Uma Heurística Aplicada à Produção em Microeletrônica

João Vitor Mascarenhas dos Santos
Departamento de Ciência da Computação, Universidade Federal de Ouro Preto
Campus Morro do Cruzeiro, Ouro Preto, Minas Gerais, 35400-000, Brasil.
joaovitormascarenhas@yahoo.com.br

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

RESUMO

Este artigo apresenta uma heurística para o Problema de Determinação do Leiaute de Matrizes de Portas (ou GMLP, do inglês *Gate Matrix Layout Problem*), um problema de aplicação prática na indústria microeletrônica. No contexto de projeto de circuitos VLSI (*Very Large Scale Integration*), este problema pede por uma permutação das colunas de uma matriz de portas tal que o número exigido de trilhas para implementação dos circuitos – e consequentemente a área – sejam minimizados. A heurística se baseia em busca em grafos seguida por procedimentos de busca local e é comparada com dois métodos da literatura em experimentos computacionais abrangentes que consideraram conjuntos de instâncias reais e artificiais da literatura. Os resultados reportados mostram que a heurística proposta, em um curto intervalo de tempo, pôde igualar boa parte dos resultados da literatura, aprimorar alguns deles e também obter um baixo *gap* quando comparada às soluções ótimas e limitantes inferiores disponíveis.

PALAVRAS CHAVE. Leiaute de Matrizes de Portas. Heurísticas. Escalonamento.

Área Principal: IND

ABSTRACT

This paper introduces a heuristic method for the Gate Matrix Layout Problem (GMLP), a problem with practical application in the microelectronics industry. In the context of Very Large Scale Integration (VLSI) design, this problem asks for a permutation of columns of a gate matrix such that the number of required tracks necessary to implement the corresponding integrated circuit, and thus the area, are minimized. The proposed heuristic is based on graph search followed by local search procedures and is compared to two methods from the literature in comprehensive computational experiments that considered four datasets publicly available of real and artificial instances. The reported results show that the proposed method, in a short amount of time, was able to match a good part of the best results available, to improve some of them and to obtain a low gap when compared to optimal solutions and lower bounds available.

KEYWORDS. Gate Matrix Layout. Heuristics. Scheduling.

Main Area: IND

1. Introdução

Atualmente, a microeletrônica está presente nas mais diversas áreas, sendo fundamental 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.

Na criação de componentes deste ramo da eletrônica, são utilizados circuitos integrados em larga escala (ou VLSI, do inglês *Very Large Scale Integration*), projetados pelo uso de uma matriz lógica programável. Com o intuito de otimizar a produção dos referidos componentes, busca-se uma ferramenta computacional para determinação do leiaute físico otimizado dos componentes para que seja possível produzir circuitos mais baratos, mais rápidos e mais compactos. A área do circuito é fator de suma importância na minimização do custo da produção, já que os mesmos são produzidos sobre discos de silício, normalmente chamados de *wafers*, onde o custo do circuito é inversamente proporcional a quantidade de circuitos feitos em um só *wafer*, ou seja, quanto menor o tamanho do circuito, uma maior quantidade poderá ser produzida em um único *wafer*, tornando-o, assim, mais barato.

Uma *Matriz de Portas* é um dispositivo lógico que dispõe de duas dimensões de *portas*, uma AND e outra OR, ambas programáveis. Conexões entre estas portas podem ser realizadas para alcançar resultados específicos. Desta forma, a matriz de portas é utilizada para elaborar circuitos lógicos combinatórios, composta por n portas lógicas e m *redes* – um subconjunto de portas ligadas por um *fio*. No caso de diferentes redes utilizarem uma mesma porta, cada rede deve ser alocada em uma *trilha* diferente garantindo assim, que não haja sobreposição das redes.

Na Figura 1(a) é apresentada uma matriz de portas com seis portas enumeradas de $P1$ a $P6$, representadas como as linhas verticais (colunas), e seis redes enumeradas de $R1$ a $R6$, representadas pelos conjuntos de pontos na horizontal. Em seguida, a Figura 1(b) mostra as conexões entre as portas da rede, onde, para ligar uma porta a outra, pode ser necessário passar por outras portas, sem utilizá-las. Um diferente leiaute das portas da matriz é apresentado na Figura 1(c), no qual é possível que diferentes redes sejam alocadas a uma mesma trilha.

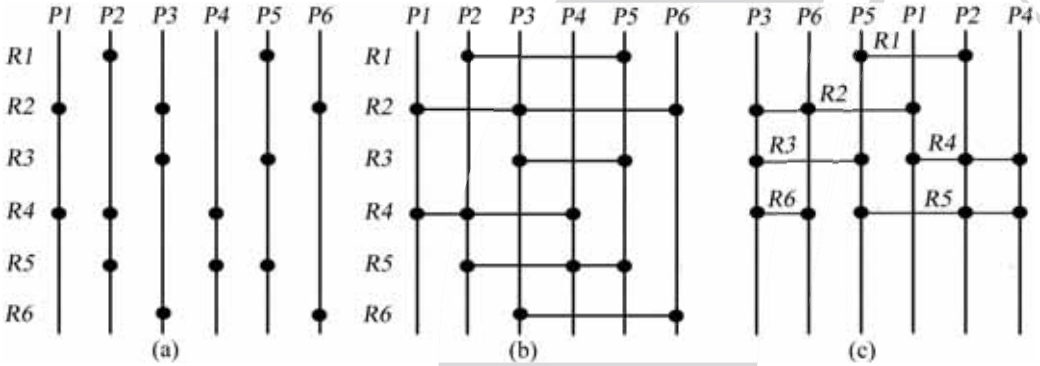

Figura 1. Matriz de Portas (a), com redes destacadas (b) e com uma permutação diferente das portas (c).

O Problema de Determinação do Leiaute de Matrizes de Portas (ou GMLP, do inglês *Gate Matrix Layout Problem*), consiste em determinar uma permutação das portas de uma matriz tal que a quantidade de trilhas necessárias para implementar todas as redes seja minimizada, e, por consequência, a área do circuito também seja minimizada. Como demonstrado na Figura 1(b), são necessárias seis trilhas para acomodar todas as redes do circuito. Já na Figura 1(c), com outra permutação das portas, são necessárias quatro trilhas apenas.

Como definido por De Giovanni et al. (2013), uma instância do GMLP pode ser descrita por uma matriz $M \in \{0, 1\}$ $m \times n$. As linhas da matriz M são associadas às m redes e as colunas são associadas às n portas, tal que $M(i, j) = 1$ se e somente se a rede i incluir a porta j. Uma

solução para o GMLP é uma sequência $\phi : [1, \ldots, n] \rightarrow [1, \ldots, n]$, em que $\phi(j)$ indica posição da porta j na matriz de portas. Tal solução define uma nova matriz M_ϕ, obtida pela permutação das colunas de M de acordo com ϕ. A partir da matriz M_ϕ, possuidora da propriedade dos 1's consecutivos, obtemos uma nova matriz $\bar{M}_\phi$ de forma que $\bar{M}_\phi(i, j) = 1$ se e somente se, de acordo com ϕ, o fio da rede i incluir ou cruzar a porta j. Portanto, dada uma matriz $M \in \{0, 1\}\ m \times n$, o objetivo do GMLP é determinar uma permutação das colunas que resulte em uma matriz $\bar{M}_\phi$ correspondente tal que a maior soma de elementos em qualquer coluna seja minimizada, o que equivale a minimizar o número máximo de trilhas necessárias para implementação do circuito correspondente.

A Figura 2(a) apresenta a matriz M do circuito apresentado anteriormente na Figura 1(a). O circuito com as portas permutadas exibido na Figura 1(c) é representado pela matriz M_ϕ na Figura 2(b), onde, após aplicar a propriedade dos 1's consecutivos, obtemos a nova matriz $\bar{M}_\phi$ onde as posições entre o primeiro e o último 1 de cada rede recebem também o valor 1, como mostra a Figura 2(c). Assim, determinar o custo da solução significa encontrar a maior soma entre as colunas. A importância da propriedade dos 1's consecutivos se deve à necessidade de representar que um fio cruza uma porta, mesmo que esta não faça parte da rede. Por exemplo, no circuito da Figura 1(c), o fio da rede $R1$ cruza a porta $P1$ e, consequentemente, na matriz da Figura 2(c) a posição $\bar{M}_\phi(1, 1)$ possui valor 1.

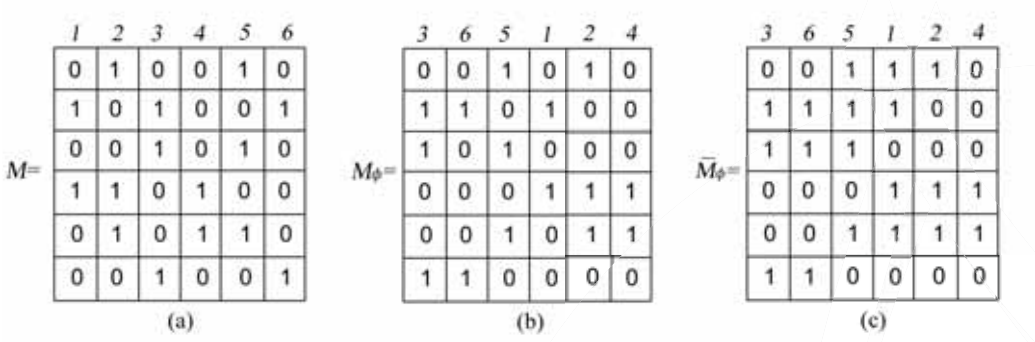

Figura 2. Matrizes M (a), M_ϕ (b) e $\bar{M}_\phi$ (c).

O Problema de Determinação do Leiaute de Matrizes de Portas foi provado ser NP-difícil por uma redução do problema de aumento grafos de intervalo (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): Problema de 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*.

Recentemente, o interesse na aplicação prática deste problema têm crescido. Com a recente instalação, em Minas Gerais, da primeira fábrica de semicondutores do hemisfério sul, este trabalho possui o potencial para gerar inovação aplicável a este novo nicho industrial.

Este artigo está organizado como a seguir. Uma breve revisão da literatura é realizada na Seção 2. Na Seção 3, a heurística proposta é descrita em detalhes, cada etapa é ilustrada. Experimentos computacionais extensivos que levaram em consideração 190 instâncias reais de empresas e artificiais da literatura são reportados na Seção 4, e finalmente, na Seção 5, conclusões sobre este trabalho de pesquisa são realizadas e trabalhos futuros são indicados.

2. Revisão da Literatura

Métodos heurísticos vêm sendo aplicados as GMLP desde o trabalho de Wing, Huang e Wang (1985), no qual dois circuitos reais foram modelados usando grafos de intervalos, considerando portas em comum em diferentes trilhas como uma interseção. Posteriormente, Hwang, Fuchs

e Kang (1987) conseguiram resultados mais relevantes com o algoritmo *min-net-cut* modificado cuja complexidade de tempo é $O(n\ log\ n)$. O mesmo algoritmo foi utilizado novamente por Chen e Hou (1988), para atingir melhores resultados que os anteriores da literatura nos 5 circuitos testados.

Na década seguinte, os trabalhos de Chen e Hu (1990) e Hu e Chen (1990) alcançaram, ou determinaram, os melhores resultados para todas as instâncias disponíveis na literatura até aquele momento. Foram apresentados dois algoritmos, *GM-Plan* e *GM-Learn*, ambos baseados em paradigmas de inteligência artificial.

No final da década de 80 e boa parte da década de 90, o GMLP foi abordado predominantemente por técnicas metaheurísticas e métodos híbridos. Shahookar et al. (1993) propuseram uma implementação de algoritmos genéticos e *Beam Search*, denominada *Genetic Beam Search*. Com a estratégia de *Busca Predatória*, Linhares (1999) alcançou os resultados obtidos por *GM-Plan* e *GM-Learn*. Além disto, das 25 instâncias, em 12 foi possível igualar o número de trilhas ao limite inferior, e em 4 foram atingidos os melhores resultados da literatura. Linhares, Yanasse e Torreão (1999) apresentaram a *Otimização Microcanônica*, um procedimento derivado da física estatística assim como o *Simulated Annealing*. Este método foi capaz de igualar todos os resultados anteriores de Chen e Hu (1990) e Hu e Chen (1990), mas não os resultados obtidos pela *Busca Predatória*.

Algoritmos Genéticos Construtivos foram utilizados por Oliveira e Lorena (2002), alcançando todos os melhores resultados conhecidos da literatura, obtidos anteriormente pela *Otimização Microcanônica*. Segundo os autores, este algoritmo genético construtivo é mais robusto do que a *Otimização Microcanônica*. Uma abordagem evolutiva de múltiplas populações foi introduzida por Mendes e Linhares (2004), agregando algoritmos genético e memético. Este trabalho apresentou resultados iguais aos da *Otimização Microcanônica*, superando todas as abordagens metaheurísticas e heurísticas anteriores da literatura – incluindo *Simulated Annealing*, *GM-Plan*, *GM-Learn*, e heurísticas construtivas – tanto em qualidade das soluções obtidas quanto em eficiência, em termos de convergência do algoritmo. A rápida convergência é justificada pela adição de uma busca local heurística que proporcionou a redução do espaço de busca, considerando apenas o gargalo do problema, o que foi denominado *colunas críticas*.

Os métodos exatos aplicados ao GMLP são encontrados na literatura com frequência muito menor. Destaca-se a formulação por programação dinâmica apresentada em Deo, Krishnamoorthy e Langston (1987). No entanto, não foram realizados experimentos computacionais que permitissem aferir a qualidade da formulação. Neste mesmo trabalho o autor prova não haver algoritmo de aproximação absoluta para o GMLP. Recentemente, De Giovanni et al. (2013) considerou o problema de minimização do comprimento dos fios utilizados nas redes, com restrições quanto ao número de trilhas utilizadas. Um algoritmo genético e um procedimento *branch-and-cut* foram propostos e testados em mais de trezentas instâncias da literatura. Especificamente, o algoritmo genético foi utilizado para estabelecer um limitante superior para o número de trilhas necessárias e o procedimento *branch-and-cut* foi utilizado para minimizar o comprimento dos fios. Para algumas instâncias foram obtidos resultados ruins com *gap* alto ou sequer foi possível resolver a relaxação linear em uma hora.

3. Método Proposto

A heurística proposta neste trabalho se baseia na representação do problema por grafos e é composta por dois métodos de pré-processamento, geração da solução inicial e aprimoramento por busca local. O Algoritmo 1 apresenta uma visão geral da heurística proposta. Nas seções seguintes, cada um dos componentes da heurística são descritos.

Conforme Yanasse (1997a), na representação do GMLP por grafos, vértices representam as redes e arestas indicam que duas redes compartilham alguma porta em comum. No circuito de exemplo da Figura 1(a) as redes $R2$ e $R4$ compartilham a porta $P1$, portanto, no grafo existe uma aresta entre os vértices 2 e 4, conforme pode ser observado na Figura 3(a).

Dois diferentes métodos de pré-processamento (linhas 1 e 3) são aplicados no intuito de diminuir as dimensões do problema, sem que seja causado algum prejuízo à qualidade da solução

final, gerando uma matriz reduzida M_r. Após à criação do grafo original (linha 3), um grafo reduzido G_r é gerado. Posteriormente, uma solução inicial é gerada por meio da aplicação da Busca em Largura ao grafo reduzido (linha 4). Esta solução inicial consiste da lista S que contém a ordem em que os vértices foram explorados.

A lista de vértices/redes é então transformada em uma sequência de portas ϕ (linha 5) que é utilizada para determinar a matriz permutada $\bar{M}_\phi$ (linha 6). Em seguida, dois procedimentos de busca local (linhas 8 e 9) são aplicados à solução corrente enquanto houver melhoria da mesma, no intuito de aprimorá-la por meio da compactação de redes individuais. Ao final, a matriz $\bar{M}_\phi$ com a solução é retornada.

Entrada: Matriz M, conjunto de portas P
1 $M_r \leftarrow$**PreProcessamentoDominancia**(M);
2 $G \leftarrow$**ConstroiGrafo**(M_r);
3 $G_r \leftarrow$**PreProcessamentoGemeos**(G);
4 $S \leftarrow$ **BFS**(G_r);
5 $\phi \leftarrow$**SequenciamentoPortas**(S, P);
6 $\bar{M}_\phi \leftarrow$**GeraMatriz**(M, ϕ);
7 **repita**
8 | **BuscaLocal1**($\bar{M}_\phi$);
9 | **BuscaLocal2**($\bar{M}_\phi$);
10 **até** *não haver melhoria em* $\bar{M}_\phi$;
11 **retorna** $\bar{M}_\phi$;

Algoritmo 1. Visão geral da heurística proposta.

3.1. Pré-processamento de Portas Dominadas

De acordo com Yanasse e Senne (2010), uma porta Pi *domina* outra porta Pj qualquer, se e somente se o conjunto de redes que utilizam a porta Pj for um subconjunto das redes que utilizam a porta Pi. Portas dominadas podem ser desconsideradas no processo de resolução do problema e sequenciadas na solução final logo após as portas que as dominam, sem prejuízo à solução, uma vez que ambas podem utilizar as mesmas trilhas.

Como exemplo, no circuito da Figura 1(b), $P2$ é utilizada pelo conjunto de redes $\{R1, R4, R5\}$, $P4$ por $\{R4, R5\}$, $P3$ pelo conjunto $\{R2, R3, R6\}$ e $P6$ por $\{R2, R6\}$. Portanto, $P4 \subset P2$ e $P6 \subset P3$. Logo $P2$ domina $P4$ e $P3$ domina $P6$. O problema original possui 6 portas, porém, $P2$ e $P4$ podem ser consideradas como uma única porta, assim como $P3$ e $P6$. Desta maneira o problema reduzido conta apenas com 4 portas.

A verificação de dominância entre portas exige que para cada par de portas sejam comparados os conjuntos de redes que as contém. Consequentemente a complexidade deste procedimento é limitada por $O(n^2m)$.

3.2. Pré-processamento de Vértices Gêmeos

Novamente de acordo com Yanasse e Senne (2010), considerando a representação do GMLP por grafos, se dois vértices adjacentes entre si compartilham do mesmo conjunto de vértices adjacentes, estes são considerados *vértices gêmeos*. Tais vértices podem ser considerados como um só, reduzindo o tamanho do problema novamente sem prejuízo à solução, uma vez que as respectivas redes são similares.

A Figura 3(a) apresenta a representação do circuito da Figura 1(a) como um grafo. Os vértices 1 e 5 são adjacentes aos vértices 3 e 4, e também são adjacentes entre si. Estes são vértices gêmeos e, portanto, podem ser considerados como um único vértice, reduzindo o tamanho do grafo, vide Figura 3(b).

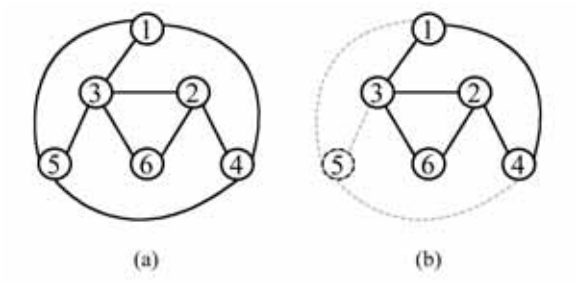

Figura 3. Representação do problema em grafo (a), em que os vértices 1 e 5 são gêmeos e (b) grafo reduzido.

A detecção de vértices gêmeos requer o exame e comparação da vizinhança de todos os pares de vértices, portanto, sua complexidade é limitada por $O(m^3)$.

3.3. Busca em Largura

Uma estratégia efetiva para solução do GMLP e problemas equivalentes é sequenciar as portas levando em consideração o relacionamento entre as redes, conforme estabelecido por Yanasse (1997b). Partindo deste princípio, a heurística proposta gera uma solução inicial que relaciona as redes que possuem algum grau de similaridade e que posteriormente diminuam o número de trilhas necessárias, guiando uma busca no grafo por meio de um critério guloso.

Uma Busca em Largura (ou BFS, do inglês *Breadth-First Search*) é realizada no grafo para este fim. No intuito de que as redes com menos portas sejam sequenciadas preferencialmente e de maneira contígua, a BFS empregada utiliza como critério de escolha do próximo vértice a ser explorado a quantidade de portas que a rede possui. Desta maneira, a BFS inicia a exploração do grafo a partir do vértice correspondente à rede com menor número de portas, e usa esta mesma regra gulosa para determinar a ordem em que os demais vértices da vizinhança serão explorados. Ao final da BFS, obtêm-se uma sequência S com a ordem de exploração dos vértices.

A Figura 4(a) mostra o grafo correspondente ao circuito da Figura 1(a), com os vértices gêmeos 1 e 5 unidos pelo pré-processamento e com os respectivos números de portas representados ao lado dos vértices. Note que, como as portas $P4$ e $P6$ foram dominadas, o número de portas das redes $R2$, $R4$ e $R6$ foi decrementado, já que as mesmas utilizavam alguma das portas dominadas. Começando pelo vértice de menor número de portas, o vértice 6, a Figura 4(b) exemplifica a primeira iteração da BFS com a exploração do vértice 6 e de sua vizinhança. À medida que os vértices são visitados, os mesmos são inseridos na fila S, que no fim, Figura 4(d),gera a solução para esta etapa do algoritmo.

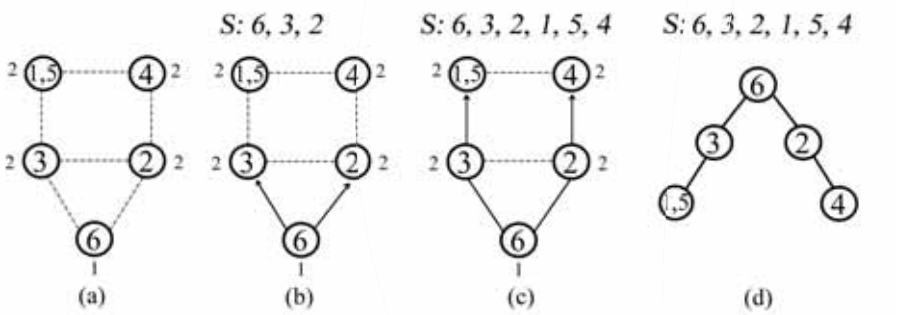

Figura 4. Aplicação da Busca em Largura proposta ao circuito da figura 1.

Claramente, o critério utilizado para guiar a busca não necessariamente leva à soluções ótimas. Com efeito, a busca em largura e o critério guloso de explorar preferencialmente redes com menos portas propostos neste trabalho compõem uma heurística, que, de acordo com os resultados reportados, produz boas soluções para o GMLP.

A busca em largura proposta requer que a vizinhança de cada vértice esteja ordenada de forma não decrescente de número de portas por rede. Considerando a representação do grafo por listas de adjacências, a complexidade deste procedimento é limitada por $O(m^2\ log\ m)$.

3.4. Sequenciamento de Portas

A sequência de redes gerada pela BFS não é a solução para o GMLP. Para obter a sequência de portas necessária para solução do problema, foi utilizado o método apresentado por Becceneri et al. (2004). O princípio deste método é manter a relação estabelecida na sequência de redes, de maneira que portas não relacionadas a determinadas redes não são sequenciadas prematuramente, o que exigiria um maior número de trilhas na solução.

Este método percorre a sequência de redes gerada anteriormente, examinando-as. A cada nova rede examinada, busca-se entre as portas ainda não sequenciadas aquelas que são utilizadas por todas as redes já examinadas até aquele instante, incluindo-as na solução. Nesta etapa, as portas dominadas voltam a ser consideradas, a fim de se obter uma solução completa para o problema. As portas dominadas são sequenciadas logo após as portas que as dominam.

Considerando a sequência de exploração obtida pela BFS, $S = [6, 3, 2, 1, 5, 4]$, o sequenciamento de portas considera inicialmente as portas, $P1$, $P2$, $P3$ e $P5$. Quando o conjunto de vértices {6, 3, 2} é examinado, a porta $P3$ pode ser sequenciada, seguida pela porta dominada $P6$, como mostra a Figura 5(a). Posteriormente, na Figura 5(b), com o exame dos próximos vértices 1 e 5, a porta $P5$ é sequenciada. Por fim, com a adição do vértice 4 ao conjunto de vértices examinados, as portas $P1$ e $P2$ finalmente podem ser sequenciadas, seguidas pela porta dominada $P4$, vide Figura 5(c). Assim, obtém-se uma sequência de portas ϕ: [3, 6, 5, 1, 2, 4], ou seja, uma solução para o problema, como mostrado no circuito da Figura 1(c).

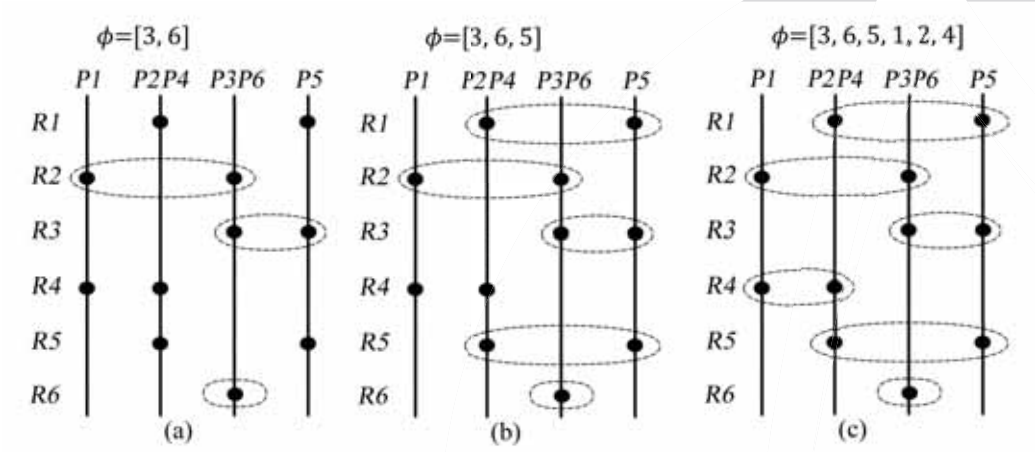

Figura 5. Sequenciamento de portas. As elipses pontilhadas indicam as redes já examinadas.

Este método para sequenciamento de portas requer a comparação do conjunto de redes já examinadas e o conjunto de redes que contêm cada porta, podendo ser executado em tempo $O(mn)$.

3.5. Aprimoramento da Solução

Intuitivamente, quando pensamos na otimização do leiaute de uma matriz de portas e na diminuição da área do circuito implementado, uma solução seria aproximar as portas de uma mesma rede tanto quanto possível. Como na Figura 1(c), para as redes $R1$ e $R5$, poderia ser interessante que as portas $P2$ e $P4$ fossem sequenciadas antes da porta $P1$, uma vez que esta última porta é cruzada por ambas as redes e, portanto, exige maior comprimento de fios. Todavia, minimizar o número de trilhas não é equivalente a reduzir o comprimento das mesmas, como mostrado por Linhares e Yanasse (2002).

Apesar desta não equivalência entre os problemas, este raciocínio é utilizado como regra heurística em uma busca local para tentar diminuir o número máximo de trilhas exigido pelo gargalo

do problema. Mais detalhadamente, em uma solução do GMLP há uma ou mais portas com soma máxima de conexões e cruzamentos – como não é possível reduzir o número de conexões de uma porta, a idéia é diminuir a quantidade de fios que a cruzam. É importante ressaltar que o objetivo não é minimizar o comprimento total dos fios utilizados, mas sim, compactar as redes que cruzam os gargalos do problema, o que equivale a reduzir os 1's consecutivos das colunas de maior soma na matriz $\bar{M}_\phi$. Desta forma, a busca local tenta compactar algumas redes, agrupando suas portas em duas etapas: uma movendo as portas para a direita e outra movendo-as para a esquerda. Todos os movimentos que não gerem piora na qualidade da solução são realizados.

A Figura 6(a) exibe o circuito dos exemplos anteriores com a solução ϕ: [3, 6, 5, 1, 2, 4], obtida pelo sequenciamento de portas. Para fins de ilustração da busca local, consideremos a rede $R2$: Na primeira etapa da busca local, haverá uma tentativa de realocação das primeiras portas da rede ($P3$ e a porta dominada por ela, $P6$), para a direita, logo antes da porta $P1$, a fim de unir as portas conectadas pela rede. No entanto, como mostra a Figura 6(b), tal alteração causa um aumento no custo da solução, de 4 para 5 trilhas, logo, a sequência de portas ϕ_2: [5, 3, 6, 1, 2, 4] é descartada. Na segunda etapa, haverá uma tentativa de realocar as portas da mesma rede para a esquerda, logo depois da primeira porta da rede, desta forma, trocando a ordem das portas $P1$ e $P5$ obtém-se a solução ϕ_3: [3, 6, 1, 5, 2, 4] de custo 4, também. Assim o algoritmo passa a considerar a solução ϕ_3 como solução final com o intuito de, nas próximas iterações, melhorar o custo da solução.

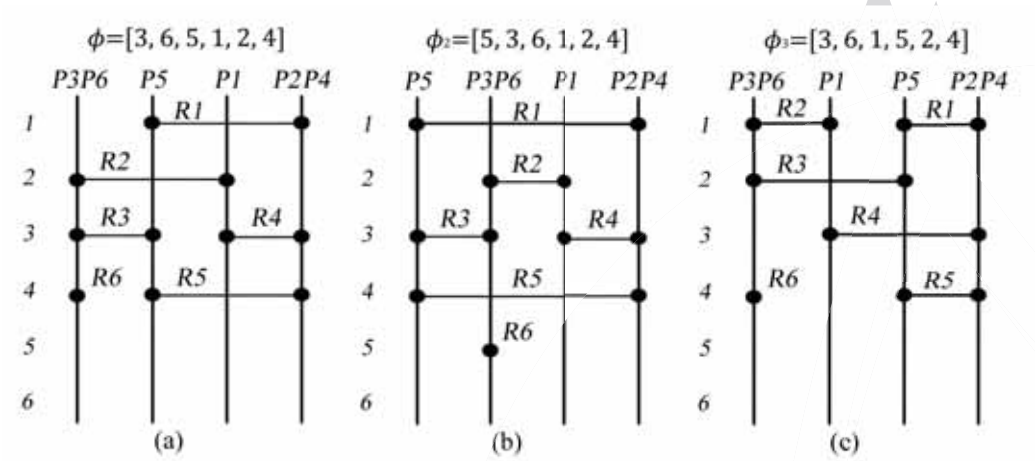

Figura 6. Aplicação da busca local.

Os dois procedimentos de busca local possuem a mesma complexidade computacional. Ambos exigem que haja uma tentativa de realocação das portas de cada rede e posterior avaliação da solução, o que limita a complexidade computacional por $O(m^2n^2)$. Todavia, na ausência de um valor específico para o número máximo de portas em qualquer rede, o cálculo realizado considera que este número seja n, muito embora este não seja o caso na prática, de acordo as instâncias reais utilizadas e com os resultados reportados a respeito do tempo de execução da heurística.

4. Experimentos Computacionais

Os experimentos computacionais foram realizados em um computador com processador *Intel i5 Quad Core* de 3.2 GHz com 16 GB RAM sob o sistema operacional Ubuntu 12.4.1. O código da heurística proposta foi escrito em C++, compilado com g ++ 4.4.1 e a opção de otimização -O3. Três conjuntos de instâncias foram utilizados nas experiências computacionais, e dois métodos diferentes da literatura foram utilizados nas comparações. Entre os conjuntos de instâncias, há um conjunto de instâncias específicas do GMLP e dois conjuntos de instâncias MOSP (*Minimization of Open Stacks Problem*), um problema industrial de formulação equivalente à formulação do GMLP, conforme mencionado na Seção 1. Onde mencionado, o *gap* (ou distância percentual) é calculado como *100×(valor da solução heurística) / valor de referência*). Os tempos de execução

não são comparados nestes experimentos pois, como os métodos foram executados em diferentes arquiteturas, uma comparação justa não se torna possível.

4.1. Instâncias Reais GMLP

O primeiro conjunto contém 25 instâncias GMLP reais de empresas asiáticas, introduzidas por Hu e Chen (1990). As dimensões variam entre 7 linhas e colunas a 202 linhas e 141 colunas. A Tabela 1 apresenta os resultados (expressos em número de trilhas) e os tempos de execução (expressos em milissegundos) da heurística proposta para estas instâncias. Os melhores resultados para este conjunto, obtidos pela Busca Predatória de Linhares (1999), também são apresentados como valores de referência.

Tabela 1. Resultados para instâncias reais GMLP.

Instância	Busca Predatória	Heuríistica Proposta	
	Resultado	Resultado	Tempo (ms)
v4000	5	6	0,44
v4050	5	6	0,59
v4090	10	10	1,33
v4470	10	10	4,43
vc1	9	9	7,82
vl	3	3	0,50
vw1	4	4	7,67
vw2	5	5	0,57
w1	4	5	0,79
w2	14	14	4,44
w3	21	22	18,33
w4	32	29	55,12
wan	6	6	2,68
wli	4	4	0,92
wsn	8	8	0,75
x0	11	12	3,24
x1	5	5	0,29
x2	6	6	0,30
x3	7	7	0,45
x4	2	2	0,26
x5	2	2	0,34
x6	2	2	0,74
x7	4	4	0,68
x8	4	4	0,74
x9	4	4	1,21

A heurística proposta foi capaz de, em uma fração de segundo, alcançar 19 dos melhores resultados obtidos pela Busca Predatória de Linhares (1999), e melhorar um deles. No entanto, a melhor solução para a instância *w4* (27) foi determinada previamente pelo Algoritmo Genético Construtivo apresentado por Oliveira e Lorena (2002). Em outros cinco casos, piores soluções foram obtidas, todos diferindo por apenas uma trilha adicional.

4.2. Instâncias Reais MOSP

Este conjunto de instâncias, fornecido pelo *SCOOP Consortium* (2009), contém 187 instâncias MOSP reais de duas empresas européias. Entretanto, a maioria destas instâncias são muito pequenas (por exemplo, 2 linhas e colunas), possuindo soluções triviais. Deste conjunto, 15 instâncias, com dimensões que variam de 10 linhas e colunas a 202 linhas e 141 colunas foram selecionadas para serem usadas nos experimentos. A Tabela 2 apresenta os resultados utilizando a mesma

configuração da tabela anterior, porém, agora *Wood. A (4)* indica um grupo de quatro instâncias e *Wood. B (11)* indica um grupo de 11 instâncias, portanto, os valores médios são apresentados. Os valores de referência são aqueles obtidos pelo algoritmo genético (GA) proposto por De Giovanni et al. (2013).

Tabela 2. Resultados para as instâncias reais MOSP.

Instância	AG	Heurística Proposta	
	Resultado	Resultado	Tempo (ms)
Wood. A (4)	5,75	5	0,63
Wood. B (11)	5,73	5,91	0,69

Para as instâncias agrupadas (*Wood. A* e *Wood. B*) não são reportados em outros trabalhos os valores individuais de cada uma das instâncias, o que impossibilita uma comparação detalhada. Os melhores resultados médios anteriores para as instâncias do grupo *Wood. A* foram melhorados pela heurística proposta, ao passo que para o grupo *Wood. B*, o *gap* foi 3,14%, ou 0,18 trilha adicional na média – o que equivale a duas trilhas adicionais considerando todo o conjunto. Mais uma vez, os tempos de execução são baixos, sendo o máximo menos de 0,02 segundos.

4.3. Instâncias Artificiais MOSP

Um terceiro conjunto de 150 instâncias artificiais MOSP grandes também foi considerado, disponível em `http://www.decom.ufop.br/marco/pesquisa/problem-instances/`. Estes casos foram gerados aleatoriamente, não possuindo nenhuma das estruturas específicas apontadas por Yanasse e Senne (2010) que possam facilitar a solução. As dimensões variam entre 150 linhas e colunas a 200 linhas e colunas. Na Tabela 3, *Random-m-n-d* identifica cada conjunto de 10 instâncias, em que m significa o número de redes, n o número de portas e d é o número de redes por porta, isto é, a demanda fixa para cada porta. Os valores médios de referência são as soluções comprovadamente ótimas ou o limitante inferior reportados por Chu e Stuckey (2009), já que nenhum dos métodos da literatura sobre o GMLP foram testados anteriormente usando estas instâncias.

Tabela 3. Resultados para as instâncias artificiais MOSP.

Instância	Solução	Heurística Proposta	
	Ótima	Resultado	Tempo (ms)
Random_150_150_2	25,90*	31,80	63,06
Random_150_150_4	61,60	67,90	243,66
Random_150_150_6	93,20*	98,70	577,61
Random_150_150_8	111,70*	118,80	769,03
Random_150_150_10	123,90*	129,80	970,50
Random_175_175_2	30,30	36,00	113,12
Random_175_175_4	73,70	81,30	377,44
Random_175_175_6	107,70	115,70	813,65
Random_175_175_8	128,30*	135,90	1313,20
Random_175_175_10	143,50*	150,40	1572,03
Random_200_200_2	36,00	44,00	144,65
Random_200_200_4	84,30	91,80	477,34
Random_200_200_6	121,50	129,00	1223,08
Random_200_200_8	147,10	155,10	1813,38
Random_200_200_10	162,80*	172,00	2000,83

*Solução ótima comprovada.

Das 150 instâncias contidas neste conjunto, apenas para 91 delas foram obtidas soluções comprovadamente ótimas pelo método exato de Chu e Stuckey (2009), desenvolvido para solução

do MOSP. Em uma arquitetura semelhante à utilizada nos experimentos deste trabalho, o tempo de execução médio foi de 3,78 horas por instância.

O *gap* total obtido pelo método proposto em relação aos valores de referência, é de 7,35%. Considerando que estas são instâncias maiores e sem estruturas especiais, tal marca é considerável. Os tempos de execução tiveram um aumento, rompendo a casa de dois segundos. Porém, apesar do aumento destes valores, eles ainda são considerados baixos e desejáveis, haja vista que se trata de uma heurística e dadas as dimensões das instâncias.

5. Conclusões e Trabalhos Futuros

O Problema de Determinação do Leiaute de Matrizes de Portas é um problema NP-difícil com aplicação industrial no projeto de circuitos VLSI que também modela com sucesso uma variedade de problemas de teoria dos grafos e de escalonamento. O objetivo do mesmo é, dada uma matriz de portas lógicas e redes que conectam estas portas, determinar uma permutação das portas (ou seja, um leiaute para matriz de portas) de tal modo que a área necessária para implementar um circuito eletrônico seja minimizada. Deste modo, estes circuitos podem ser produzidos de forma mais barata e mais rápida.

Este trabalho propõe uma heurística de duas fases. Na primeira fase o problema é modelado como um grafo com estrutura específica e usa a bem conhecida busca em largura para gerar um sequenciamento das redes do problema. Na segunda fase, a partir deste sequenciamento, a heurística encontra a solução para o problema de fato, uma permutação das portas. No intuito de melhorar a solução gerada, dois procedimentos de busca local são aplicados para compactar determinados conjuntos de componentes do circuito.

Experimentos computacionais abrangentes envolvendo 190 instâncias de três conjuntos diferentes, incluindo instâncias reais, foram apresentados. A heurística proposta foi capaz de igualar uma boa parte dos melhores resultados disponíveis, e melhorar alguns deles. Foram reportados também experimentos com um conjunto de instâncias inédito no contexto do problema abordado.

Os trabalhos futuros serão concentrados em melhorar a busca local utilizada, experimentando diferentes técnicas e também no desenvolvimento de um método exato para ajudar a orientar a busca heurística, transformando o método em uma heurística híbrida, ou *matheuristic*.

A heurística sugerida pode ser incorporada a métodos exatos, pois fornece um limite superior razoável para o problema, ou pode ser utilizada diretamente como uma opção viável para obter rapidamente boas soluções para o GMLP.

6. Agradecimentos

Esta pesquisa foi apoiada pelo Conselho Nacional de Desenvolvimento Científico e Tecnológico – CNPq (processo 441565/2014-0) e pela Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – CAPES (processo 5157/11).

Referências

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

Chen C. Y. R. e Hou C. Y. (1988), A new algorithm for cmos gate matrix layout. In Computer-Aided Design, 1988. ICCAD-88. *Digest of Technical Papers., IEEE International Conference on*, pages 138–141.

Chen S. J. e Hu, Y. H. (1990). GM-learn: an iterative learning algorithm for cmos gate matrix layout. *Computers and Digital Techniques, IEE Proceedings*, 137(4):301–309.

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

De Giovanni, L., Massi, G., Pezzella, F., Pfetsch, M. E., Rinaldi, G. e Ventura, P. (2013), A heuristic and an exact method for the gate matrix connection cost minimization problem. *International Transactions in Operational Research*, 20(5):627–643.

Deo, N., Krishnamoorthy, M. S. e Langston, M. A. (1987), Exact and approximate solutions for the gate matrix layout problem. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 6(1):79–84.

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

Hwang, D. K., Fuchs, W. K. e Kang, S. M. (1987). An efficient approach to gate matrix layout. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 6(5):802–809.

Kashiwabara T, 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 and Systems*, Tokyo, Japan, p.657–60.

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

Linhares, A. e Yanasse, H. H. e Torreão J. R. A. (1999), Linear gate assignment: a fast statistical mechanics approach. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 18(12):1750–1758.

Linhares, A. (1999), Synthesizing a predatory search strategy for VLSI layouts. *Evolutionary Computation, IEEE Transactions on*, 3(2):147–152.

Mendes, A. e Linhares, A. (2004), A multiple-population evolutionary approach to gate matrix layout. *Int. J. Systems Science*, 35(1):13–23.

Oliveira A. C. M. e Lorena, L. A. N. (2002), A constructive genetic algorithm for gate matrix layout problems. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 21(8):969–974.

SCOOP (Sheet Cutting and Process Optimization for Furniture Industry) Consortium (2009). Disponível em: http://www.scoop-project. net. Acessado em 19 de Abril de 2015.

Shahookar, K., Khamisani, W., Mazumder, P. e Reddy, S. M. (1994), Genetic beam search for gate matrix layout. *Computers and Digital Techniques, IEE Proceedings on*, 141(2):123–128.

Smith B. e Gent, I. *Constraint modeling challenge*. In Barbara Smith and Ian Gent, editors, The fifth workshop on modelling and solving problems with constraints, Edinburgh, Scotland, 2005. IJCAI.

Wing, O., Huang, S. e Wang, R. (1985), Gate matrix layout. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 4(3):220– 231.

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

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

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