistores em 9 e 28 segundos

Kashiwabara e Fujisawa (1979) (*apud* Linhares e Yanasse, 2002) provaram que o GMLP é NP-difícil reduzindo-o ao problema de aumento de grafos de intervalo (IGAP). Linhares e Yanasse (2002) mostraram diferentes áreas de aplicação do problema: Corte Modificado (*Modified Cutwidth*), Problema de Minimização de Pilhas Abertas, *Minimization of Open Stacks Problem* (MOSP), 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.*

Chen e Hu (1988) utilizaram o algoritmo *min-net-cut* com uma representação dinâmica das conexões do circuito e criaram um algoritmo para o GMLP que analisa a dualidade dos CMOS. O algoritmo tem complexidade $O(n \log n)$ onde n é o número de portas. Nos experimentos realizados, o *min-net-cut* obteve os melhores resultados nos 5 circuitos testados retirados da literatura (Hwang et al., 1987). Chen e Hu (1990b) apresentaram dois algoritmos, *GM-PLAN* e *GM-LEARN*, ambos baseados em paradigmas de inteligência artificial. *GM-LEARN* usa resultados gerados anteriormente obtidos a partir do *GM-PLAN* para melhorar a busca e utiliza uma função "aprender" para modificar as heurísticas utilizadas no *GM-PLAN* com base nos resultados anteriores. O *GM-PLAN* utiliza duas estratégias de inteligência artificial: planejamento hierárquico e políticas de meta-planejamento. Os resultados do *GM-LEARN* foram comparados com 16 circuitos retirados da literatura e, em 8 deles, o melhor resultado foi superado. Em outros 6 circuitos, os melhores resultados foram igualados e em apenas 2 circuitos, piores resultados foram obtidos. Já o *GM-PLAN* foi testado com 12 circuitos retirados da literatura e foram obtidos resultados para apenas 7 deles. O algoritmo conseguiu

melhor resultado para um circuito, e, nos outros 6 circuitos foram igualados os melhores resultados já conhecidos.

Shahookar et al. (1994) descrevem uma nova implementação de algoritmo genético para problemas de permutação utilizando o *beam search* para reduzir o espaço de busca. O algoritmo foi testado em 3 conjuntos de instâncias *benchmark* e o melhor resultado obtido foi em relação ao comprimento das redes.

Linhares et al. (1999) utilizaram uma heurística derivada da estatística mecânica: otimização microcanônica (*microcanonical optimization algorithm*). Os autores reportam que o algoritmo de otimização microcanônica foi capaz de atingir os resultados relatados por outras abordagens e superou, em 7 trilhas, as melhores soluções encontradas até então. Os resultados foram comparados com 5 heurísticas (recozimento simulado de Hong et al. (1989), métodos de construção unidirecional e bidirecional de Hong et al. (1989) e as heurísticas de inteligência artificial *GM-PLAN* e *GM-LEARN* de Chen e Hu (1990b)). Para a solução do problema de atribuição de redes para trilhas foi utilizado um algoritmo de tempo polinomial conhecido como *left-edge*.

Linhares (1999) utilizou uma estratégia de busca predatória, onde a área de busca é restringida e a cada iteração é determinada uma solução. Esse algoritmo é baseado no comportamento dos predadores na natureza, onde, ao encontrar uma presa, o predador procura em volta novas presas. Das 25 instâncias testadas, o método conseguiu igualar os resultados em 12 delas e em 4 dessas instâncias conseguiu melhorar os resultados da literatura, reportados em Heinbuch (1988). No restante das instâncias, os resultados obtidos por *GM-PLAN* e *GM-LEARN* (Chen e Hu (1990a)), foram atingidos.

Mendes e Linhares (2004) utilizaram algoritmos genético e memético e uma população hierarquicamente estruturada com base em uma árvore ternária completa. Nesta árvore foram colocados grupos de subpopulações sobrepostas, constituídas por um líder e três suportes. Os lideres contêm a melhor solução dentre todos os elementos do grupo, considerando o número de trilhas. Além disso, o algoritmo proposto apresenta uma busca local especialmente adaptada, que depende de uma vizinhança definida. Este trabalho apresentou resultados iguais aos da otimização microcanônica e superou todas as abordagens metaheurísticas e heurísticas anteriores da literatura.

Oliveira e Lorena (2002) utilizaram um algoritmo genético construtivo. Este algoritmo tem características novas em comparação com um algoritmo genético tradicional. Este inclui uma população de tamanho dinâmico composto por esquemas e estruturas, nas quais há a possibilidade de usar heurísticas para representá-las e definir funções de aptidão. Os autores modelaram o GMLP como o problema de otimização bi-objetivo(reduzir o numero de trilhas e diminuir o tamanhos dos fios). Os problemas utilizados foram os maiores encontrados na literatura até então. O algoritmo alcançou os melhores resultados da literatura, obtidos pela otimização microcanônica e busca predatória, sendo considerado pelos autores mais robusto

que esta.

Giovanni et al. (2013) considerou não só o GMLP, mas também o problema de minimização de custo de conexões do GMLP, levando em consideração o comprimento dos fios. Foram utilizados dois algoritmos, o primeiro uma heurística baseada em uma abordagem genética em conjunto com uma definição composta e dinâmica da função de *fitness*. O segundo, um algoritmo *branch-and-cut* baseado em uma nova formulação de programação linear inteira. Em casos pequenos, o algoritmo *branch-and-cut* comprova os melhores resultados gerados pela abordagem genética. Os autores utilizaram 330 instâncias de diferentes conjuntos de dados. Em 11 instâncias, os métodos propostos foram melhores que os resultados já conhecidos.

Gonçalves et al. (2014) apresentaram um algoritmo genético de chaves aleatórias viciadas (*Biased Random-Key Genetic Algorithm* (BRKGA)) para resolver o MOSP. Esta abordagem é divida em quatro fases: Representação e Decodificação do Cromossomo; Construção da Solução; Ajuste do Cromossomo e Avaliação. Na primeira fase, a solução do problema possui duas representações, direta e indireta: a representação direta é a padrão do problema, enquanto a indireta é um vetor de chaves randômicas que na fase de decodificação é ordenado de maneira crescente. A fase de construção da solução é composta por duas etapas que são repetidas ate que todos os padrões sejam sequenciados: A primeira etapa seleciona o padrão a ser inserido e a segunda seleciona a posição na solução parcial para inserir o padrão. A busca local gera soluções que não obedecem às regras impostas pelo cromossomo, e para adequar a solução é necessário utilizar uma fase de ajuste. A heurística ajusta a ordem dos genes de acordo com a posição de cada padrão na solução final, de modo a reduzir o número de gerações necessárias para obtenção de melhores valores. A fase final utiliza uma função *fitness* específica do MOSP. Os autores utilizaram 6141 instâncias disponíveis na literatura, divididas nas classes: *Harvey*; *Simonis*; *Shaw*; *Miller e Wilson* (Smith e Gent, 2005); *Faggioli e Bentivoglio* (Faggioli e Bentivoglio, 1998); *VLSI* (Chen e Hu, 1990b); *SCOOP* (*Disponível em http://www.scoopproject.net*); *Harder150* (De Giovanni et al., 2013) e *Becceneri* (Becceneri, 1999). Os experimentos realizados mostraram que esta nova proposta igualou ou superou todas as outras abordagens da literatura, obtendo os melhores resultados conhecidos.

Capítulo 3

Fundamentação Teórica

Este capítulo apresenta a fundamentação teórica do GMLP. Nas próximas seções são apresentadas uma definição formal do problema GMLP, a definição do método de pré-processamento, representação por grafos, busca local e por fim a definição do método de busca local iterativa.

3.1 O Problema de Determinação de Leiaute de Matrizes de Portas

Formalmente, uma instância do GMLP contendo m redes e n portas pode ser descrita por uma matriz P binária $m \times n$, em que p_{ij}=1 se a rede i incluir a porta j e p_{ij}=0, caso contrário. A Figura 3.1 ilustra este formato, em que 5 redes (nomeadas de A a E) e 6 portas (enumeradas de 1 a 6) são representadas.

	1	2	3	4	5	6
A	1	0	0	0	0	1
B	0	1	0	1	0	0
C	0	0	1	0	1	1
D	1	0	0	1	0	0
E	0	0	1	0	1	0

Figura 3.1: : Exemplo de matriz de binária original

Uma solução para o GMLP consiste em uma permutação π das n colunas da matriz P. Ao alterarmos a posição relativa das portas, é necessário reservar espaço para os fios, porque os mesmos não podem se cruzar em uma mesma trilha (vide Figura 3.2).

Desta forma, uma solução π para este problema induz uma matriz permutação Q^{π} que contém as colunas de P na ordem determinada por π. A matriz Q possui a propriedade dos 1s consecutivos, de maneira que q_{ij}^{π}=1 se e somente se o fio da rede i incluir ou cruzar

a porta j. Desta forma, todos os elementos da matriz Q que estão entre entradas de valor 1 originalmente também são considerados como tendo o valor 1. A Equação 3.1 descreve formalmente preenchimento da matriz Q.

	1	2	3	4	5	6
A	1	**1**	**1**	**1**	**1**	1
B	0	1	**1**	1	0	0
C	0	0	1	**1**	1	1
D	1	**1**	**1**	1	0	0
E	0	0	1	**1**	1	0

Figura 3.2: : Exemplo de matriz de binária com 1s consecutivos

$$q_{ij}^{\pi} = \begin{cases} 1 & \text{se } \exists x, \exists y \ \mid \pi(x) \leqslant j \leqslant \pi(y) \text{ e } p_{ix} = p_{iy} = 1, \\ 0 & \text{caso contrário} \end{cases} \tag{3.1}$$

Uma solução para o GMLP é avaliada maximizando a soma dos 1s das colunas da matriz Q de acordo com a Equação 3.2. Neste problema as colunas de maior soma são consideradas o gargalo do problema, tendo em vista que o numero da soma define o numero de trilhas necessárias.

$$Z_{GMLP}^{\pi}(Q^{\pi}) = \max_{j \in \{1,\ldots,n\}} \sum_{i=1}^{m} q_{ij}^{\pi} \tag{3.2}$$

A Figura 3.3 mostra uma segunda permutação determinada como solução. O valor da soma das colunas mostra o número mínimo de trilhas que podemos usar, já que não podemos ter duas redes usando a mesma porta na mesma trilha, pois isso mudaria a lógica do circuito. A soma das colunas da Figura 3.3 é igual a 2, mostrando que o circuito pode ser implementado usando apenas 2 trilhas. Uma outra permutação possível para as portas do exemplo seria π =[3, 5, 1, 6, 4, 2]. Esta permutação, gera uma matriz compactada com três trilhas. Daí surge a necessidade de utilizar métodos computacionais para determinar uma permutação que leve a um leiaute otimizado, tendo em vista que o número de possíveis permutações é $n!$.

	3	5	6	1	4	2
A	0	0	1	1	0	0
B	0	0	0	0	1	1
C	1	1	1	0	0	0
D	0	0	0	1	1	0
E	1	1	0	0	0	0

Figura 3.3: : Exemplo de matriz de binária permutada.

Por fim, a função objetivo do GMLP é determinar a permutação π das colunas de P tal

que o número máximo de 1s consecutivos em uma coluna seja minimizado, vide Equação 3.3.

$$\min_{\pi \in \Pi} Z^{\pi}_{GMLP}(P) \tag{3.3}$$

3.2 Pré-processamento dos Dados de Entrada

Métodos de pré-processamento dos dados de entrada de problemas combinatórios são utilizados comumente na tentativa de reduzir as dimensões dos problemas pela identificação de informações redundantes ou identificação de estruturas especiais.

Adaptando o conceito introduzido por Yanasse e Senne (2010), uma porta dominada é aquela que compartilha todas as suas redes com outra(s) porta(s) do problema. Em outras palavras, seja $R(i)$ uma função que retorna o conjunto de redes que utilizam a porta i, temos que uma porta i é dominada por uma porta j se $R(i) \subseteq R(j)$. Portas dominadas podem ser retiradas do problema, uma vez que posteriormente podem ser sequenciadas imediatamente após a sua porta dominante, sem o aumento do número de trilhas. Para identificarmos a dominância entre portas, é necessário comparar as redes de todos os pares de portas, exigindo complexidade de $O(n^2m)$, em que n denota o número de portas e m o número de redes.

A Figura 3.4 apresenta a matriz original utilizada como exemplo, onde $R(1) = \{A, B, D\}$, $R(2) = \{B, D\}$ e $R(3) = \{D\}$. Desta forma, a porta 1 domina as portas 2 e 3, que podem então ser removidas do processo de solução do problema e adicionadas posteriormente na solução após a porta 1.

	1	2	3	4	5
A	1	0	0	1	0
B	1	1	0	1	0
C	0	0	0	1	1
D	1	1	1	0	1
E	0	0	0	0	1

Figura 3.4: : Matriz Original.

A Figura 3.5 apresenta a matriz resultante após o pré-processamento, sem as colunas que representam as portas dominadas.

	1	4	5
A	1	1	0
B	1	1	0
C	0	1	1
D	1	0	1
E	0	0	1

Figura 3.5: : Matriz resultante do pré-processamento.

3.3 Representação por Grafos

Yanasse (1997) mostrou que qualquer instância MOSP (um problema de formulação equivalente ao GMLP) pode ser modelada como um problema em grafos. Nesta modelagem, cada vértice corresponde a uma rede, havendo uma aresta entre dois vértices se e somente se existir pelo menos uma porta compartilhada entre as duas redes correspondentes.

Esta representação permite que durante a solução do problema seja levada em consideração a relação entre portas e redes e também das redes entre si.

A Figura 3.6 apresenta o grafo associado ao exemplo anterior.

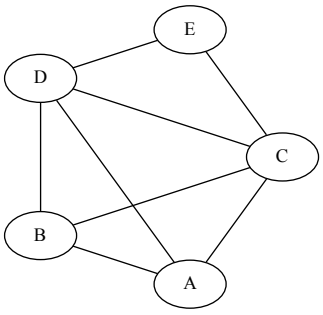

Figura 3.6: : Exemplo de grafo gerado a partir do exemplo anterior.

No exemplo anterior temos que as redes A, B e D compartilham a porta 1, portanto, há uma aresta entre cada par de vértices associados a estas portas induzindo um clique de tamanho 3 no grafo ({A, B, C}). Analogamente, dois outros cliques de tamanho 3 são induzidos pelo compartilhamento da porta 4 pelas redes A, B e C e pelo compartilhamento da porta 5 entre as redes C, D e E. Outros cliques podem ser induzidos pela união dos cliques anteriores, como é o caso do clique {A, B, C, D}. Os grafos associados a instâncias do GMLP são, portanto,

formados pela união dos cliques induzidos pelas portas. Arestas paralelas e laços não são considerados nesta representação.

É interessante notar que diferentes instâncias GMLP podem possuir grafos associados idênticos. A remoção de colunas no pré-processamento apresentado não implica na alteração do grafo resultante.

3.4 Busca em Largura

Cormen et al. (2001) define a *Busca em Largura* (ou BFS, do inglês *Breadth-First Search*) do seguinte modo: dado um grafo e um vértice de origem, a busca em largura explora sistematicamente as arestas do grafo para "descobrir" cada vértice que pode ser alcançado a partir da origem. A busca explora toda a vizinhança de um vértice por vez e utiliza a estrutura de fila como guia: cada novo vértice explorado é adicionado ao final da fila. Ao final, a BFS retorna uma lista que contém a sequência dos vértices na ordem em que foram explorados. A complexidade computacional da BFS é $O(V+E)$ onde V é o número de vértices e E o número de arestas.

3.5 Busca Local Iterativa

A versão clássica da busca local é um processo de melhoria iterativo. O processo inicia a partir de uma solução inicial, e buscas são realizadas em sua vizinhança por uma solução de menor custo. A solução inicial pode ser obtida a partir de uma heurística gulosa, que gera uma solução visando, a cada passo o maximo benefício. A partir de uma solução inicial s, associa-se uma vizinhança N de s. Cada solução s' da vizinhança $N(s)$ é alcançada a partir de s por uma operação chamada *movimento*, e a solução s' é chamada de solução *vizinha*. Seleciona-se uma solução vizinha que seja melhor que a solução corrente, e a busca prossegue iterativamente até que não seja encontrada uma solução melhor que a corrente, ou seja, até que seja atingido um ótimo local.

O método Busca Local Iterativa (*Iterated Local Search* – ILS) constrói iterativamente uma sequência de soluções. O ILS emprega diferentes procedimentos de busca local de modo a gerar novas soluções correntes, correspondentes aos ótimos locais de cada vizinhança induzida. Também são utilizados mecanismos de perturbação de soluções, que consistem alterar os elementos da solução, e, desta forma, mudar o foco da busca local para outra parte do espaço de busca, (Glover et al., 2003). Na Busca Local Iterativa, quando uma solução melhor é encontrada na vizinhança, o foco da busca passa a ser esta solução melhor encontrada e sua vizinhança é examinada antes de ser realizada a perturbação.

O Algoritmo 1 apresenta a Busca Local Iterativa, onde *Sol_ Corr* é a solução analisada pela busca, sendo a base para a formação da vizinhança *N(Sol_ Corr)*. A variável *Sol* guarda a *Sol_ Corr* e a variável Sol^* guarda a melhor solução visitada pela busca. Já a função $\sigma(Sol)$

executa uma transformação (ou tambem denominada *perturbação*) sobre *Sol*. A constante K define a magnitude da pertubação.

Algoritmo 1: Pseudocódigo da Busca Local Iterativa

```
1  Entrada: K, σ
2  Inicialização: Gerar Solução inicial
3  Sol_Corr ← Sol_inicial, Sol* ← Sol_Corr
4  enquanto regra de parada não for verdadeira faça
5  |  Determine a melhor solução t ∈ N(Sol_Corr)
6  |  se f(f) < f(Sol_Corr) então
7  |  |  Sol_Corr ← t;
8  |  fim
9  |  se f(Sol_Corr) < f(Sol*) então
10 |  |  Sol* ← Sol_Corr;
11 |  fim
12 |  Sol_Corr ← k σ (Sol){atribuir a Sol_Corr k ϵ operações sobre Sol}
13 fim
```

Capítulo 4

Metodologia

Neste capítulo são descritas as etapas do método proposto para a solução do problema de determinação do leiaute de matrizes de portas. Foi aplicado uma BFS no grafo gerado, com o resultado da BFS as portas do problema são sequenciadas, posteriormente foi aplicada uma busca local iterativa, onde são aplicados 4 métodos de busca local e um metodo de perturbação.

4.1 Busca em Largura Aplicada ao GMLP

Especificamente na heurística proposta, o vértice de origem da BFS é aquele de menor grau. Adicionalmente, durante a BFS os vértices de menor grau possuem prioridade de exploração em cada vizinhança. Empates são decididos em favor do vértice de menor índice lexicográfico.

O critério de priorização de vértices de menor grau, introduzido por Carvalho e Soma (2015), é uma regra heurística que possui o intuito de gerar uma lista das redes em que redes relacionadas entre si apareçam contiguamente e em que redes relacionadas a várias outras sejam sequenciadas por último. A partir desta lista de redes obtêm-se a lista de portas como descrito na próxima seção.

A Figura 4.1 ilustra (em sentido horário) um exemplo de execução da BFS no mesmo grafo dos exemplos anteriores. A BFS se inicia pelo vértice E, de grau dois, e explora todos os seus vértices vizinhos. Como os vizinhos de E possuem o mesmo grau o desempate é resolvido a favor do vértice C. Na sequência da busca os vértices D, A e B são explorados nesta ordem. A BFS gera como saída uma lista L com a ordem em que os vértices foram visitados, no caso $L = \{E, C, D, A, B\}$.

4.2 Sequenciamento das Portas

Conforme mencionado anteriormente a BFS retorna apenas a lista de redes L. Esta lista é então examinada para que o sequenciamento das portas, a solução do problema, seja obtida.

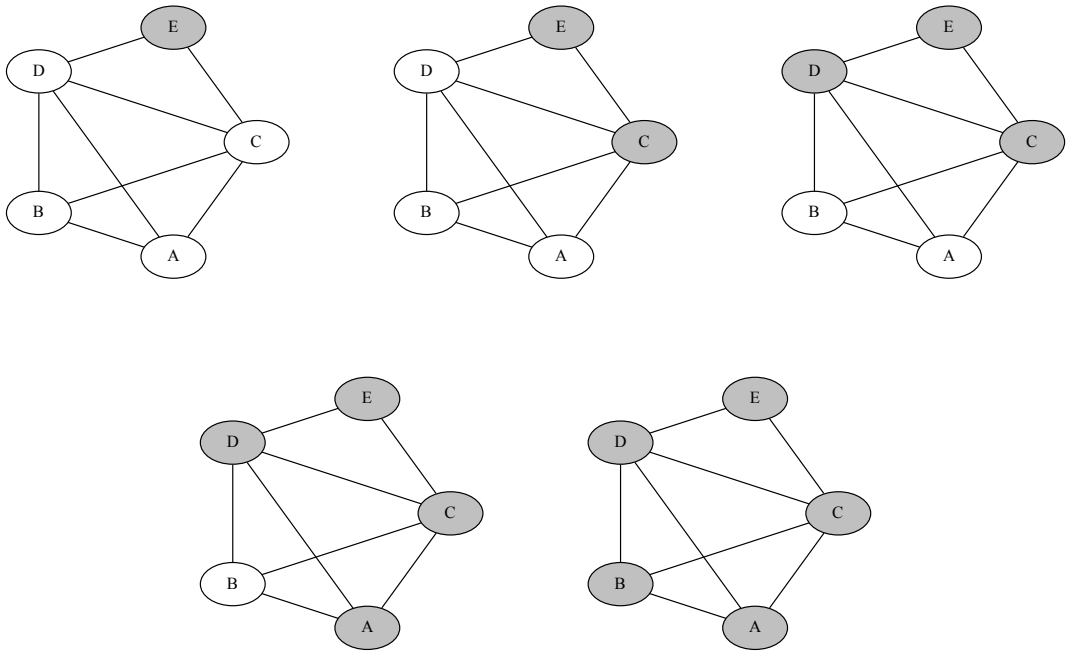

Figura 4.1: : Exemplo de uma BFS aplicada no grafo do exemplo anterior, em sentido horário.

O método proposto por Becceneri et al. (2004) percorre o conteúdo da lista L examinando-a e, quando todas as redes que utilizam uma determinada porta tiverem sido examinadas a porta correspondente é sequenciada. Quando uma porta dominante é sequenciada, as portas dominadas por ela são inseridas imediatamente após. Este processo é executado até que todas as portas tenham sido sequenciadas. Este procedimento possui complexidade limitada por $O(nm)$, em que n denota o número de portas e m o número de redes.

A Figura 4.2 ilustra o sequenciamento das portas dos exemplos anteriores.

	1	4	5
E	0	0	1
C	0	1	1
D	1	0	1

	1	4	5
E	0	0	1
C	0	1	1
D	1	0	1
A	1	1	0
B	1	1	0

Figura 4.2: : Ilustração do sequênciamento das portas.

Considerando a lista $L = \{E, C, D, A, B\}$, gerada anteriormente pela BFS, ao examinarmos as portas E, C e D (à esquerda na Figura 4.2) todas as redes que utilizam a porta 5 foram percorridas, deste modo, podemos sequênciar a porta 5. Posteriormente, ao examinarmos as redes A e B podemos sequenciar a porta 1 e todas as portas dominadas por ela (portas 1 e

3), bem como a porta 4. A sequência das portas na solução obtida é, portanto, [5, 1, 2, 3, 4], cujo número máximo de trilhas é 4.

4.3 Busca Local Iterativa

As próximas seções descrevem quatro métodos de busca local, o método de perturbação e a implementação de ILS proposta. Nesta seção apresentamos o desenvolvimento da busca local iterativa.

4.3.1 Buscas Locais 1 e 2

O processo de busca local 1 se inicia pela identificação dos gargalos e as redes relacionadas aos uns consecutivos neste gargalo. Posteriormente, identifica-se quais portas são contidas nestas redes e tenta-se realocá-las em uma estratégia *best insertion* restrita. Caso a porta se encontre em uma posição à direta do gargalo, esta é realocada em cada posição à esquerda do mesmo, de modo a verificar a possibilidade de melhora na solução. Caso ocorra tal melhoria, a solução corrente é alterada de acordo. De maneira análoga, caso a porta se encontre à esquerda do gargalo, o processo é realizado com a realocação nas posições da direita.

Considerando uma solução hipotética $s = [1, 2, 3, 4, 5, 6, 7]$, em que um dos gargalos seria a porta 4 e as portas correspondentes às redes envolvidas seriam 1, 3, 6 e 7. A cada passo é selecionada uma destas portas e a mesma é inserida em todas as posições do lado oposto ao gargalo. Selecionando a porta 1 uma possível solução seria $s = [2, 3, 4, 1, 5, 6, 7]$. Caso esta solução seja melhor do que a corrente, a esta é alterada e novos gargalos devem ser encontrados.

O processo da busca local 2 é similar ao processo de busca local 1, porem a solução é alterada mesmo quando houver soluções com o mesmo custo da solução corrente. Isto permite que, mesmo sem melhoria da solução, haja movimentação dentro do espaço de busca.

4.3.2 Busca Local 3

A busca local 3 tem como passo inicial determinar os gargalos, as redes relacionadas e suas portas correspondentes, assim como nas buscas locais anteriores. Cada porta é realocada em todas as posições possíveis, como no método *best insertion*, independente da posição do gargalo, de modo a tentar obter melhores soluções. Caso ocorra melhora, a solução corrente é atualizada.

Utilizando o mesmo exemplo da busca local 1 e 2 teriamos: Considerando a solução $s = [1, 2, 3, 4, 5, 6, 7]$, em que o gargalo seria a porta 4, e as portas relacionadas seriam 1, 3, 6 e 7. A cada passo é selecionada uma porta e a mesma é inserida em todas as posições da solução. Selecionando a porta 1 algumas das soluções possíveis seriam $s = [2, 1, 3, 4, 5, 6, 7]$,

$s = [2, 3, 4, 5, 1, 6, 7]$, entre outras. Se alguma dessas soluções for melhor que a solução corrente, então a mesma é atualizada.

4.3.3 Busca Local 4

Para a busca local 4, somente os gargalos são determinados. Cada gargalo identificado é realocado em todas as posições, como no método *best insertion*, de modo a tentar obter melhores soluções. Caso ocorra melhora, a solução corrente é atualizada.

Utilizando o mesmo exemplo da busca locais anteriores, temos a solução $s = [1, 2, 3, 4, 5, 6, 7]$ e o gargalo sendo a porta 4. Esta porta é inserida em todas as posições da solução, de $s = [4, 1, 2, 3, 5, 6, 7]$ até $s = [1, 2, 3, 5, 6, 7, 4]$. Se alguma dessas soluções for melhor que a solução corrente, então a mesma é atualizada.

4.3.4 Perturbação

O processo de perturbação consiste em trocar de posição pares aleatórios de portas em uma solução. O processo se inicia com a geração de um vetor de pares de portas distintas com todas as permutações possíveis (1,2), (1,3), (1,4), ...,(n-1,n). A magnitude de perturbação é de $\lceil m \times 0,25 \rceil$, sendo que no mínimo uma troca será feita. Foi utilizada a heurística *2-opt* proposta por Croes (1958), em que dois elementos são eliminados da solução e inseridos novamente de maneira cruzada.

4.3.5 Visão Geral

O Algoritmo 2 apresenta a Busca Local Iterativa, proposta, em que Sol_Corr é a solução atual analisada e atualizada (se adequada) pelas quatro buscas locais. A variável Sol^* armazena a melhor solução visitada pela busca, a função $\sigma(Sol\_Corr)$ realiza a perturbação sobre Sol_Corr e a constante K define a magnitude da perturbação.

A função F conta o número máximo de trilhas da solução gerada, se o número de trilhas for igual ao *lower bound* ou o número de iterações da ILS atingir 100, o processo é finalizado.

Algoritmo 2: Pseudocódigo da Busca Local Iterativa proposta

Entrada: K, σ.
Inicialização: Gerar Solução inicial
$Sol\_Corr \leftarrow Sol\_inicial$;
$Sol^* \leftarrow Sol\_Corr$;
enquanto *critérios de parada não forem atendidos* **faça**
    Perturbação:
    $Sol\_Corr \leftarrow \sigma\,(Sol\_Corr, k)$;// `realizar K perturbações` σ `sobre` Sol_Corr
    $Sol\_Corr \leftarrow Busca\_Local\_2(Sol\_Corr)$;
    $Sol\_Corr \leftarrow Busca\_Local\_3(Sol\_Corr)$;
    $Sol\_Corr \leftarrow Busca\_Local\_4(Sol\_Corr)$;
    $Sol\_Corr \leftarrow Busca\_Local\_1(Sol\_Corr)$;
    se $f(Sol\_Corr) < f(Sol^*)$ **então**
        $Sol^* \leftarrow Sol\_Corr$
    fim
fim

Capítulo 5

Experimentos Computacionais

Os experimentos computacionais, foram realizados em um computador com processador Intel core i3 2.3GHZ com 4 GB de RAM, usando Windows 10. O código foi escrito em C++.

Todas as instâncias foram rodadas 20 vezes cada, pois a heurística proposta utiliza parâmetros aleatórios.

As tabelas das seções a seguir apresentam para cada instância/grupo de instâncias, o nome, o número de trilhas (m), o número de portas (n), o número de instâncias (i), os resultados ótimos (OPT), e, para cada método proposto, o melhor resultado obtido (S^*) – expresso em número de trilhas, o tempo (T) de execução – expresso em segundos e a distância percentual entre os resultados obtidos e os valores ótimos (*gap*), calculada conforme definido na Equação 5.1.

$$100 \times (S^* - OPT)/OPT \tag{5.1}$$

Os resultados ótimos usados como referência são retirados de Gonçalves et al. (2014), detentor dos melhores resultados da literatura.

Tabelas adicionais apresentam o *ranking* ordinal médio de todos os métodos individualmente, por conjunto de instâncias. O *ranking* ordinal é obtido pela análise dos resultados obtidos para cada um das instâncias consideradas, indicando a ordem dos métodos de acordo com seu desempenho. Desta forma, ao método de melhor desempenho é atribuído o valor 1, ao segundo o valor 2 e assim por diante. No caso de empates o mesmo valor é atribuído a aos métodos.

Nas análises, os métodos de busca local implementados são enumerados de *BL1* a *BL4*.

5.1 Instâncias VLSI (Chen e Hu, 1988)

Este conjunto possui um total de 25 instâncias oriundas do projeto de circuitos *VLSI* reais de empresas asiáticas introduzidas por Chen e Hu (1988). Estas instâncias possuem matrizes

de dimensões que variam entre 5×7 até 202×141. A Tabela 5.1 apresenta os resultados obtidos neste conjunto de instâncias por cada busca local isoladamente.

Para o conjunto apresentado, o método *BFS* gera soluções iniciais equivalentes às melhores da literatura para 17 instâncias. O *gap* médio deste método é de 6,82%. Os métodos *BL2* e *BL3* combinados com a *BFS*, conseguem alcançar a melhor solução da literatura em 18 instâncias do conjunto.

A Tabela 5.2 apresenta os resultados obtidos pelo ILS para o mesmo conjunto de instâncias.

O *gap* médio da *ILS* é 4,71%, isto indica que a heurística proposta apresentou resultados satisfatórios para as instâncias VLSI reais. O tempo de execução médio é 137,8 segundos, sendo fortemente influenciado pelas instâncias *w3*, *w4* e *x9*, as maiores instâncias do conjunto. Na instância *w3* a heurística diminuiu o valor da função objetivo obtida pela *BFS* em cerca de 16%.

Tabela 5.1: Resultados para instâncias VLSI reais.

	OPT	BFS			BL 1			BL 2			BL 3			BL 4		
		S^*	T	*gap*	S^*	T	*gap*	S^*	T	*gap*	S^*	T	*gap*	S^*	T	*gap*
v4000.txt	5	6	0,0	20,00%	6	0,5	20,00%	6	0,2	20,00%	6	0,0	20,00%	6	0,0	20,00%
v4050.txt	5	6	0,0	20,00%	6	1,7	20,00%	6	0,6	20,00%	6	0,2	20,00%	6	0,0	20,00%
v4090.txt	10	**10**	0,0	0,00%	**10**	0,4	0,00%	**10**	0,3	0,00%	**10**	0,0	0,00%	**10**	0,0	0,00%
v4470.txt	9	12	0,0	33,33%	12	28,8	33,33%	12	4,3	33,33%	12	4,1	33,33%	12	0,1	33,33%
vc1.txt	9	**9**	0,0	0,00%	**9**	0,0	0,00%	**9**	0,0	0,00%	**9**	0,0	0,00%	**9**	0,0	0,00%
vl.txt	3	3	0,0	0,00%	3	0,0	0,00%	3	0,0	0,00%	3	0,0	0,00%	3	0,0	0,00%
vw1.txt	4	**4**	0,0	0,00%	**4**	0,0	0,00%	**4**	0,0	0,00%	**4**	0,0	0,00%	**4**	0,0	0,00%
vw2.txt	5	**5**	0,0	0,00%	**5**	0,9	0,00%	**5**	0,1	0,00%	**5**	0,1	0,00%	**5**	0,0	0,00%
w1.txt	4	5	0,0	25,00%	5	0,2	25,00%	5	0,1	25,00%	**4**	0,0	0,00%	**4**	0,0	0,00%
w2.txt	14	17	0,0	21,43%	15	6,6	7,14%	16	1,6	14,29%	15	0,7	7,14%	16	0,0	14,29%
w3.txt	18	25	0,0	38,89%	25	34,4	38,89%	22	5,7	22,22%	22	19,0	22,22%	25	0,4	38,89%
w4.txt	27	33	0,1	22,22%	33	386,1	22,22%	32	16,9	18,52%	32	111,2	18,52%	33	1,5	22,22%
wan.txt	6	**6**	0,0	0,00%	**6**	0,0	0,00%	**6**	0,0	0,00%	**6**	0,0	0,00%	**6**	0,0	0,00%
wli.txt	4	**4**	0,0	0,00%	**4**	0,0	0,00%	**4**	0,0	0,00%	**4**	0,0	0,00%	**4**	0,0	0,00%
wsn.txt	8	**8**	0,0	0,00%	**8**	1,1	0,00%	**8**	0,6	0,00%	**8**	0,1	0,00%	**8**	0,0	0,00%
x0.txt	11	12	0,0	9,09%	12	25,1	9,09%	12	6,2	9,09%	12	3,9	9,09%	12	0,2	9,09%
x1.txt	5	**5**	0,0	0,00%	**5**	0,7	0,00%	**5**	0,1	0,00%	**5**	0,1	0,00%	**5**	0,0	0,00%
x2.txt	6	**6**	0,0	0,00%	**6**	0,1	0,00%	**6**	0,2	0,00%	**6**	0,0	0,00%	**6**	0,0	0,00%
x3.txt	7	**7**	0,0	0,00%	**7**	0,1	0,00%	**7**	0,6	0,00%	**7**	0,0	0,00%	**7**	0,0	0,00%
x4.txt	2	**2**	0,0	0,00%	**2**	0,0	0,00%	**2**	0,0	0,00%	**2**	0,0	0,00%	**2**	0,0	0,00%
x5.txt	2	**2**	0,0	0,00%	**2**	0,0	0,00%	**2**	0,0	0,00%	**2**	0,0	0,00%	**2**	0,0	0,00%
x6.txt	2	**2**	0,0	0,00%	**2**	0,0	0,00%	**2**	0,0	0,00%	**2**	0,0	0,00%	**2**	0,0	0,00%
x7.txt	4	**4**	0,0	0,00%	**4**	2,7	0,00%	**4**	0,2	0,00%	**4**	0,2	0,00%	**4**	0,0	0,00%
x8.txt	4	4	0,0	0,00%	4	22,8	0,00%	4	1,0	0,00%	4	2,0	0,00%	4	0,2	0,00%
x9.txt	4	4	0,0	0,00%	4	196,7	0,00%	4	3,5	0,00%	4	18,5	0,00%	4	1,1	0,00%

Tabela 5.2: Resultados da ILS para instâncias VLSI reais.

	OPT	ILS		
		S^*	T	gap
v4000.txt	5	6	2,7	20,00%
v4050.txt	5	6	21,7	20,00%
v4090.txt	10	**10**	4,8	0,00%
v4470.txt	9	12	126,3	33,33%
vc1.txt	9	**9**	0,1	0,00%
vl.txt	3	**3**	0,0	0,00%
vw1.txt	4	**4**	0,0	0,00%
vw2.txt	5	**5**	4,7	0,00%
w1.txt	4	**4**	1,1	0,00%
w2.txt	14	**14**	20,9	0,00%
w3.txt	18	22	663,5	22,22%
w4.txt	27	33	1108,3	22,22%
wan.txt	6	**6**	0,0	0,00%
wli.txt	4	**4**	0,0	0,00%
wsn.txt	8	**8**	8,5	0,00%
x0.txt	11	**11**	563,7	0,00%
x1.txt	5	**5**	13,2	0,00%
x2.txt	6	**6**	2,6	0,00%
x3.txt	7	**7**	7,1	0,00%
x4.txt	2	**2**	0,0	0,00%
x5.txt	2	**2**	0,1	0,00%
x6.txt	2	**2**	0,1	0,00%
x7.txt	4	**4**	8,4	0,00%
x8.txt	4	**4**	72,1	0,00%
x9.txt	4	**4**	816,9	0,00%

5.2 Instâncias SCOOP

Este conjunto, fornecido pelo SCOOP *Consortium* [1], contém 187 instâncias MOSP reais. Porém, a maioria destas instâncias são muito pequenas (por exemplo, 2 linhas e colunas), podendo ser descartadas. Deste conjunto, 24 instâncias com dimensões significativas, variando de 10 linhas e colunas a 49 linhas e 134 colunas, foram selecionadas para serem usadas nos experimentos. A Tabela 5.3 apresenta os valores acumulados para solução.

Para este conjunto de instâncias, a *BFS* proposta obteve um resultado satisfatório com um *gap* médio de 7,89%. Todas as buscas locais obtiveram bons resultados, com o tempo de execução médio de 0,65 segundos para este conjunto de instâncias. Porém, destaca-se a a busca local *BL3*, que obteve melhores resultados, com um *gap* médio de 3,78%.

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

A Tabela 5.4 apresenta os resultados obtidos pelo ILS para o mesmo conjunto de instâncias.

A heurística proposta, obteve bons resultados com um *gap* médio de 3,3%. Para o grupo de instâncias B (as 12 últimas instâncias), a heurística igualou os melhores resultados da literatura para 11 instâncias. O tempo de execução médio para este conjunto foi de 13,7 segundos.

Tabela 5.3: Resultado para instâncias SCOOP.

Instância	m	n	OPT	BFS			BL1			BL2			BL3			BL4		
				S^*	T	gap	S^*	T	gap	S^*	T	gap	S^*	T	gap	S^*	T	gap
A_FA+AA-_1	107	37	12	16	0	33,33%	15	14,3	25,00%	15	1,8	25,00%	15	2,3	25,00%	15	0,2	25,00%
A_FA+AA-_11	99	28	11	13	0	18,18%	13	11,8	18,18%	12	1,0	9,09%	13	1,8	18,18%	13	0,2	18,18%
A_FA+AA-_12	75	20	9	12	0	33,33%	11	7,3	22,22%	11	1,5	22,22%	11	1,0	22,22%	11	0,2	22,22%
A_FA+AA-_13	134	37	17	24	0	41,18%	24	10,5	41,18%	20	1,8	17,65%	23	2,1	35,29%	24	0,0	41,18%
A_FA+AA-_15	68	18	9	11	0	22,22%	10	2,0	11,11%	10	0,8	11,11%	10	0,3	11,11%	10	0,1	11,11%
A_FA+AA-_2	75	19	11	13	0,0	18,18%	13	0,4	18,18%	12	0,3	9,09%	13	0,0	18,18%	13	0,0	18,18%
A_FA+AA-_6	79	21	13	16	0,0	23,08%	16	0,7	23,08%	15	0,3	15,38%	14	2,2	7,69%	14	0,0	7,69%
A_FA+AA-_8	82	28	11	15	0,0	36,36%	15	13,1	36,36%	13	1,7	18,18%	13	3,4	18,18%	15	0,2	36,36%
A_AP-9.d_3	20	16	6	**6**	0,0	0,00%	**6**	1,0	0,00%	**6**	0,2	0,00%	**6**	0,2	0,00%	**6**	0,0	0,00%
A_AP-9.d_10	20	13	6	**6**	0,0	0,00%	**6**	0,3	0,00%	**6**	0,3	0,00%	**6**	0,0	0,00%	**6**	0,0	0,00%
A_AP-9.d_11	27	21	6	7	0,0	16,67%	**6**	1,2	0,00%	**6**	0,5	0,00%	**6**	0,2	0,00%	7	0,0	16,67%
A_AP-9.d_6	31	20	5	**5**	0,0	0,00%	**5**	3,2	0,00%	**5**	1,1	0,00%	**5**	0,3	0,00%	**5**	0,0	0,00%
B_12F18_11	21	15	6	**6**	0,0	0,00%	**6**	0,0	0,00%	**6**	0,1	0,00%	**6**	0,0	0,00%	**6**	0,0	0,00%
B_12M18_12	31	22	6	7	0,0	16,67%	7	1,3	16,67%	7	1,4	16,67%	7	0,0	16,67%	7	0,0	16,67%
B_18AB1_32	15	11	6	7	0,0	16,67%	**6**	0,0	0,00%	7	0,0	16,67%	**6**	0,0	0,00%	**6**	0,0	0,00%
B_18CR1_33	20	18	4	**4**	0,0	0,00%	**4**	2,0	0,00%	**4**	0,0	0,00%	**4**	0,3	0,00%	**4**	0,1	0,00%
B_22X18_50	14	11	10	**10**	0,0	0,00%	**10**	0,2	0,00%	**10**	0,1	0,00%	**10**	0,0	0,00%	**10**	0,0	0,00%
B_23B25_52	29	21	5	**5**	0,0	0,00%	**5**	0,4	0,00%	**5**	0,1	0,00%	**5**	0,0	0,00%	**5**	0,0	0,00%
B_39Q18_82	14	10	5	**5**	0,0	0,00%	**5**	0,0	0,00%	**5**	0,0	0,00%	**5**	0,0	0,00%	**5**	0,0	0,00%
B_42F22_93	18	10	5	**5**	0,0	0,00%	**5**	0,2	0,00%	**5**	0,1	0,00%	**5**	0,0	0,00%	**5**	0,0	0,00%
B_CARLET_137	14	12	5	**5**	0,0	0,00%	**5**	0,0	0,00%	**5**	0,0	0,00%	**5**	0,0	0,00%	**5**	0,0	0,00%
B_CUC28A_138	37	26	6	**6**	0,0	0,00%	**6**	0,0	0,00%	**6**	0,0	0,00%	**6**	0,0	0,00%	**6**	0,0	0,00%
B_GTM18A_139	24	20	5	**5**	0,0	0,00%	**5**	3,1	0,00%	**5**	5,5	0,00%	**5**	0,4	0,00%	**5**	0,1	0,00%

Tabela 5.4: Resultado da ILS para instâncias SCOOP.

Instância	m	n	OPT	ILS		
				S^*	T	gap
A_FA+AA-_1	107	37	12	14	65,8	16,67%
A_FA+AA-_11	99	28	11	12	103,4	9,09%
A_FA+AA-_12	75	20	9	10	69,0	11,11%
A_FA+AA-_13	134	37	17	21	78,1	23,53%
A_FA+AA-_15	68	18	9	10	17,9	11,11%
A_FA+AA-_2	75	19	11	12	10,2	9,09%
A_FA+AA-_6	79	21	13	14	15,5	7,69%
A_FA+AA-_8	82	28	11	13	71,0	18,18%
A_AP-9.d_3	20	16	6	**6**	13,6	0,00%
A_AP-9.d_10	20	13	6	**6**	1,8	0,00%
A_AP-9.d_11	27	21	6	**6**	6,9	0,00%
A_AP-9.d_6	31	20	5	**5**	26,3	0,00%
B_12F18_11	21	15	6	**6**	1,7	0,00%
B_12M18_12	31	22	6	7	26,3	16,67%
B_18AB1_32	15	11	6	**6**	1,3	0,00%
B_18CR1_33	20	18	4	**4**	27,5	0,00%
B_22X18_50	14	11	10	**10**	1,0	0,00%
B_23B25_52	29	21	5	**5**	4,7	0,00%
B_39Q18_82	14	10	5	**5**	0,0	0,00%
B_42F22_93	18	10	5	**5**	4,6	0,00%
B_CARLET_137	14	12	5	**5**	0,1	0,00%
B_CUC28A_138	37	26	6	**6**	0,0	0,00%
B_GTM18A_139	24	20	5	**5**	30,5	0,00%

5.3 Instâncias Faggioli e Bentivoglio (1998)

Conjunto, proposto por Faggioli e Bentivoglio (1998) contém 300 instâncias MOSP artificiais. Os problemas foram agrupados em subconjuntos de 10 instâncias. A Tabela 5.5 apresenta os resultados.

Neste conjunto de instâncias, as buscas locais obtiveram resultados aceitáveis, podendo destacar a *BL2* que obteve os melhores resultados com um *gap* de 9,15% e um tempo de execução médio de 30,4 segundos, melhorando o resultado gerado pela BFS em 4,26% dos experimentos.

A Tabela 5.6 apresenta os resultados da ILS para o conjunto de instâncias.

O tempo médio de execução para as instâncias deste conjunto foi de 3052,4 segundos sendo este o maior conjunto utilizado. O *gap* médio obtido pela heurística proposta foi 6,24%, indicando que a heurística proposta obteve resultados bons para estas instâncias.

Tabela 5.5: Resultados para Faggioli e Bentivoglio (1998).

m	n	i	OPT	BFS			BL1			BL2			BL3			BL4		
				S^*	T	gap	S^*	T	gap	S^*	T	gap	S^*	T	gap	S^*	T	gap
10	10	10	55	**55**	0,0	0,00%	**55**	14,7	0,00%	**55**	3,9	0,00%	**55**	1,5	0,00%	**55**	0,0	0,00%
10	20	10	62	69	0,0	11,29%	67	13,6	8,06%	67	4,5	8,06%	66	1,2	6,45%	67	0,0	8,06%
10	30	10	61	68	0,0	11,48%	68	7,3	11,48%	65	1,0	6,56%	67	0,8	9,84%	67	0,0	9,84%
10	40	10	77	79	0,0	2,60%	79	5,3	2,60%	79	2,1	2,60%	79	0,4	2,60%	79	0,0	2,60%
10	50	10	82	85	0,0	3,66%	85	1,8	3,66%	85	0,9	3,66%	85	0,1	3,66%	84	0,0	2,44%
15	10	10	66	67	0,0	1,52%	67	4,3	1,52%	67	3,3	1,52%	67	0,0	1,52%	67	0,0	1,52%
15	20	10	72	82	0,0	13,89%	81	5,5	12,50%	79	2,3	9,72%	81	0,0	12,50%	81	0,0	12,50%
15	30	10	73	85	0,0	16,44%	83	5,0	13,70%	81	2,3	10,96%	83	0,0	13,70%	84	0,0	15,07%
15	40	10	72	83	0,0	15,28%	82	4,6	13,89%	79	1,3	9,72%	81	0,2	12,50%	82	0,0	13,89%
15	50	10	74	84	0,0	13,51%	80	3,2	8,11%	81	0,9	9,46%	79	0,0	6,76%	80	0,0	8,11%
20	10	10	75	76	0,0	1,33%	76	270,0	1,33%	76	25,1	1,33%	76	35,0	1,33%	76	1,9	1,33%
20	20	10	85	90	0,0	5,88%	90	226,0	5,88%	87	13,9	2,35%	89	31,7	4,71%	90	1,6	5,88%
20	30	10	88	103	0,0	17,05%	102	147,7	15,91%	101	12,5	14,77%	101	19,1	14,77%	101	1,3	14,77%
20	40	10	85	102	0,0	20,00%	101	123,9	18,82%	98	14,8	15,29%	99	18,4	16,47%	99	1,5	16,47%
20	50	10	79	96	0,0	21,52%	91	123,5	15,19%	90	11,5	13,92%	90	14,9	13,92%	90	1,6	13,92%
25	10	10	80	**80**	0,0	0,00%	**80**	21,9	0,00%	**80**	15,5	0,00%	**80**	2,5	0,00%	**80**	0,0	0,00%
25	20	10	98	111	0,0	13,27%	111	22,5	13,27%	107	12,6	9,18%	111	2,3	13,27%	111	0,0	13,27%
25	30	10	105	120	0,0	14,29%	118	30,7	12,38%	115	10,6	9,52%	118	3,4	12,38%	119	0,0	13,33%
25	40	10	103	122	0,0	18,45%	120	38,6	16,50%	116	9,8	12,62%	118	5,3	14,56%	118	0,0	14,56%
25	50	10	100	128	0,0	28,00%	125	31,1	25,00%	114	7,7	14,00%	120	4,2	20,00%	121	0,0	21,00%
30	10	10	78	**78**	0,0	0,00%	**78**	774,6	0,00%	**78**	31,5	0,00%	**78**	103,5	0,00%	**78**	3,6	0,00%
30	20	10	111	121	0,0	9,01%	119	485,3	7,21%	119	33,4	7,21%	119	70,1	7,21%	120	2,0	8,11%
30	30	10	122	135	0,0	10,66%	134	443,4	9,84%	131	36,0	7,38%	134	59,0	9,84%	135	1,5	10,66%
30	40	10	121	142	0,0	17,36%	140	396,5	15,70%	135	33,2	11,57%	138	65,1	14,05%	139	4,7	14,88%
30	50	10	112	144	0,0	28,57%	142	227,7	26,79%	133	23,6	18,75%	139	38,7	24,11%	140	1,7	25,00%
40	10	10	84	**84**	0,0	0,00%	**84**	1643,7	0,00%	**84**	73,1	0,00%	**84**	242,4	0,00%	**84**	6,1	0,00%
40	20	10	130	137	0,0	5,38%	137	1136,6	5,38%	136	78,4	4,62%	137	161,6	5,38%	137	3,8	5,38%
40	30	10	145	167	0,0	15,17%	166	772,0	14,48%	162	76,8	11,72%	164	127,2	13,10%	165	3,3	13,79%
40	40	10	149	176	0,0	18,12%	173	845,4	16,11%	166	61,0	11,41%	171	135,4	14,77%	174	3,5	16,78%
40	50	10	146	181	54,0	23,97%	176	797,6	20,55%	170	57,1	16,44%	175	118,1	19,86%	180	2,3	23,29%

Tabela 5.6: Resultados do *ILS* para instâncias Faggioli e Bentivoglio (1998).

m	n	i	OPT	ILS		
				S^*	T	gap
10	10	10	55	**55**	112,6	0,00%
10	20	10	62	**62**	96,6	0,00%
10	30	10	61	**61**	28,6	0,00%
10	40	10	77	**77**	44,3	0,00%
10	50	10	82	**82**	20,0	0,00%
15	10	10	66	**66**	42,6	0,00%
15	20	10	72	75	46,0	4,17%
15	30	10	73	77	40,8	5,48%
15	40	10	72	76	45,1	5,56%
15	50	10	74	75	27,3	1,35%
20	10	10	75	**75**	2154,2	0,00%
20	20	10	85	87	1494,3	2,35%
20	30	10	88	96	1075,2	9,09%
20	40	10	85	95	1105,7	11,76%
20	50	10	79	86	816,2	8,86%
25	10	10	80	**80**	231,4	0,00%
25	20	10	98	105	269,2	7,14%
25	30	10	105	111	281,1	5,71%
25	40	10	103	112	285,9	8,74%
25	50	10	100	108	244,4	8,00%
30	10	10	78	**78**	5924,1	0,00%
30	20	10	111	115	3786,1	3,60%
30	30	10	122	127	4159,1	4,10%
30	40	10	121	131	3149,4	8,26%
30	50	10	112	127	1801,4	13,39%
40	10	10	84	**84**	15266,0	0,00%
40	20	10	130	136	8146,7	4,62%
40	30	10	145	159	6397,4	9,66%
40	40	10	149	163	5171,7	9,40%
40	50	10	146	169	5321,7	15,75%

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

Este 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 5.7. Os resultados apresentados se referem aos valores acumulados para todas as instâncias do grupo.

Para este conjunto de instâncias a *BFS* obteve um *gap* médio de 6,7% com um tempo de execução médio de 0,2 segundos. Das buscas locais propostas apenas duas obtiveram melhor resultado que a *BFS*, sendo que a *BL3* a de melhor resultado, com *gap* médio de 6,35% e um tempo médio de execução de 77,9 segundos.

A Tabela 5.8 apresenta o resultado da heurística proposta para o conjunto de instâncias.

No conjunto apresentado a heurística *ILS* obtém *gap* igual a zero em cinco subconjuntos de instâncias. O *gap* máximo do método é 15,09%. Em três dos subconjuntos o *ILS* tem *gap* inferior a 4%. Vale ressaltar que o subconjunto *Shaw*, que contém 25 instâncias, foram obtidos os resultados ótimos para 22 instâncias. O tempo médio obtido pela *ILS* foi de 2652,0 segundos.

Tabela 5.7: Resultados para instâncias do *First Constraint Modeling Challenge*.

Instância	m	n	i	OPT	BFS			BL1			BL2			BL3			BL4		
					S^*	T	*gap*	S^*	T	*gap*	S^*	T	*gap*	S^*	T	*gap*	S^*	T	*gap*
nwrsSmaller	20	10	2	7	**7**	0,0	0,00%	**7**	0,0	0,00%	**7**	0,0	0,00%	**7**	0,0	0,00%	**7**	0,0	0,00%
nwrsSmaller	25	15	2	14	15	0,0	7,14%	15	2,9	7,14%	15	798,5	7,14%	15	173,1	7,14%	15	12,0	7,14%
nwrsLarger	30	20	2	24	24	0,0	0,00%	24	31,4	0,00%	24	4,4	0,00%	24	5,9	0,00%	24	0,3	0,00%
nwrsLarger	60	25	2	26	28	0,0	7,69%	28	237,3	7,69%	28	40,2	7,69%	27	53,9	3,85%	28	1,6	7,69%
GP1	50	50	4	155	161	0,3	3,87%	161	2880,3	3,87%	161	89,9	3,87%	161	351,9	3,87%	161	7,0	3,87%
Shaw	20	20	25	342	355	2,0	3,51%	354	416,1	3,51%	349	53,3	2,05%	354	54,9	3,51%	354	2,0	3,51%
Miller	40	20	1	13	**13**	0,0	0,00%	**13**	105,2	0,00%	**13**	8,2	0,00%	**13**	15,9	0,00%	**13**	0,4	0,00%
SP4-1	25	25	1	9	10	0,0	11,11%	10	2,1	11,11%	10	0,9	11,11%	10	0,2	11,11%	10	0,0	11,11%
SP4-2	50	50	1	19	22	0,0	10,53%	21	130,8	10,53%	21	7,1	10,53%	21	20,5	10,53%	22	0,0	15,79%
SP4-3	75	75	1	34	39	0,0	14,71%	39	395,8	14,71%	39	51,8	14,71%	39	55,9	14,71%	39	0,8	14,71%
SP4-4	100	100	1	53	62	0,0	15,09%	61	769,0	15,09%	60	142,9	13,21%	61	124,5	15,09%	61	3,4	15,09%

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

Instância	m	n	i	OPT	ILS		
					S^*	T	gap
nwrsSmaller	20	10	2	7	**7**	6,1	0,00%
nwrsSmaller	25	15	2	14	**14**	44,4	0,00%
nwrsLarger	30	20	2	24	**24**	481,5	0,00%
nwrsLarger	60	25	2	26	27	3776,0	3,85%
GP1	50	50	4	155	161	17644,4	3,87%
Shaw	20	20	25	342	345	3173,3	0,88%
Miller	40	20	1	13	**13**	705,9	0,00%
SP4-1	25	25	1	9	**9**	23,2	0,00%
SP4-2	50	50	1	19	21	0,0	10,53%
SP4-3	75	75	1	34	38	886,1	11,76%
SP4-4	100	100	1	53	61	2431,6	15,09%

5.5 Comparação Entre Os Métodos Propostos

A Tabela 5.9 apresenta o *ranking* ordinal dos métodos para cada conjunto de instâncias. Conforme mencionado anteriormente, o *ranking* ordinal exprime o posicionamento dos métodos de acordo com o desempenho correspondente nos experimentos realizados, permitindo uma comparação direta entre os mesmos.

Tabela 5.9: *Ranking* dos métodos para cada instância.

	BL1	BL2	BL3	BL4	ILS
VLSI	5	3	1	4	1
SCOOP	4	3	2	4	1
FAGGIOLI	5	2	3	4	1
SHAW MILER	4	2	3	5	1

Como esperado, a heurística proposta *ILS* obteve os melhores resultados em todas as instâncias, quando comparado a todas as buscas locais aplicadas isoladamente. Dentre as buscas locais, destacam-se *BL3* e *BL2*, que obtiveram os melhores resultados, ao passo que as outras duas buscas locais se revezaram com o pior desempenho.

Capítulo 6

Considerações Finais e Propostas de Trabalhos Futuros

Este trabalho apresentou o estudo inicial do problema de determinação do leiaute de matrizes de portas, incluindo sua definição e uma revisão sistemática da literatura, gerando toda a base conceitual.

Foi apresentada a implementação do método busca local iterativa para a solução do problema tratado. A referida heurística utiliza pré-processamento dos dados de entrada, representação e busca em grafos e uma etapa de sequenciamento para geração da solução inicial. Foram implementados 4 métodos de busca local e um de perturbação para uso em conjunto. Cada uma dessas etapas foi descrita em detalhes e ilustrada através de exemplos.

Experimentos computacionais incluíram instâncias reais e artificiais da literatura. De acordo com os resultados reportados, apesar do bom desempenho para parte das instâncias, é possível refinar os métodos para que haja maior aprimoramento das soluções iniciais geradas.

Como trabalhos futuros propõe-se a implementação das metaheurísticas Busca Local Adaptativa (*Adaptive Local Search*), em conjunto com os procedimentos de busca local utilizados neste trabalho.

Referências Bibliográficas

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

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. *Computers and Operations Research 31(14): 2315–2332.*

Carvalho, M. A. M. e Soma, N. Y. (2015). A bfs applied to the minimization. *J Oper Res Soc*, 66(6):936–946.

Chen, S.-J. e Hu, Y. H. (1988). A new algorithm for cmos gate matrix layout. *IEEE International Conference*, 21(8):969–974.

Chen, S.-J. e Hu, Y. H. (1990a). 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.*

Chen, S.-J. e Hu, Y. H. (1990b). A heuristic and an exact method for the gate matrix connection cost minimization problem. *Computers and Digital Techniques, IEE Proceed.*

Cormen, T. H.; Stein, C.; Rivest, R. L. e Leiserson, C. E. (2001). *Introduction to Algorithms.* 2nd edn, mcgraw-hill higher education: Cambridge, ma. edição.

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

De Giovanni, L.; Massi, G. e Pezzella, F. (2013). An adaptive genetic algorithm for large-size open stack problems. *International Journal of Production Research*, 51(3):682–697.

Enderlein, R. (1994). *Microeletrônica: Uma introdução, ao universo dos microchips, seu funcionamento, fabricação e aplicações.* Usp 1994 edição.

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.

Giovanni, L. D.; Massib, G.; Pezzella, F.; Pfetschc, M.; Rinaldid, G. e Venturad, P. (2013). Gm-learn : an iterative learning algorithm for cmos gate matrix layout. *International Transactions in Operational Research, 20(5):627–643.*

Glover, F.; Kochenberger, G. e Gary, A. (2003). Handbook of metaheuristics, volume 57 of international series in operations research & management science.

Gonçalves, J. F.; Resende, M. G. C. e Costa, M. D. (2014). A biased random-key genetic algorithm for the minimization of open stacks problem. *International Transactions in Operational Research.*

Heinbuch, D. V. (1988). Cmos3 cell library. *Reading, MA: AddisonWesley.*

Hong, Y. S.; Park, K. H. e Kim, M. (1989). A heuristic for ordering the columns in one-dimensional logic array. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on, 9(8):836–845IEEE Trans. Computer-Aided Design, vol. 8, pp. 547–562.*

Hwang, D. K.; Fuchs, W. K. e Kang, S. M. (1987). An efficient approach to gate matrix layout. *IEEE Trans. On Computer-Aided Design, Vol. CAD-6, No. 5, pp. 802-809, September.*

Kashiwabara, T. e Fujisawa, T. (1979). Np-completeness of the problem of finding a minimum clique number interval graph containing a given graph as a subgraph. *Proceedings of the 1979 IEEE International Symposium on Circuits e Systems, Tokyo, Japan, July. p. 657–60.*

Linhares, A. (1999). Synthesizing a predatory search strategy for vlsi layouts. *IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION.*

Linhares, A. e Yanasse, H. H. (2002). Connections between cutting-pattern sequencing, vlsi design, and exible machines. *Computers and Operations Research 29.*

Linhares, A.; Yanasse, H. H. e Torreão, J. R. A. (1999). Linear gate assignment: A fast statistical mechanics approach. *IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS.*

Mendes, A. e Linhares, A. (2004). A multiple-population evolutionary approach to gate matrix layout. *International Journal of Systems Science.*

Oliveira, A. C. M. e Lorena, L. A. N. (2002). A constructive genetic algorithm for gate matrix layout problems. *IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS, VOL. 21, NO. 8, AUGUST.*

Shahookar, K.; Khamisani, W.; Mazurnder, P. e Reddy, S. (1994). Genetic beam search for gate matrix layout. *Genetic beam search for gate matrix layout.*

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

Tooley, M. (2006). *Circuitos eletrônicos, Fundamentos e Aplicações.* Elsevier 2006 edição.

Wing, O.; Huang, S. e Wang, R. (1985). A constructive genetic algorithm for gate matrix layout problems. *IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN, VOL. CAD-4, NO. 3.*

Yanasse, H. H. (1997). A transformation for solving a pattern sequencing problem 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.