Uma Heurística Adaptativa Aplicada ao Projeto de Circuitos Eletrônicos

Vinícius Gandra Martins Santos
Departamento de Ciência da Computação, Universidade Federal de Ouro Preto
Campus Morro do Cruzeiro, Ouro Preto, Minas Gerais, 35400-000, Brasil.
gandra.vinicius@gmail.com

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 um novo algoritmo para solução do Problema de Determinação de Leiaute de Matrizes de Portas (*Gate Matrix Layout Problem*, ou GMLP), um problema de aplicação prática na engenharia e indústria. Este problema combinatório é NP-Difícil e consiste em determinar a disposição física dos componentes de um circuito eletrônico que minimize a quantidade de trilhas necessárias para a implementação do mesmo. O problema abordado também pode ser utilizado para modelar com sucesso um grande conjunto de problemas de escalonamento em Pesquisa Operacional. Neste trabalho, é reportada pela primeira vez na literatura a aplicação do método metaheurístico Busca Adaptativa em Grandes Vizinhanças (*Adaptive Large Neighborhood Search*) para solução do GMLP. Os experimentos computacionais envolveram 395 instâncias da literatura e, para a maioria delas, o método apresentou resultados satisfatórios, com baixo *gap* e sendo capaz de igualar boa parte dos melhores resultados conhecidos na literatura.

PALAVRAS CHAVE. Leiaute de Matrizes de Porta, Heurística, Escalonamento.

Área Principal: IND, MH.

ABSTRACT

This paper presents a new algorithm to solve the Gate Matrix Layout Problem (GMLP), a problem with direct practical application in engineering and industry. This combinatorial problem is NP-Hard and consists on finding a physical arrangement of components of an electronic circuit that minimizes the number of necessary tracks in the printed circuit. The problem modeling is rather generic and can be used to successfully model a large set of scheduling problems in operations research. In this work, it is reported for the first time the application of the metaheuristic Adaptive Large Neighborhood Search as a mean to the solution for the GMLP. The computational experiments included 395 instances from the literature and, for the largest part of them, satisfactory results were generated, with low gap and also matching a good part of the best known results of the literature.

KEYWORDS. Gate Matrix Layout, Heuristics, Scheduling.

Main Area: IND, MH.

1. Introdução

Circuitos Integrados de Larga Escala (ou VLSI, do inglês *Very Large Scale Integration*) são comuns na microeletrônica e possuem um grande número de componentes dispostos em um mesmo *chip*, um substrato fino de material semicondutor, normalmente de silício. A disposição dos componentes eletrônicos influencia no tamanho do *chip* e consequentemente, seu custo. Tendo em vista que *chips* menores gastam menos matéria-prima e são mais rápidos por terem seus componentes mais próximos uns dos outros, é desejável o desenvolvimento de processos computacionais que permitam a obtenção de um leiaute físico otimizado dos componentes de forma a minimizar a área total do circuito eletrônico, desenvolvendo assim circuitos mais baratos, rápidos e compactos.

Os circuitos podem ser representados por uma matriz de portas programáveis. Este dispositivo utiliza duas dimensões, AND e OR, em que as colunas representam as *portas*, cujas eventuais conexões são feitas por um *fio* e representadas pelas linhas. Estes fios formam um conjunto de conexões entre portas que corresponde a uma *rede*. No circuito físico, as redes são implementadas por meio de *trilhas*. Diferentes trilhas devem ser utilizadas para alocar redes que utilizem uma mesma porta, no intuito de evitar sobreposições de redes. Deste modo, a matriz de portas é utilizada para elaborar circuitos lógicos combinatórios, compostos por n portas lógicas e m redes.

Na Figura 1(a) é apresentada uma matriz de portas contendo 6 portas enumeradas de P1 a P6 e 6 redes enumeradas de R1 a R6. As portas são representadas por linhas verticais enquanto as redes são representadas na horizontal por um conjunto de pontos conectados através de um fio. Note que o fio pode cruzar portas que não estão sendo conectadas, como pode ser observado na rede R2, em que o fio cruza as portas P3, P4 e P5, embora estas não façam parte da rede. Na tentativa de otimizar o leiaute do circuito, deve-se permutar a ordem das portas da matriz, tendo em vista que na produção de circuitos eletrônicos esta operação não altera a lógica do circuito, desde que, os fios não se cruzem. Na Figura 1(b) é apresentado um novo leiaute das portas da matriz, que por sua vez, permite a possibilidade de ser compactado. Em seguida, a Figura 1(c) apresenta o leiaute compactado, ou seja, diferentes redes alocadas a uma mesma trilha.

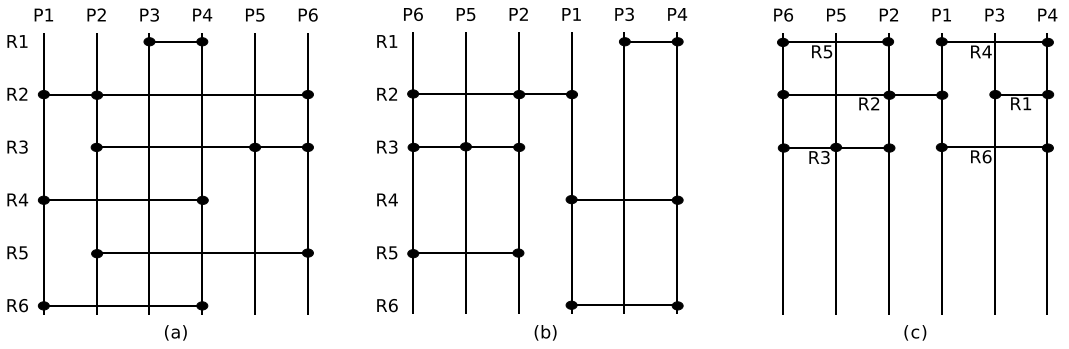

Figura 1: Matriz de portas (a) original, permutada (b), e compactada (c).

O Problema de Determinação de Leiaute de Matrizes de Portas (*Gate Matrix Layout Problem*, GMLP) consiste em determinar a permutação ótima de portas de modo a minimizar a quantidade de trilhas necessárias para implementar o circuito integrado correspondente e consequentemente minimizar a área e custo de produção do mesmo [Wing et al., 1985]. Como demostrando na Figura 1(a), são necessárias seis trilhas para acomodar todas as redes do circuito. No entanto, após a permutação das portas são necessárias apenas três, como pode ser visto na Figura 1(c).

Uma instância do GMLP pode ser representada pelo número m de redes, o número de portas n, e por uma matriz $m \times n$ $M = m_{ij} \rightarrow \{1, 0\}$, em que a entrada m_{ij} possui valor 1 se somente se a rede i incluir a a porta j, e valor 0 caso contrário. A Tabela 1(a) apresenta a matriz M do circuito representado anteriormente na Figura 1(a), em que suas conexões são representadas por valores 1, e o restante das entradas possui valor 0.

Tabela 1: Matriz binária (a) com adição dos 1s consecutivos em negrito e permutação π_1 (b) e π_2 (c).

	P1	*P2*	*P3*	*P4*	*P5*	*P6*
R1	0	0	1	1	0	0
R2	1	1	0	0	0	1
R3	0	1	0	0	1	1
R4	1	0	0	1	0	0
R5	0	1	0	0	0	1
R6	1	0	0	1	0	0

(a)

	P6	*P5*	*P2*	*P1*	*P3*	*P4*
R1	0	0	0	0	1	1
R2	1	**1**	1	1	0	0
R3	1	1	1	0	0	0
R4	0	0	0	1	**1**	1
R5	1	**1**	1	0	0	0
R6	0	0	0	1	**1**	1

(b)

	P6	*P1*	*P2*	*P5*	*P3*	*P4*
R1	0	0	0	0	1	1
R2	1	1	1	0	0	0
R3	1	**1**	1	1	0	0
R4	0	1	**1**	**1**	**1**	1
R5	1	**1**	1	0	0	0
R6	0	1	**1**	**1**	**1**	1

(c)

Uma solução para o GMLP consiste em uma permutação $\pi = [1, ..., n]$ das portas da matriz *M*, em que $\pi(j)$ indica a posição da porta *j* em π. A matriz permutação M^π tem suas portas sequenciadas de acordo com a permutação em π e é possuidora da propriedade dos 1s consecutivos, de maneira que $m^\pi_{ij} = 1$ se e somente se o fio da rede *i* incluir ou cruzar a porta *j*. Desta forma, todos os elementos com valor 0 entre os elementos de valor 1 das extremidades de cada rede são considerados também tendo o valor 1. Essa propriedade evita que redes sejam sobrepostas pois reserva espaço para os fios, o que é importante pois redes sobrepostas alteram a lógica do circuito.

Uma solução para o GLMP é avaliada de acordo com a soma de cada coluna da matriz M^π, em que a coluna de maior soma é chamada de *coluna crítica* ou *gargalo* e resulta no valor da solução. Na Tabela 1(b) e Tabela 1(c) duas soluções $\pi_1 = [6, 5, 2, 1, 3, 4]$ e $\pi_2 = [6, 1, 2, 5, 3, 4]$ são ilustradas contendo a propriedade dos 1s consecutivos que estão destacados em negrito. Observe que, a matriz compactada resultante da permutação de portas π_1 necessita apenas de 3 trilhas para ser implementada, tendo todas suas colunas como críticas, ou seja, com a maior soma é igual a 3. Enquanto isso, a matriz resultante de π_2 necessita de 5 trilhas, sendo P1 e P2 as colunas críticas.

Kashiwabara e Fujisawa [1979] (*apud* Linhares e Yanasse, 2002), provaram que o GMLP é NP-difícil pela redução do problema de aumento de grafos de intervalo. Linhares e Yanasse [2002] demonstraram que o GMLP possui diferentes problemas de formulação equivalente: 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*.

Para a solução do GMLP, este trabalho propõe implementar o método Busca Adaptativa em Grandes Vizinhanças (*Adaptive Large Neighborhood Search*, ALNS), uma metaheurística recente que utiliza buscas locais e perturbações para explorar uma porção ampla das possíveis soluções para problemas combinatórios. Este algoritmo é robusto, se adapta a várias características das instâncias e dificilmente fica preso em ótimos locais. Esta é a primeira aplicação do método ALNS ao GMLP reportada na literatura.

Este trabalho está organizado da seguinte forma: uma breve revisão da literatura é apresentada a Seção 2. O desenvolvimento está descrito na Seção 3, em que o método proposto está detalhadamente ilustrado. A Seção 4 demonstra os resultados obtidos pelo trabalho proposto considerando instâncias reais e artificiais da literatura. Por fim, a conclusão e propostas de trabalhos futuros são apresentados na Seção 5.

2. Revisão da Literatura

Métodos exatos, heurísticos e metaheurísticos são desenvolvidos para o GMLP desde a década de 80. Wing et al. [1985] representaram o problema por grafos de intervalo e propuseram um método heurístico para resolver o mesmo. Em seguida, Hwang et al. [1987] adaptaram o algoritmo *min-net-cut* ao problema e alcançaram resultados interessantes para a época. O mesmo algoritmo foi utilizado por Chen e Hou [1988], que representaram as conexões do circuito de forma dinâmica e criaram um algoritmo $O(nlogn)$, que analisa a dualidade dos circuitos semicondutores de metal-óxido complementar (*Complementary Metal-Oxide-Semiconductor*). Este algoritmo superou os

resultados para todas as 5 instâncias utilizadas retiradas da literatura, devidas a Hwang et al. [1987].

Na década de 90, princípios de inteligência artificial foram utilizadas por Hu e Chen [1990] e Chen e Hu [1990], que apresentaram os algoritmos Gm-Plan e Gm-Learn respectivamente. Destaca-se o Gm-Learn, que foi testado utilizando dezesseis circuitos retirados da literatura e que, superou os melhores resultados para oito circuitos. Para os outros oito circuitos restantes, seis tiveram seus resultados igualados ao melhor resultado conhecido e em somente dois circuitos, piores resultados foram encontrados.

Subsequentemente, Shahookar et al. [1994] apresentaram um método híbrido, *Genetic Beam Search*, implementando um algoritmo genético e um método *Beam Search*. Os melhores resultados obtidos foram em relação ao comprimento das redes. Ao final da década de 90, Linhares [1999] propôs uma estratégia de busca local, a Busca Predatória. Esta busca proposta determinou o melhor resultado até então para 4 instâncias, e alcançou os mesmos resultados que Hu e Chen [1990] e Chen e Hu [1990] para todas as outras instâncias da literatura. Além disto, o número mínimo de trilhas foi alcançado em 12 das 25 instâncias utilizadas. No mesmo ano, Linhares et al. [1999], conseguiram superar os resultados do Gm-Plan e Gm-Learn em três circuitos, sendo superado em apenas um caso e tendo igualado todos os outros resultados anteriores com a Otimização Microcanônica (OM), um procedimento derivado do *Simulated Annealing*.

Posteriormente, algoritmos memético e construtivos foram explorados por alguns autores. Mendes et al. [2002] e Mendes e Linhares [2004] propuseram algoritmos meméticos, com destaque para este ultimo. Com o algoritmo memético e outros dois métodos foi possível alcançar todos os melhores resultados da literatura até o momento e apresentar um bom desempenho quando comparado com algoritmos anteriores, como Gm-Plan, Gm-Learn, *simulated annealing* e heurísticas construtivas. Outra abordagem foi o algoritmo genético construtivo (CGA) proposto por de Oliveira e Lorena [2002], que utiliza esquemas para construir soluções que otimizam o número de trilhas e o comprimento das redes. O CGA foi comparado com o OM [Linhares et al., 1999] e obteve os mesmos resultados (em número de trilhas) para todas as instâncias em um tempo menor.

Recentemente, heurísticas foram utilizadas por Santos e Carvalho [2015] e Gonçalves et al. [2016]. Santos e Carvalho [2015] propuseram uma heurística que 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. Os testes foram feitos utilizando 190 instâncias, entre elas instâncias inéditas no contexto do problema. A heurística foi capaz de igualar boa parte dos melhores resultados disponíveis, e melhorar alguns deles. Gonçalves et al. [2016] propuseram uma metaheurística para o MOSP composta por um algoritmo genético de chaves aleatórias viciadas (*Biased Radom-Key Genetic Problem*, BRKGA), e uma busca local. Atualmente, este método representa o estado da arte relacionado a metaheurísticas aplicada ao problema. Os testes aplicados ao BRKGA envolvem mais de 6000 instâncias existentes na literatura e, de acordo com os resultados reportados, o BRKGA foi capaz de encontrar todos os melhores resultados conhecidos.

3. Método Proposto

Ropke e Pisinger [2006] introduziram a metaheurística Busca Adaptativa em Grandes Vizinhanças (*Adaptive Large Neighborhood Search*, ALNS). Este método utiliza um conjunto de diferentes heurísticas para buscas locais simples e rápidas que competem entre si para modificar a solução corrente explorando suas vizinhanças através de movimentos de remoção e inserção de elementos. Inicialmente, deve-se escolher um conjunto N^- de heurísticas de remoção de elementos que intensificam e diversificam a busca removendo q elementos da solução por iteração. É necessário também escolher um conjunto N^+ de heurísticas de inserção de elementos que possa construir uma solução completa a partir de uma solução parcial. Essas heurísticas são geralmente baseadas em métodos gulosos de bom desempenho para o problema em questão.

Uma busca local é composta por uma heurística de remoção e uma heurística de inserção de elementos. O ALNS, por sua vez, possui diversas dessas heurísticas podendo assim combiná-

las para criar variadas buscas locais que induzem diferentes vizinhanças, sendo assim, é de suma importância que a escolha de qual busca local utilizar seja feita de forma eficiente.

O ALNS utiliza um método estatístico baseado nas iterações anteriores do algoritmo para decidir a prioridade da aplicação das heurísticas. A ideia é observar o desempenho de cada heurística ao longo das iterações e atribuir pontos para cada uma de acordo com o desempenho da mesma naquela iteração, tendo em vista que quanto mais a vizinhança $N_i \in \{N^+ \cup N^-\}$ contribui para uma solução, maior será a quantidade de pontos atribuídos à mesma e consequentemente, maior a chance de ser selecionada em iterações futuras. A probabilidade da vizinhança N_i ser selecionada é dada pela divisão dos pontos da vizinhança N_i, denotados por r_i, pelo somatório dos pontos de todas as vizinhanças.

O ALNS seleciona uma heurística de remoção e uma outra de inserção a cada iteração. A seleção é feita através do método da *roleta*. A roleta é representada por um intervalo $R = [0...1] \in \mathbb{R}$, em que cada vizinhança i recebe uma fatia da roleta proporcional à sua probabilidade $P(N_i)$ de ser escolhida. Sendo assim, quanto maior for a probabilidade da vizinhança ser escolhida, maior será o a sua fatia de intervalo na roleta e consequentemente maior a chance da vizinhança ser selecionada. Para que a seleção ocorra, um número $\nu \in (0, 1)$ é aleatoriamente escolhido e os valores das fatias são sequencialmente acumulados tal que a heurística correspondente à fatia cujo valor acumulado extrapolar o valor de ν é selecionada.

Estes pontos utilizados para a criação da roleta são ajustados de forma adaptativa. A execução do ALNS é dividida em blocos de iterações de tamanho θ chamados *segmentos*, durante os quais os pontos das heurísticas são cumulativamente calculados. Por exemplo, se a metaheurística for executada por 1000 iterações podemos ter 10 segmentos em que $\theta = 100$. Sendo $\overline{r}_{i,j}$ os *pontos observados* da heurística i no segmento j, podemos classificá-los em três categorias: σ_1, quando a última remoção ou inserção resultou no melhor resultado até o momento; σ_2, quando a última remoção ou inserção resultou em uma solução cujo custo seja menor que o da solução corrente; e σ_3, quando a última remoção ou inserção resulta em uma solução que é aceita por um critério de aceitação, porém com o custo maior que o da solução corrente.

A terceira categoria tem em vista que soluções piores, quando viáveis, contribuem para uma perturbação na solução corrente e devem ser levadas em consideração. No final de cada segmento, os pontos acumulados para cada heurística passam por um processo de *suavização*, calculado conforme a Equação 1, em que a_i é o número de vezes que a heurística i foi chamada durante o segmento j. O fator de reação $\rho \in (0, 1)$ controla o quão rápido o ajuste de pontos do algoritmo reage às mudanças nos pontos de cada heurística. Quando $\rho = 1$ a roleta é baseada somente nos pontos do segmento imediatamente anterior, por outro lado, quando $\rho < 1$, os pontos dos segmentos anteriores são todos levados em consideração. Caso a heurística i não tenha sido chamada nenhuma vez durante o segmento, sua pontuação permanecerá a mesma do segmento imediatamente anterior. Esta estratégia está intimamente relacionada com os conceitos de memória de curto e longo prazo, comuns em otimização combinatória.

$$r_{i,j+1} = \begin{cases} r_{i,j} & \text{Se } a_i = 0. \\ \rho\frac{\overline{r}_{i,j}}{a_i} + (1-\rho)r_{i,j} & \text{Se } a_i \neq 0. \end{cases} \quad (1)$$

Cada nova solução gerada pelo ALNS com o custo maior que o da solução corrente passa por um critério de aceitação dinâmico dado por um método similar ao *simulated annealing*, que restringe gradativamente a qualidade das soluções. Desta forma, ao longo das iterações, as soluções para serem aceitas devem possuir valores cada vez melhores. Uma solução π' gerada a partir de outra solução π é aceita com probabilidade calculada de acordo com a Equação 2. Originalmente, utiliza-se uma taxa de resfriamento exponencial, em que a temperatura T, sendo $T > 0$, decresce de acordo com a expressão $T = T \times c$, no qual $c \in \{0...1\}$ representa a taxa de resfriamento. A temperatura inicial é definida por $T_{start} = 0,05 f(\pi_0)/\ln 2$, em que uma solução corrente 5% pior

que a solução inicial π_0 é aceita com probabilidade de 50%.

$$e^{-(f(\pi')-f(\pi))/T} \tag{2}$$

O ALNS é estruturado como apresentado no Algoritmo 1. O método parte de uma solução viável e a considera a melhor até então (linhas 1 e 2). Em seguida, um laço de repetição (linhas 3 a 22) é executado enquanto um determinado critério de parada não for atendido. Ao entrar no laço, uma heurística de remoção e outra de inserção são selecionadas através do método da roleta (linha 4), e em seguida aplicadas à solução atual π para gerar uma nova solução π' (linha 5). Posteriormente, o resultado de π' é comparado com a solução corrente π (linhas 6 a 13) e, caso seja melhor, se torna a nova solução corrente (linha 7). Uma segunda comparação é feita para checar se a solução corrente π é também melhor que a melhor solução π_{best} (linhas 9 a 12), se for melhor, π_{best} é atualizado (linha 10). No entanto, se π' possuir valor igual ou pior que a solução corrente, deve-se avaliá-la de acordo com o critério de aceitação (linhas 14 a 17) e, caso seja aceita, se torna a nova solução corrente (linha 15). Após a análise da nova solução, os pontos das vizinhanças são alterados (linha 18) conforme o valor recebido na linha 8, 11 ou 16. Ao final de cada segmento (linha 19), os pontos das vizinhanças são suavizados (linha 20) e a probabilidade nas roletas recalculada (linha 21). Ao final, a melhor solução π_{best} é retornada (linha 24).

Algoritmo 1: ALNS.

```
1  Input: solução viável π;
2  π_best ← π;
3  while critério de parada não for atingido do
4      Escolha uma vizinhança de remoção N⁻ e uma vizinhança de inserção N⁺
         usando a roleta de seleção baseado nos pontos obtidos anteriormente {r_j};
5      Gere uma solução nova π' a partir de π usando a heurística de remoção e
         inserção escolhidas;
6      if f(π') < f(π) then
7          π ← π';
8          sigma ← σ_2;
9          if f(π) < f(π_best) then
10             π_best ← π;
11             sigma ← σ_1;
12     else if π' atender o critério de avaliação then
13         π ← π';
14         sigma ← σ_3;
15     Atualize os pontos r_j de N⁻ e N⁺ conforme sigma;
16     if contador de interação for múltiplo de θ then
17         Suavize os pontos r_j acumulados;
18         Atualize a probabilidade das vizinhanças na roleta conforme nova
             pontuação;
19 return π_best;
```

A solução inicial a qual o método parte é gerada conforme descrito por Santos e Carvalho [2015]. Os autores propuseram os métodos de pré-processamento de portas dominadas, representação por grafos, busca em largura e sequenciamento de portas, para a geração de uma solução inicial.

Tabela 2: Matriz binária (a) com adição dos 1s consecutivos em negrito e permutação π_1 (b) e π_2 (c).

	P1	*P2*	*P3*	*P4*	*P5*
R1	0	0	0	1	1
R2	1	**1**	**1**	**1**	1
R3	0	0	1	**1**	1
R4	0	0	1	1	0
R5	1	0	0	0	0
R6	0	1	0	0	0

(a)

	P1	*P2*	*P5*	*P3*	*P4*
R1	0	0	1	**1**	1
R2	1	**1**	1	0	0
R3	0	0	1	1	0
R4	0	0	0	1	1
R5	1	0	0	0	0
R6	0	1	0	0	0

(b)

	P2	*P4*	*P3*	*P5*	*P1*
R1	0	1	**1**	1	0
R2	0	0	0	1	1
R3	0	0	1	1	0
R4	0	1	1	0	0
R5	0	0	0	0	1
R6	1	0	0	0	0

(c)

3.1. Vizinhanças de Remoção

Cada vizinhança de remoção recebe uma solução, representada por uma sequência de portas, e removem portas da solução obtendo dois subconjuntos de portas: um representa a solução parcial e o outro as portas que devem ser inseridas na mesma por alguma vizinhança de inserção. A Tabela 2 será utilizada como referência dos exemplos para algumas das seções seguintes.

3.1.1. Remoção de Colunas Críticas

A primeira vizinhança de remoção calcula e então remove todas as colunas críticas da solução. É plausível acreditar em uma melhora da solução ao remover todas suas portas de maior valor para depois inserí-las em uma melhor posição. O exemplo da Tabela 2(a) tem $P4$ como coluna crítica, portanto, esta deve ser removida.

3.1.2. Remoção de Uns Consecutivos em Colunas Críticas

A segunda vizinhança de remoção identifica as colunas críticas da solução e seus respectivos 1s consecutivos. Devem ser removidas portas que causem 1s consecutivos na coluna crítica, ou seja, portas que estão ativas nas extremidades das redes em que se encontra um determinado 1 consecutivo da coluna crítica. Este método tem a intenção de diminuir a quantidade de 1s consecutivos nas colunas críticas para reduzir o valor da solução. O ALNS dispõe de uma implementações que remove portas à esquerda da coluna crítica, e outra que remove portas à direita.

Na Tabela 2(a), $P4$ é a coluna crítica. As portas $P1$ e $P5$ compõe a rede $R2$ e as portas $P3$ e $P5$ compõe a rede $R3$, redes que cruzam $P4$ causando 1s consecutivos (em negrito). Sendo assim, a porta $P5$ deve ser removida, tendo em vista que o método escolhido remove portas à direita. De forma análoga, as portas $P1$ e $P3$ seriam removidas no caso de remoção à esquerda.

3.1.3. Remoção de Uns Consecutivos em Linhas

A terceira vizinhança é análoga a anterior, no entanto, a análise passa a ser por linhas. De forma aleatória, uma linha é selecionada e tem seus 1s consecutivos analisados. Então, as portas que causam os 1s consecutivos são removidas da solução. Na Tabela 2(a), se a linha $R2$ for sorteada, a porta $P1$ é removida caso a vizinhança remova portas à esquerda do bloco de 1s consecutivos, e a porta $P5$ é removida caso a linha $R2$ ou $R3$ sejam selecionadas e a remoção seja pela direita.

3.1.4. Remoção Aleatória

A quarta vizinhança seleciona q portas de forma aleatória para serem removidas. Essa vizinhança tem o intuito de perturbar a solução, tendo em vista que utiliza um critério aleatório para a remoção. O valor de q é calculado conforme a Equação 3, proposta por Pereira et al. [2015].

$$q = \lfloor m - \sqrt{(1-u)(m-1)^2} + 0.5 \rfloor \tag{3}$$

A Equação 3 segue uma distribuição triangular e seleciona aleatoriamente um número entre $[1, m]$, em que m é o número de portas e u é uma variável aleatória entre $[0, 1]$. Essa operação pode ter grande impacto quando o número de portas é grande.

3.1.5. Remoção de Portas Relacionadas

A quinta vizinhança remove portas que estão de alguma forma relacionadas. A similaridade neste caso é medida através da quantidade de 1s consecutivos. Uma porta é aleatoriamente selecionada, então as portas que possuem o mesmo número de 1s consecutivos que a porta já selecionada são também removidas. Com este método acredita-se que portas similares tem maiores chances de serem realocadas com sucesso. Na Tabela 2(a), se a porta $P2$, que possui um único 1 consecutivo (em negrito), for selecionada para ser removida, a porta $P3$ o será também por possuir o mesmo número de 1s consecutivos.

3.2. Vizinhanças de Inserção

Cada vizinhança de inserção recebe uma solução parcial e um conjunto de portas γ para ser inserido na solução. Cada porta do conjunto é selecionada aleatoriamente e inserida uma por vez na solução. Os métodos de inserção não aceitam pioras, sendo assim, uma porta é inserida na solução somente se a mesma não piorar o valor da função objetivo. Ao final, retorna-se uma solução com o mesmo ou um melhor valor da função objetivo em relação à solução original. A Tabela 2(b) e Tabela 2(c) serão utilizadas como referência dos exemplos para algumas das seções seguintes.

3.2.1. Inserção Aleatória

A primeira vizinhança de inserção insere cada porta em uma posição aleatória. Esta inserção em especial não analisa a função objetivo, deixando para o ALNS decidir se a solução deve ser aceita ou não. A utilização desta inserção introduz diversidade à solução, que posteriormente pode ser aceita pelo ALNS mesmo que tenha um valor de solução pior.

3.2.2. Inserção Limitada por Coluna

A segunda vizinhança insere as portas na primeira posição que não piore a função objetivo, no entanto, o método recebe uma porta limite e uma direção para a inserção. Dado o método de remoção de 1s consecutivos em linhas (Seção 3.1.3), se a linha $R2$ for selecionada e a porta $P5$ for removida, temos: solução parcial $\pi = [1, 2, 3, 4]$, $P4$ como porta limite e esquerda como direção de inserção. Deste modo, a porta $P5$ deve ser inserida antes da porta $P4$. Após a inserção a Tabela 2(b) demonstra um resultado possível, $\pi_1 = [1, 2, 5, 3, 4]$, em que a porta $P5$ foi inserida imediatamente antes da porta $P3$, melhorando a função objetivo de 4 para 3 trilhas.

3.2.3. Inserção na Melhor Posição

A terceira vizinhança considera a função objetivo para inserir cada porta na melhor de todas as posições possíveis. Considere o método de remoção de 1s consecutivos em colunas críticas (Seção 3.1.2), em que à esquerda da coluna crítica $P4$ as portas $P1$ e $P3$ foram removidas. A Tabela 2 (c), demonstra uma solução possível $\pi_2 = [2, 4, 3, 5, 1]$, em que as portas $P1$ e $P3$ foram inseridas na melhor posição possível e a função objetivo diminuiu de 4 para 3 trilhas.

3.2.4. Inserção por Arrependimento

A quarta e última vizinhança de inserção tenta melhorar a heurística da melhor posição adicionando um mecanismo de *lookahead*. Ao selecionar de forma aleatória as portas para inserção, as portas mais difíceis podem ser inseridas por último, enquanto o ideal seria inserí-las o mais rápido possível. Desta forma, o método proposto não utiliza critério aleatório para selecionar as portas que devem ser inseridas, ao invés disso, as portas são ordenadas de acordo com o valor de *arrependimento* e então inseridas uma a uma na ordem estabelecida.

Para calcular o arrependimento deve-se primeiro encontrar a melhor e a segunda melhor posição de inserção de cada porta. Considere Δf_i^q como o custo da função objetivo ao inserir a porta i na q-ésima melhor posição. Por exemplo, Δf_i^2 indica o custo de inserir a porta i na segunda melhor posição. Desta forma, a cada iteração, a heurística de inserção escolhe a porta a ser inserida de acordo com a Equação 4, em que o arrependimento é a diferença no custo de inserir a porta na melhor posição possível ou na segunda melhor.

$$i = max_{i \in \gamma}(\Delta f_i^2 - \Delta f_i^1). \tag{4}$$

Por exemplo, considere as portas $P3$, $P5$ e $P6$ para serem inseridas, com valor de arrependimento 5, 10 e 7, respectivamente. As portas devem ser inseridas na melhor posição possível seguindo a ordem de inserção $P5$, $P6$ e por fim $P3$. Em outras palavras, deseja-se inserir primeiro a porta que causa o maior arrependimento caso não seja inserida naquele momento.

4. Experimentos Computacionais

O ambiente computacional adotado consiste em um computador com processador *Intel Core i5* de 3.0 GHz com 8 GB RAM, utilizando o sistema operacional Ubuntu 15.10. O método proposto foi codificado utilizando a linguagem C++ e compilado com g++ 5.2.1. Os parâmetros usados pelo ALNS e seus respectivos valores são: σ_1, σ_2 e σ_3 com valores 15, 25 e 5 respectivamente, taxa de resfriamento c igual a 0,23, fator de reação ρ igual a 0,66, número de iterações igual a 800 e tamanho de cada segmento θ igual a 60.

Quatro conjuntos de instâncias foram utilizados e divididos em diferentes seções. Devido ao fator de aleatoriedade do método, foram realizadas 10 execuções independentes para cada uma das instâncias.

As tabelas a seguir estão organizadas da seguinte maneira: a coluna *Instância* indica o nome de cada instância e as colunas seguintes *m* e *n* representam o número de redes e portas respectivamente e os resultados ótimos são apresentados na coluna *OPT*; o melhor resultado obtido pelo ALNS é representado na coluna *S**, a média dos resultados encontrados nas 10 execuções do ALNS são representados na coluna *S* e a solução inicial é apresentada na coluna S_0; por fim, o tempo de execução, expresso em segundos, se encontra na coluna *T*. Para melhor efeito de comparação de resultados, a distância percentual entre a solução ótima e a encontrada pelo ALNS é também apresentada, calculada como $gap = 100 \times (S^* - OPT)/OPT$. Por fim, o desvio padrão entre as diferentes soluções encontradas para cada instância é indicado na coluna σ. Devido a restrições de espaço, as tabelas apresentam os resultados para as instâncias cujo resultado ótimo não foi obtido pelo ALNS.

4.1. Instâncias Reais VLSI

O primeiro conjunto de instâncias foi introduzido por Hu e Chen [1990]. Estas instâncias são reais e oriundas de empresas asiáticas. O conjunto possui 25 instâncias com matrizes de dimensões que variam entre 5×7 e 202×141. A Tabela 3 apresenta os resultados para este conjunto.

Tabela 3: Instâncias reais VLSI.

Instância	*m*	*n*	*OPT*	*S**	*S*	S_0	*T*	*gap*	σ
v4470	47	37	9	12	12,8	13	22,37	33,33	0,40
w3	70	84	18	30	31,7	34	66,27	66,67	1,00
w4	141	202	27	52	52,8	53	896,37	92,59	0,40
x0	48	40	11	15	16,3	18	15,32	36,36	0,78

O ALNS obteve 84% das soluções ótimas. Apesar disto, o método não obteve bons resultados para instâncias maiores, como por exemplo w3 e w4. Estas instâncias influenciam fortemente o *gap* médio de 9,16% e o tempo médio de execução de 40,62 segundos. Ao analisar as outras instâncias nota-se para a maioria que o *gap* é 0,00% e o tempo de execução é menor do que 3 segundos, ratificando a influência das instâncias w3 e w4 nos valores médios. A solução média não está distante da melhor solução obtida pelo ALNS, gerando um baixo desvio padrão. De fato, para 80% das instâncias deste grupo, as soluções médias igualaram a melhor solução obtida, ou seja, todas as execuções do ALNS obtiveram a mesma solução, demonstrando a robustez do método.

4.2. Instâncias SCOOP

O segundo conjunto possui instâncias reais para o problema de Minimização de Pilhas Abertas (um problema de formulação equivalente ao GMLP). Este conjunto, fornecido pelo SCOOP

Consortium[1] contém 24 instâncias selecionadas, com dimensões significativas. Os resultados são apresentados na Tabela 4.

Tabela 4: Instâncias reais SCOOP. Valores em negrito indicam soluções ótimas.

Instância	m	n	OPT	S^*	S	S_0	T	gap	σ
A_FA+AA-1	37	107	12	15	16,50	18	22,88	25,00	0,81
A_FA+AA-13	37	134	17	21	22,30	27	24,78	23,53	1,00
A_FA+AA-6	21	79	13	14	14,30	17	3,28	7,69	0,46
A_FA+AA-8	28	82	11	13	13,90	17	9,09	18,18	0,70
B_REVAL-145	49	60	7	8	8,90	9	18,76	14,29	0,30

O ALNS obteve boas soluções para este conjunto de instâncias com baixo tempo de execução, em torno de 4,24 segundos. Três das instâncias se destacaram por possuírem um tempo médio de execução maior do que 18 segundos e *gap* alto se comparado às outras instâncias. Enquanto isso, para a maior parte das outras instâncias o método encontrou solução ótima em menos de 3 segundos. Das 24 instâncias, o método obteve solução ótima para 79,16% delas com um *gap* médio de 3,70%. Os resultados se diferenciam pouco ao longo das execuções, desta forma, observa-se um desvio padrão pequeno (0,22%). Além disso, 62,50% das soluções médias se igualaram à melhor solução obtida pelo ALNS, ou seja, obtiveram o mesmo valor em 10 execuções independentes, reforçando a ideia de robustez do método.

4.3. Instâncias do *First Constraint Modeling Challenge*

O terceiro conjunto de 46 instâncias MOSP foi proposto para o *First Constraint Modeling Challenge* [Smith e Gent, 2005]. A escolha deste subconjunto foi feita levando em conta as dimensões das instâncias e a ausência de estruturas especiais que podem facilitar a solução. A Tabela 5 apresenta os resultados e possui uma coluna i adicional para indicar a quantidade de instâncias em cada grupo. A média dos resultados é apresentada quando i for maior que 1.

Tabela 5: Instâncias selecionadas do *First Constraint Modeling Challenge*.

Instância	m	n	i	OPT	S^*	S	S_0	T	gap	σ
GP1	50	50	4	38,75	39,50	39,73	39,75	101,62	1,94	0,07
GP2	100	100	4	76,25	77,00	77,33	77,50	3826,62	0,98	0,20
Shaw	20	20	25	13,68	13,72	13,96	14,68	1,64	0,29	0,23
SP4-2	50	50	1	19	24	24	27	23,57	26,32	0,75
SP4-3	75	75	1	34	40	40	40	168,58	17,65	0,00
SP4-4	100	100	1	53	62	62	63	532,67	16,98	0,30

Para este conjunto foi obtida a solução ótima para 39 das 46 instâncias, com *gap* médio de 5,35%. Apesar do tempo de execução médio ser de 391,62 segundos, o mesmo é baixo para a maioria dos conjuntos, menor do que 40 segundos. O tempo é fortemente influenciado pelo grupo de instâncias GP2 e pelas instâncias SP4-3 e SP4-4. O *gap* médio é também fortemente influenciado pelas 3 últimas instâncias, as quais possuem *gap* individual superior a 15%. Ao longo das 10 execuções os resultados diferenciam pouco, dessa forma, o método apresentou um desvio padrão de 0,17%. Sendo assim, conclui-se mais uma vez a respeito da robustez do método.

4.4. Instâncias Faggioli e Bentivoglio

O quarto conjunto de instâncias, proposto por Faggioli e Bentivoglio [1998], é composto por 300 instâncias artificiais para o MOSP. Os problemas foram agrupados em grupos de 10 instâncias cada de acordo com o número de linhas e colunas. A Tabela 6 apresenta os resultados obtidos, em que os resultados apresentados são a média das 10 instâncias de cada grupo.

[1]Disponível em http://www.scoopproject.net

Tabela 6: Instâncias agrupadas de Faggioli e Bentivoglio.

m	n	OPT	$S*$	S	S_0	T	gap	σ
15	20	7,20	7,40	7,65	9,80	0,59	2,78	0,26
15	30	7,30	7,70	7,98	9,80	0,82	5,48	0,32
20	20	8,50	8,60	9,02	10,10	1,21	1,18	0,34
20	30	8,80	9,50	10,11	12,30	1,78	7,95	0,47
20	40	8,50	9,10	9,78	12,80	2,08	7,06	0,48
20	50	7,90	8,30	8,98	11,40	2,42	5,06	0,39
25	20	9,80	10,70	10,95	12,30	2,45	9,18	0,33
25	30	10,50	11,70	12,64	14,80	3,56	11,43	0,59
25	40	10,30	11,80	12,33	14,80	3,93	14,56	0,49
25	50	10,00	11,40	12,24	15,20	4,69	14,00	0,60
30	20	11,10	11,90	12,61	13,20	4,57	7,21	0,49
30	30	12,20	13,60	14,47	15,70	6,07	11,48	0,51
30	40	12,10	13,90	14,88	16,50	6,96	14,88	0,60
30	50	11,20	14,00	14,81	16,40	7,01	25,00	0,58
40	20	13,00	14,10	14,49	14,90	12,08	8,46	0,26
40	30	14,50	16,40	17,44	17,90	16,81	13,10	0,45
40	40	14,90	18,20	19,19	19,90	17,49	22,15	0,46
40	50	14,60	18,70	19,52	20,30	18,86	28,08	0,45

Neste conjunto de instâncias, o ALNS obteve um *gap* médio relativamente baixo, 6,97%, e obteve dentre as 300 instâncias 54% (ou 164) das soluções ótimas. A média de tempo de execução para cada instância foi de aproximadamente 4 segundos. O desvio padrão para este conjunto de instâncias ainda continua baixo (0,29%), desta forma, conclui-se que o método se prova também robusto para este conjunto de instâncias.

5. Conclusão e Trabalhos Futuros

Neste trabalho foi apresentada uma abordagem para o problema de Determinação de Leiaute de Matrizes de Portas (ou GMLP, do inglês *Gate Matrix Layout Problem*), um problema NP-Difícil com ampla aplicação industrial que visa otimizar a área de circuitos eletrônicos para que estes possam ser mais baratos, rápidos e compactos. A abordagem reportada consiste em uma implementação da metaheurística Busca Adaptativa em Grandes Vizinhanças (ou ALNS, do inglês *Adaptive Large Neighborhood Search*). É importante ressaltar que esta é a primeira aplicação reportada do ALNS para solução do GMLP. Os experimentos computacionais envolveram 395 instâncias de 4 diferentes conjuntos da literatura, incluindo instâncias reais e artificiais. Os resultados obtidos foram comparados com as soluções ótimas da literatura. Os dados reportados revelam taxas de soluções ótimas que podem chegar até 84%, com maior *gap* médio de 9,16%. Os trabalhos futuros serão concentrados em aprimorar as vizinhanças utilizadas pelo ALNS, no intuito de obter melhores resultados, e também otimizar o método em busca de melhores tempos computacionais.

6. Agradecimentos

Esta pesquisa foi apoiada pelo Conselho Nacional de Desenvolvimento Científico e Tecnológico, pela Fundação de Apoio à Pesquisa do Estado de Minas Gerais e pela Universidade Federal de Ouro Preto.

Referências

Chen, C. 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 Int. Conf. on*, p. 138–141. IEEE.

Chen, S.-J. e Hu, Y. H. (1990). Gm-learn: an iterative learning algorithm for cmos gate matrix layout. *IEE P-COMPUT DIG T*, 137(4):301–309.

de Oliveira, A. C. M. e Lorena, L. A. N. (2002). A constructive genetic algorithm for gate matrix layout problems. *IEEE T COMPUT AID D*, 21(8):969–974.

Faggioli, E. e Bentivoglio, C. A. (1998). Heuristic and exact methods for the cutting sequencing problem. *Eur. J. of Oper. Res.*, 110(3):564–575.

Gonçalves, J. F., Resende, M. G., e Costa, M. D. (2016). A biased random-key genetic algorithm for the minimization of open stacks problem. *Int. Trans. in Oper. Res.*, 23(1-2):25–46.

Hu, Y. H. e Chen, S.-J. (1990). Gm plan: a gate matrix layout algorithm based on artificial intelligence planning techniques. *IEEE T COMPUT AID D*, 9(8):836–845.

Hwang, D. K., Fuchs, W. K., e Kang, S. M. (1987). An efficient approach to gate matrix layout. *IEEE T COMPUT AID D*, 6(5):802–809.

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 and Systems, Tokyo, Japan, July. p. 657–60.*

Linhares, A. (1999). Synthesizing a predatory search strategy for vlsi layouts. *IEEE T EVOLUT COMPUT*, 3(2):147–152.

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

Linhares, A., Yanasse, H. H., e Torreão, J. R. (1999). Linear gate assignment: a fast statistical mechanics approach. *IEEE T COMPUT AID D*, 18(12):1750–1758.

Mendes, A., França, P., Moscato, P., e Garcia, V. (2002). Population studies for the gate matrix layout problem. In *Ibero-American Conference on Artificial Intelligence*, p. 319–328. Springer.

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

Pereira, M. A., Coelho, L. C., Lorena, L. A., e De Souza, L. C. (2015). A hybrid method for the probabilistic maximal covering location–allocation problem. *Comp. & Oper. Res.*, 57:51–59.

Ropke, S. e Pisinger, D. (2006). An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation science*, 40(4):455–472.

Santos, J. V. M. e Carvalho, M. A. M. (2015). Uma heurística aplicada à produção em microeletrônica. In *Anais do XLVII Simpósio Brasileiro de Pesquisa Operacional*.

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

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

Wing, O., Huang, S., e Wang, R. (1985). Gate matrix layout. *IEEE T COMPUT AID D*, 4(3): 220–231. ISSN 0278-0070.