UNIVERSIDADE FEDERAL DE OURO PRETO
Instituto de Ciências Exatas e Biológicas – Departamento de Ciência da Computação
DECOM

ALGORITMOS HEURÍSTICOS E METAHEURÍSTICOS PARA A MINIMIZAÇÃO DO CUSTO DE CONEXÕES EM MATRIZES DE PORTAS

Relatório Final, referente ao período 04/14 à 02/15 apresentado à Universidade Federal de Ouro Preto, como parte das exigências do programa de iniciação cientifica / Fundação de Amparo à Pesquisa do Estado de Minas Gerais (FAPEMIG).

Estudante: João Vitor Mascarenhas dos Santos
Orientador: Marco Antonio Moreira De Carvalho

Ouro Preto - Minas Gerais - Brasil
Fevereiro de 2015

Algoritmos Heurísticos e Metaheurísticos para a minimização do custo de conexões em matrizes de portas

O presente projeto de pesquisa propõe o desenvolvimento de abordagens heurísticas e metaheurísticas para a otimização de leiautes de matrizes de portas, um problema combinatório relacionado à dobradura de matrizes lógicas programáveis no projeto de circuitos eletrônicos integrados em larga escala, como por exemplo, processadores de computadores e celulares. Nestes circuitos eletrônicos, portas são conectadas entre si por trilhas, e no intuito de minimizar os custos destas conexões e a área destes circuitos, busca-se um leiaute otimizado das portas e das trilhas que os conectam. Desta forma, é possível produzir componentes mais baratos, mais rápidos e mais compactos. O problema objeto de estudos do presente projeto possui aplicação prática direta na indústria microeletrônica, um ramo da eletrônica voltado à integração de circuitos eletrônicos, estando presente na informática, nas telecomunicações, nos controles de processos industriais, na automação dos serviços bancários e comerciais e nos bens de consumo.

A heurística proposta baseia-se em uma busca no grafo correspondente ao circuito e é comparada a dois métodos da literatura em experimentos computacionais abrangentes que consideraram quatro conjuntos de instâncias disponíveis publicamente de casos reais e artificiais. Os resultados apresentados mostram que o método proposto, em um curto espaço de tempo, foi capaz de igualar boa parte dos melhores resultados disponíveis, melhorar alguns deles e obter uma pequena diferença quando comparado com as melhores soluções e limites inferiores.

Índice

Introdução

No campo da microeletrônica, um ramo da eletrônica voltado à integração de circuitos eletrônicos promovendo uma miniaturização dos componentes em escala microscópica, os elementos que a compõe são fabricados em proporções de mícrons ou mesmo nanômetros, transformando-se em parte da área de nanotecnologia. A combinação dos componentes em um mesmo projeto é usualmente nomeada de circuito integrado (CI), ou também, *chip*, dentre eles estão os processadores, memórias de computadores, entre outros.

A partir de *wafers*, placas de silício, sobre as quais são construídas estruturas laminares através de um método chamado fotolitografia, são produzidos os circuitos integrados. Comumente, vários circuitos são produzidos com um mesmo *wafer* por vez a fim de utilizar todo espaço disponível. Sendo assim, o custo para fabricação do *chip* será o custo de produção do *wafer* dividido pela quantidade de circuitos construídos nele. Desta forma, quanto maior a área ocupada por um *chip* no *wafer*, mais dispendioso ele será. Em consequência da otimização do leiaute, os circuitos se tornam menores e mais baratos, já que utilizam menor quantidade de matéria prima.

Para elaborar circuitos lógicos combinatórios é utilizado um dispositivo lógico programável chamado de Matriz de Portas. Tal dispositivo dispõe de uma dimensão de portas AND que podem ser ligadas a uma outra dimensão de portas OR, ambas programáveis. Sendo assim, ligações entre portas podem ser realizadas para alcançar resultados específicos. A Figura 1 apresenta o arranjo dos planos de uma Matriz de Portas, que permite que um grande número de funções lógicas seja produzido.

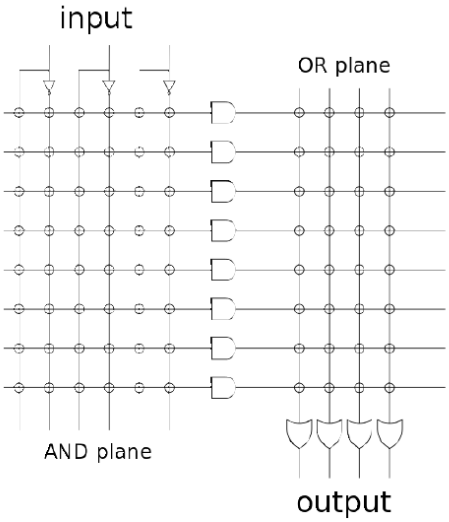


Figura 1. Exemplo de *Matriz Lógica Programável*, ou *Matriz de Portas*.

O problema de leiaute de matriz de portas (GMLP, do inglês *Gate Matrix Layout Problem*) é determinando como o problema de encontrar uma permutação de portas de tal forma que a quantidade necessária de trilhas seja minimizada, de modo que a área e o custo de fabricação do circuito equivalente sejam minimizados.

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.

O modelo de uma matriz de portas é demonstrado na Figura 2(a), em que as linhas verticais representam as portas e as horizontais as redes, que indicam como as portas serão conectadas umas com as outras. A rede B, por exemplo, conecta as portas 1, 4, 5, 6 e 7,

enquanto a rede D conecta as portas 1, 2 e 3, na Figura 2(b). Para ligar as portas e criar as devidas conexões são empregados fios condutores. Vale ressaltar que, dependendo do arranjo das portas, para liga as de uma mesma rede pode haver a necessidade de passar por outras portas não pertencentes aquela rede. Além disso, como é apresentado na Figura 2(c), fios sem ligação podem ser alocados em uma mesma trilha de conexão, como acontece com as redes D e E.

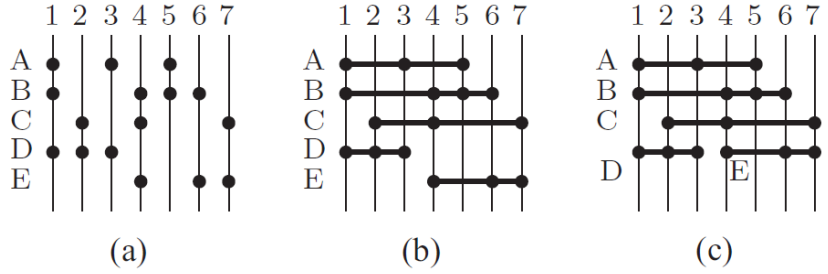


Figura 2. Exemplo de Matriz de Portas: (a) conexões exigidas, (b) redes já com fios e (c)t rilhas de conexão.

Uma análise sobre a eficácia do leiaute de um circuito pode ser feita através do custo das conexões, determinado pelo comprimento total dos fios utilizados, e da área total do circuito, indicada pela quantidade de trilhas.

O arranjo apresentado na Figura 2 tem um custo de 19 unidades de fios e 4 trilhas, que equivalem ao máximo de trilhas usadas. A Figura 3 apresenta um arranjo aperfeiçoado, em que a ordem das portas foi alterada passando de 1,2,3,4,5,6 e 7 para 1,3,5,2,4,6 e 7, que gasta apenas 15 unidades de fios em 3 trilhas.

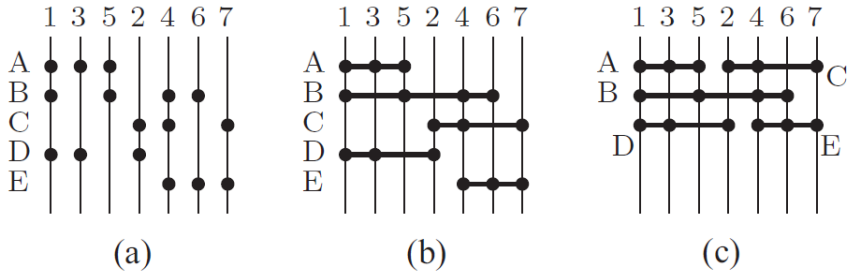


Figura 3. Leiaute otimizado da Matriz de Portas anterior.

Uma instância do GMLP pode ser descrita por uma matriz $M \in \{0, 1\}^{m \times n}$ e um parâmetro $\lambda \in Z_+$ representando o número de trilhas disponíveis e, portanto, um limitante superior para o número máximo de trilhas necessárias, indicando que quaisquer permutações com número de trilhas maior que λ são inviáveis. As linhas da matriz M são associadas às m redes e as colunas são associadas às n portas e $M(i, j)$=1 se e somente se a rede i incluir a porta j.

Uma solução para o GMLP em uma sequência ϕ: $[1, \cdots, n] \rightarrow [1, \cdots, n]$ 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 matriz M_ϕ é possuidora da propriedade dos 1s consecutivos, desta forma que M_ϕ (i, j)= 1 se e somente se, de acordo com ϕ, o fio da rede i incluir ou cruzar a porta j.

O GMLP foi provado ser NP-difícil por uma redução do problema de aumento grafos de intervalo (IGAP – *Interval Graph Augmentation Problem*) Kashiwabara, Fujisawa (1979) (*apud* Linhares e Horácio, 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*), 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 Separ*

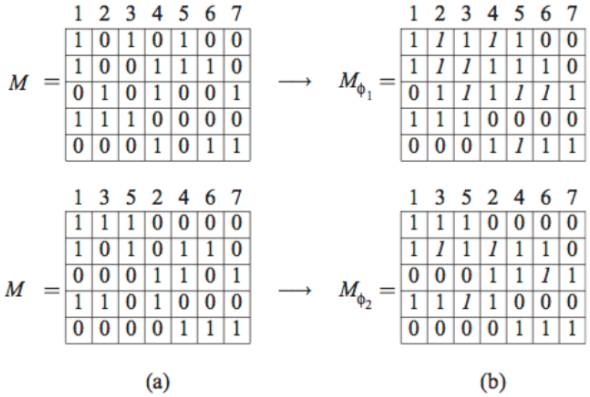


Figura 4. Matrizes *M* (a) e matrizes $M_{\phi 1}$ e $M_{\phi 2}$ das sequências [1, 2, 3, 4, 5, 6, 7] e [1, 3, 5, 2, 4, 6, 7], com os elementos trocados em itálico.

Objetivos

- Construir heurísticas consistentes e robustas para serem aplicadas na solução de problemas de leiaute de matrizes de portas, com o intuito de obter soluções próximas da ótima em pouco tempo.
- Analisar o comportamento heurístico de diferentes formulações para problemas de sequenciamento;
- Buscar métodos para melhorar as soluções obtidas pelas heurísticas usando programação linear
- Encontrar novas soluções ótimas para conjuntos de problemas já conhecido por meio de formulações já desenvolvidas.
- Aplicar os métodos desenvolvidos em problemas reais visando um avanço para indústrias do país.
- Produzir trabalhos para serem publicados em periódicos e eventos nacionais e internacionais, a fim de contribuir para a promoção dos centros de pesquisas nacionais e também da tecnologia.
- Apresentar um modelo de otimização que corresponda as características de diferentes aplicações do problema usando os modelos da literatura.
- Construir um software que através de ferramentas de Inteligência Computacional, como métodos heurísticos e metaheurísticos, consiga determinar o leiaute de matrizes de portas.
- Utilizar dados reais e instâncias disponíveis publicamente para avaliar o software implementado.
- Promover o Departamento de Computação e a Universidade Federal de Ouro Preto por meio da divulgação dos resultados obtidos neste projeto de pesquisa.

Revisão de literatura

Considerando os autores que aplicaram métodos heurísticos para a resolução do GMLP, cita-se o trabalho seminal de Wing, Huang e Wang (1985) que também trataram o *net assignment.* Neste trabalho o GMLP foi modelado usando grafos de intervalos, considerando portas em comum em diferentes trilhas como uma interseção. Estas trilhas com portas em comum podem ser representadas por uma única trilha, diminuindo a área necessária para implementação do circuito. Como experimento, obtiveram leiautes para circuitos CMOS de 180 e 306 transistores em 9 e 28 segundos, respectivamente, em um computador mainframe. No entanto, Hwang, Fuchs e Kang (1987) conseguiram resultados mais relevantes com o algoritmo *min-net-cut* modificado em conjunto com *dynamic-net-lists* e complexidade de tempo *O(n log n)*, em que *n* representa o número total de transistores e conexões de portas. Tais resultados apresentados foram a diminuição do número de trilhas no circuito em relação ao trabalho anterior, gerando um leiaute com 11 trilhas a menos na instância de 360 transistores e 71 portas. O algoritmo *min-net-cut* foi utilizado novamente por Chen e Hou (1988), que juntamente com uma representação dinâmica das ligações de um circuito criaram um algoritmo para o GMLP que analisa a dualidade do CMOS. O algoritmo, cuja complexidade de tempo é O(*n* log *n*), em que *n* é o número de portas, apresentou melhores resultados que os anteriores da literatura nos 5 circuitos testados. Já Singh e Chen (1990) modelaram o problema utilizando multigrafos e equações lógicas para determinar a sequência das portas. Os experimentos computacionais reportados consideraram 5 instâncias e incluíram o algoritmo de Hwang, Fuchs e Kang (1987), o qual foi superado em todas as instâncias. Chen e Hu (1990a, 1990b) apresentaram dois algoritmos, *GM_Plan* e *GM_Learn*, ambos baseados em paradigmas de inteligência artificial que alcançaram, ou determinaram, os melhores resultados da literatura até aquele momento. Estes mesmos trabalhos abordaram as mesmas instâncias utilizadas por Deo, Krishnamoorthy e Langston (1987), Huang e Wing (1989), Heinbuch (1988), Chang, Chang e Hsu (1987), Wing (1982), Wing e Huang (1985), Li (1983), Leong (1986), Nakatani et al. (1986), Wing (1983). No final da década de 80 e boa parte da década de 90 o GMLP foi abordado predominantemente por técnicas metaheurísticas, relacionadas na sequência. Após este período, Singh e Chen (1992) voltam a utilizar o algoritmo *mincut* na implementação de uma solução para o GMLP. Comparações quanto ao número de trilhas foram feitas com a implementação do algoritmo apresentado por Hwang, Fuchs e Kang (1987), que também é utiliza o *mincut*. Apesar de conseguirem melhores resultados em todas as instâncias, o algoritmo apresentado neste artigo demora mais tempo para alcançar a solução.

Shahookar et al. (1993) propôs uma nova implementação de algoritmo genético em conjunto com *beam search* a qual foi denominada *Genetic Beam Search*. O algoritmo genético do pacote Gênesis (Grefenstette e Schraudolph, 1987) foi utilizado e adaptado por funções de avaliação apropriadas para implementar o *beam search*. Considerando três instâncias que também foram utilizadas por Hong, Park e Kim (1989), foi possível diminuir o número de trilhas em um caso e obter menor comprimento de rede em todos.

Com a estratégia de busca predatória, a qual restringe a área de busca e a cada nova melhoria encontra uma solução, Linhares (1999) alcançou os resultados obtidos por Chen e Hu (1990a, 1990b). Além disso, das 25 instâncias, em 12 conseguiu igualar o número de trilhas ao limite inferior, obtido pela seleção da maior soma de coluna da matriz original, e em 4 conseguiu os melhores resultados da literatura, para os outros casos ele conseguiu alcançar os resultados obtidos pelo *GM_Plan* e o *GM_Learn*. Linhares, Yanasse e Torreão (1999) apresentaram a otimização microcanônica, um procedimento derivado da física estatística assim como o *simulated annealing,* no entanto, considera um sistema termicamente isolado. Dentre os 30 circuitos testados, o método foi capaz de igualar todos os resultados anteriores

de Hong, Park e Kim (1989) nos circuitos lógicos unidimensionais e de Chen e Hu (1990a, 1990b) nos bidimensionais, dos quais em 5 superou com grande vantagem os melhores leiautes.

O algoritmo genético foi utilizado novamente por Oliveira e Lorena (2002). Foi proposta uma abordagem construtiva que, considerando as onze instâncias introduzidas por Chen e Hu (1990a, 1990b) e Linhares, Yanasse e Torreão (1999), alcançou todos os melhores resultados conhecidos da literatura, obtidos anteriormente pela otimização microcanônica de Linhares, Yanasse e Torreão (1999). 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 et al. (2002), que utilizou um algoritmo memético como motor de busca. Este trabalho igualou os resultados da busca predatória de Linhares (1999). No entanto, o critério de parada do algoritmo foi o tempo de execução que variou de 30 segundos a 60 minutos. Já a abordagem de população múltipla de Mendes e Linhares (2004), agregou algoritmos genético e memético. Este trabalho apresentou resultados iguais aos da otimização microcanônica de Linhares, Yanasse e Torreão (1999), superando todas as abordagens metaheurísticas e heurísticas anteriores da literatura – incluindo *simulated annealing*, *microcanonical 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*. A única diferença de desempenho entre este algoritmo e a otimização microcanônica de Linhares, Yanasse e Torreão (1999) está na alta eficiência, em termos do número de avaliações, sendo que tal redução de esforço varia entre 80% e 90%, dependendo do tamanho da instância.

Por fim, Giovanni et al. (2013) considerou não só o GMLP, mas também o problema de minimização de custo de conexões do GMLP, e problemas correlatos, como o tempo e o número máximo de pilhas abertas. Dois algoritmos foram utilizados, sendo o primeiro uma heurística baseada em uma abordagem genética em conjunto com a definição composta e dinâmica da função de fitness (GAG), e o segundo, um *branch-and-cut*, que explora a flexibilidade de uma nova formulação de programação inteira com base na propriedade dos 1's consecutivos em matrizes. Foram utilizadas as 330 instâncias de Chen e Hu (1990a), SCOOP Team (2009), Faggioli e Bentivoglio (1998) e Smith e Gent (2005). Em comparação com Lorena (2002), nas 11 instâncias VLSI de Chen e Hu (1990a), o método proposto apresentou uma performance melhor. Especificamente nas instâncias do MOSP foram obtidos resultados ruins com *gap* alto ou sequer resolveram a relaxação linear em uma hora.

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.

Uma versão restrita do GMLP fixa previamente o número de trilhas a serem utilizadas na solução, caracterizando o *k*-GMLP, em que o parâmetro *k* indica tal número de trilhas. Esta variante do GMLP possui complexidade polinomial, dado que existem algoritmos com complexidade $O(n)$ e $O(n^2)$. Kinnersley e Kinnersley (1994) criaram um algoritmo de complexidade cúbica (O(n3), em que n se refere ao número de portas) baseado em teoria dos grafos para a versão de decisão do k-GMLP, com $\underline{k}=3$. Karonski, Scheinerman e Singer-Cohen (1999) introduziram o conceito de grafos de intervalo aleatórios com aplicação à caracterização do k-GMLP a partir de qualquer instância e para qualquer valor de *k*.

Para maiores detalhes sobre o *net assignment,* referimos o leitor aos trabalhos Shu, Wu e Kang (1988), Rim e Nakajima (1988), Xu, Kuh e Chan (1990),

PLA folding: Hu, Moerder e Morgenthaler (1992), Yamada e Nakayama (1992), Ferreira e Song (1992).

Material e Métodos

Para o desenvolvimento do projeto, foi aplicada uma metodologia com as seguintes etapas:

- Revisão da literatura levando em conta heurísticas para a resolução do GMLP e de outros problemas correspondentes, cobrindo uma vasta área com artigos de 1983 até 2013.
- Estudo dos principais métodos de resolução apresentados na literatura como de sucesso para a resolução do problema considerado e de problemas correlatos;
- Implementação e testes com técnicas de inteligência computacional promissoras, ainda não consideradas na literatura para o GMLP;
- Projeto e análise experimental de algoritmos construtivo e de busca local para o problema;
- Implementação das técnicas já consideradas promissoras pela literatura para a resolução do problema;
- Avaliação computacional das implementações utilizando a base de dados do SCOOP Team (2006), Smith e Gent (2005), Chen e Hu(1990).
- Escrita de um relatório técnico e de artigos científico;
- Documentação do software.

Resultados e Discussão

Além de promissores resultados computacionais com instâncias já utilizadas na literatura, a partir do presente projeto de pesquisa, foi possível submeter um artigo em uma conferência de classificação A1, o *International Joint Conferences on Artificial Intelligence* (IJCAI), e posteriormente será enviado para o Simpósio Brasileiro de Pesquisa Operacional (SBPO).

A seguir são apresentados experimentos computacionais abrangentes envolvendo 256 casos de quatro conjuntos diferentes, incluindo casos reais.

Instâncias reais VLSI

Este primeiro conjunto contém 25 casos VLSI reais introduzidos por Chen e Hu (1990). A Tabela 1 apresenta os resultados (expressos em número de faixas) e os tempos de utilização (expresso em milissegundos) da heurística proposta para estes casos. Os melhores resultados para este conjunto, obtido pelo *Predatory Search* de Linhares (1999) também são apresentados como valores de referência.

Tabela 1. Resultados das instâncias reais VLSI

	Predatory Search	Heurística Proposta	
Instância	Resultado	Resultado	Tempo (ms)
v4000	5	5	3,93
v4050	5	6	2,60
v4090	10	10	8,96
v4470	10	9	40,47
vc1	9	9	3,98
vl	3	3	0,24
vw1	4	4	0,20
vw2	5	5	0,58
w1	4	4	2,76
w2	14	14	50,61
w3	21	19	218,90
w4	32	28	774,80
wan	6	6	0,61
wli	4	4	1,00
wsn	8	8	3,73
x0	11	11	36,00
x1	5	5	0,53
x2	6	6	1,05
x3	7	7	2,53
x4	2	2	0,26
x5	2	2	0,79
x6	2	2	3,15
x7	4	4	0,87
x8	4	4	2,83
x9	4	4	11,04

Em uma quantidade muito pequena de tempo, a heurística proposta foi capaz de igualar todos os melhores resultados menos um, que foi determinado pela *Predatory Search* de Linhares (1999), e melhorar a três deles.

Instâncias pequenas MOSP reais

Este conjunto de instâncias, fornecido pelo SCOOP Team (2009), disponível em *http: //www.scoop-project*. net, contém 187 casos reais do MOSP. No entanto, a maioria destes casos são muito pequenas, por exemplo, duas linhas e colunas. Desse conjunto, 24 casos, com dimensões que variam de 10 linhas e colunas a 202 linhas e 141 colunas foram escolhidos para uso nos experimentos. A Tabela 2 apresenta os resultados utilizando o mesmo estilo da tabela anterior, exceto para as duas primeiras linhas, onde "Wood. A (4)" indica um grupo de quatro casos e "Wood. B (11)" indica um grupo de 11 casos, por consequência, os valores médios são apresentados nestas linhas. Os valores de referência são os obtidos por meio do algoritmo genético (AG) proposto por De. Giovanni et al. (2013).

Tabela 2. Resultados das instâncias pequenas MOSP reais

	GA	Heurística Proposta	
Instância	Resultado	Resultado	Tempo (ms)
Wood. A (4)	5,75	4,75	3,01
Wood. B(11)	5,73	5,81	4,16
A_FA_1	12	13	103,66
A_FA_11	11	11	57,32
A_FA_12	9	9	29,10
A_FA_13	18	18	197,19
A_FA_15	9	10	21,53
A_FA_2	11	11	31,00
A_FA_6	13	13	64,03
A_FA_8	12	12	65,82
B_REVAL_145	7	7	56,96

Quatro dos melhores resultados anteriores para as instâncias do grupo Wood. A foram melhorados pela heurística proposta. Os resultados mais conhecidos para 7 casos individuais foram pareados, e para o outro grupo de casos e por dois dos casos individuais, piores resultados foram gerados. Mais uma vez, os tempos de execução são baixos, sendo, no máximo, menos de meio segundo.

Instâncias pequenas MOSP artificiais

Os resultados para um subconjunto de 45 instâncias MOSP, propostas por Smith e Gent (2005) são apresentados na Tabela 3. Este subconjunto foi selecionado levando em conta as dimensões dos casos e a ausência de estruturas especiais que podem facilitar a solução, tal como analisado por Yanasse e Senne (2010). Novamente, as primeiras duas linhas indicam grupos, sendo os valores médios exibido e os valores de referência são os obtidos por meio do algoritmo genético (GA) proposto por De Giovanni et al. (2013), quando aplicável - os autores do algoritmo genético não informaram as soluções individuais para todas as instâncias.

Para o primeiro grupo de casos, todos os resultados foram igualados, enquanto que para o segundo os resultados diferem por uma fração. Considerando-se os casos individuais, três resultados foram igualados e outros cinco piores resultados foram obtidos. No entanto, fica claro que a heurística proposta enfrentou dificuldades em resolver este conjunto de casos, já que os tempos de execução cresceram para 2,55 minutos para alguns casos individuais. Apesar do aumento desses valores, eles ainda são considerados baixos e desejável.

Tabela 3. Resultados das instâncias pequenas MOSP artificiais

	GA	Heurística Proposta	
Instância	Resultado	Resultado	Tempo (ms)
Shaw (25)	13,68	13,68	30,70
Wilson (12)	19	19,5	2475,21
NWRS1	-	3	1,52
NWRS2	-	4	2,47
NWRS3	-	7	9,81
NWRS4	-	7	17,25
NWRS5	-	12	50,20
NWRS6	-	12	52,13
NWRS7	10	10	117,10
NWRS8	16	16	223,08
GP1	-	45	8431,32
GP2	-	41	8347,87
GP3	-	41	7371,09
GP4	-	33	5305,29
GP5	95	96	153276,28
GP6	75	75	127602,70
GP7	75	78	153019,75
GP8	60	67	81140,10
SP1	-	9	11,06
SP2	-	20	102,48
SP3	34	36	567,96
SP4	53	56	1532,99

Instâncias grandes MOSP artificiais

Um terceiro conjunto de 150 instâncias artificiais grandes do MOSP também foi considerado, disponível em http://www.decom.ufop.br/marco/pesquisa/problem-instances/. Estes casos foram gerados aleatoriamente, não têm nenhuma das estruturas específicas apontadas por Yanasse e Senne (2010) que facilitam a solução e as dimensões variam entre 150 linhas e colunas para 200 linhas e colunas. Na Tabela 4, "Random_r_c_d" identifica cada conjunto de instâncias 10, em que r significa o número de linhas, c representa o número de colunas e d é o número de redes por porta, isto é, o custo fixo para cada porta. Os valores médios de referência são as soluções ideais provadas ou o melhor limite inferior disponível (previsto por Chu e Stuckey (2009)), já que nenhum dos métodos anteriores foram testados usando essas instâncias.

Tabela 4. Resultados das instâncias grandes MOSP artificiais

		Heurística Proposta	
Instância	Solução ótima	Resultado	Tempo (ms)
Random_150_150_2	25,9	30	789,06
Random_150_150_4	61,6	65,7	2655,74
Random_150_150_6	93,2	97,3	6664,61
Random_150_150_8	111,7	115,4	12610,57
Random_150_150_10	123,9	127,2	20158,1
Random_175_175_2	30,3	35,11	1232,49
Random-175_175_4	73,7	78,9	4106,86
Random_175_175_6	107,7	112,8	10225,45
Random_175_175_8	128,3	132,3	18666,04
Random_175_175_10	143,5	147,1	31320,01
Random_200_200_2	36	41,2	1774,56
Random_200_200_4	84,3	88,9	5430,38
Random_200_200_6	121,5	126	13430,33
Random_200_200_8	147,1	152,8	25742,51
Random_200_200_10	162,8	167,8	42548,39

A diferença percentual geral (calculado como 100 (valor de referência de valor da solução heurística) / valor de referência) é de 4,62%. Considerando que estas são instâncias maiores sem estruturas especiais, esta é uma marca considerável. Surpreendentemente, os tempos de operação são menores que os obtidos para alguns casos menores, como no conjunto anterior: o tempo médio de execução permaneceu sob 43 segundo, mesmo para os maiores casos.

Conclusões

O cronograma inicial proposto para este projeto de iniciação científica foi cumprido rigorosamente, e os objetivos almejados foram alcançados.

Especificamente, a heurística desenvolvida foi capaz de igualar uma boa parte dos melhores resultados disponíveis na literatura, além melhorar alguns deles e obter uma diferença de 4,62% quando comparado com limites inferiores e soluções ótimas, quando disponíveis.

Um artigo científico foi submetido ao *International Joint Conferences on Artificial Intelligence* (IJCAI) 2015, ainda em fase de revisão, e um novo artigo será submetido ao Simpósio Brasileiro de Pesquisa Operacional em abril deste ano.

Referências bibliográficas

Chang, Y. C., Chang, S. C. and Hsu, L. H., "Automated Layout Generation Using Gate Matrix Approach," Proc. 24th DA Conf., pp 552-558, 1987.

Chen, C.Y.R., Hou, C.Y., 1988 New Algorithm for CMOS Gate Matrix Layout. IEEE Computer-Aided Design, ICCAD-88, 138 – 141.

Chen, S. J. and Hu, Y. H., 1990, "GM_Learn: An iterative learning algorithm for CMOS gate matrix layout," Proc. Inst. Elect. Eng., vol. 137, pt. E, pp. 301–109, 1990.

Chen, S. J. and Hu, Y. H., 1990, GM_Plan: A gate matrix layout algorithm based on artificial intelligence planning techniques. IEEE Transactions on Computer-Aided Design, 9, 836–845.

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

D. K. Hwang, W. K. Fuchs, and S. M. Kang, *An eficient approach to gate matrix layout*, IEEE Trans. Computer Aided Design, CAD-6 (1987), pp. 802-809.

Deo, N., Krishnamoorthy, M. S., and Langston, M. A., "Exact and Approximate Solutions for the Gate Matrix Layout Problem." IEEE Tr. CAD, vol. CAD-6, no. 1, pp. 79-84, 1987.

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

De. Giovanni, L., Massi, G., Pezzella, F., Pfetsch, M. E., Rinaldi, G. e Ventura, P., 2013, A heuristic and an exact method for gate matrix connection cost minimization problem. International Transactions in Operational Research, v20, pp 627-643.

Grefenstette, J. J., Schraudolph, N. N., "A User's Guide to GENESIS 1.2ucsd" CSE Dept., University of California, San Diego, 1987.

Heinbuch, D. V., CMOS3 Cell Librury, Addison-Wesley , Massachusetts, 1988.

Horacio Hideki Yanasse e Edson Luiz Frana Senne. 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, Jun. 2010.

Huang, S., and Wing, O., "Improved Gate Matrix Layout", IEEE Trans. on CAD, vol. 8, no. 8, August 1989, pp 875-889.

Leong, H. W., "A New Algorithm for Gate Matrix Layout," Proc. ICCAD, pp. 316-319, 1986.

Li, J. T., "Algorithms for Gate Matrix Layout" Proc. ISCAS, pp. 1013-1016, 1983.

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

Linhares, A., Yanasse, H., and Torreão, J., 1999, Linear Gate Assignment: a fast statistical mechanics approach. IEEE Transactions on Computer-Aided Design on Integrated Circuits and Systems, 18, 1750–1758.

Linhares, A., Yanasse, H.H., 2002. Connections between cutting-pattern sequencing, VLSI design, and flexible machines. Computers & Operations Research 29, 1759–1772.

Mendes, A. e Linhares, A., 2004, A multiple-population evolutionary approach to gate matrix layout. International Journal of Systems Scienc, v 35, pp 13-23.

Mendes, A., França, P., Moscato, P. e Garcia, V., 2002, Population Studies for the Gate Matrix Layout Problem. Springer Berlin Heidelberg Advances in Artificial Intelligence, pp 319-328.

Möhring, R., 1990. Graph problems related to gate matrix layout and PLA folding. Computing 7, 17–51.

Nakatani, K., Fujii, T., Kikuno, T. and Yoshida, N., "A Heuristic Algorithm for Gate Maaix Layout," Proc. ICCAD, pp. 324-327, 1986.

Oliveira, A.C.M., 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 21, 8, 969–974.

SCOOP Team, 2009. Sheet cutting and process optimization for furniture industry – SCOOP Project. Avaliable at http://www.scoop-project.net (accessed February 3, 2015).

Shahookar, K., Khamisani, W., Mazumder, P. e Reddy, S.M., 1993, *Genetic beam search for Gate Matrix Layout.* IEEE, Computers and Digital Techniques, 23 – 128.

Singh, U., Chen, C.Y.R., *A Transistor Reordering Technique for Gate Matrix Layout.* IEE Design Automation Conferenc, 462 – 467, 1990.

Singh, U., Chen, C.Y.R., *From Logic to Symbolic Layout for Gate Matrix,* 1992. IEEE, Computer-Aided Design of Integrated Circuits and Systems, 216 – 227.

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

Wing, O., "Automated Gate Matrix Layout," Proc. ISCAS, pp. 681-685, 1982.

Wing, O., "Interval-Graph-Based Circuit Layout," Proc. ICCAD, pp. 84-85, 1983.

Wing, O., Huang, S. and Wang, R., "Gate Matrix Layout," IEEE Tr. CAD, vol. CAD-4, no. 3, pp. 220-231, 1985.

WING, O., HUANG, S., and WANG, R., 1985, Gate matrix layout. IEEE Transactions on Computer-Aided Design, 4, 220–231.

Y. Hong, K. Park, M. Kim "A Heuristic Algorithm for Ordering the Columns in One-Dimensional Logic Arrays" IEEE Trans. on Computer-Aided Design, Vol. 8, No. 5, pp. 547- 562, 1989

Anexos

Artigo *A Two Phase Heuristic for the Gate Matrix Layout Problem* submetido para a *International Joint Conferences on Artificial Intelligence* (IJCAI).

A Two Phase Heuristic for the Gate Matrix Layout Problem

João Vitor Mascarenhas dos Santos and Marco Antonio Moreira de Carvalho
Computer Science Department, Universidade Federal de Ouro Preto
Ouro Preto, Minas Gerais, Brazil
joaovitormascarenhas@yahoo.com.br, mamc@iceb.ufop.br

Abstract

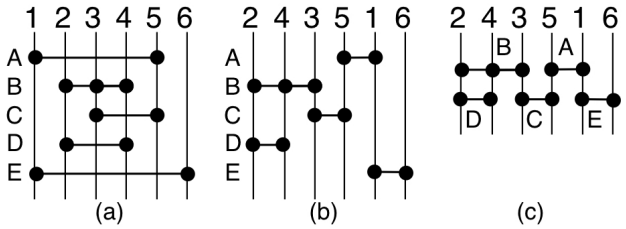
This paper introduces a two-phase heuristic method for the Gate Matrix Layout Problem (GMLP), a practical application problem. In the context of Very Large Scale Integration (VLSI) design, this problem asks for a permutation of columns of a gate matrix such that the number of required tracks necessary to implement the corresponding integrated circuit, and thus the area, are minimized. The proposed heuristic is based on graph search 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.

1 Introduction

In the context of Very Large Scale Integration (VLSI) design, a *Programmable Gate Matrix*, or just *Gate Matrix* is a programmable logical device used to implement combinatorial logic circuits. A gate matrix has a programmable AND gates plane that may be connected to a plane of OR gates equally programmable.

In electronic circuits implemented by a gate matrix, the *gates* correspond to *circuit nodes*, and different connections among them are necessary to produce specific logical functions. Each connection uses a *wire* and a subset of nodes, called *net*.

Figure 1(a) shows a sample gate matrix with six gates (numbered from 1 to 6) represented vertically and five nets (named from A to E) represented horizontally. Net A connects gates 1 and 5, net B connects gates 2, 3 and 4 and so on. Depending on the *layout* of the matrix gates (that is, the order of the gates), in order to create a net, it may be necessary the wire to cross some gates that do not belong to that net, as the net A from figure 1(a) that crosses gates 2, 3 and 4.

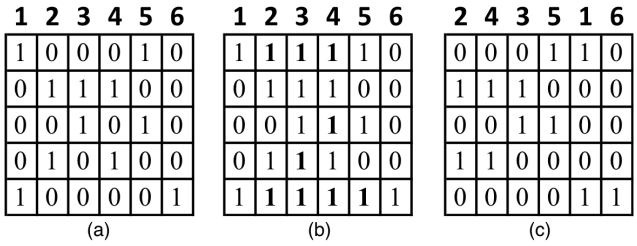


Original gate matrix (a), with gates permuted (b) and compacted (c).

The underlying logic implemented by the circuit is not changed if the gates layout is changed, as in figure 1(b), and clearly the overall circuit area is a function of the number of *tracks*, (i. e. physical rows) needed to implement the nets of the circuit. The *Gate Matrix Layout Problem* (GMLP) asks for a layout of gates such that the number of necessary tracks, and therefore the circuit area, are minimized. According to Ferreira and Song [**?**], the problem is divided into two steps: the *permutation step* (figure 1(b)) and the *compaction step* (figure 1(c)). In the first step, a permutation of gates is determined, while in the second step, different nets are grouped in single tracks when possible – what can be done in polynomial time.

Formally, a GMLP instance can be described as a binary matrix M such that entry $m_{ij} = 1$ if net i includes gate j and $m_{ij} = 0$ otherwise. This matrix holds the consecutive ones property due to the nets implementation: the wire connecting two gates needs to cross every intermediary gate, thus, in each row every zero between two ones will turn into a fill-in, that is, will assume value one. The GMLP asks for a permutation of columns such that the maximum column sum (the number of necessary tracks), including the fill-ins is minimized.

Figure 2 shows (a) the matrix that corresponds to figure 1(a), (b) the consecutive ones property and (c) the solution with the gates permutation π=[2, 4, 3, 5, 1, 6]. The maximum column sum is 2, thus, the associated circuit can be implemented using 2 tracks, as shown in figure 1(c).



Original matrix (a), consecutive ones highlighted (b) and permuted columns (c).

This NP-Hard problem [**?**] has great practical importance in the microelectronics industry, being relevant in the areas of Information Technology, Telecommunications, Industrial Process Control, Automation & Control and also in the manufacturing of consumer goods, with direct practical applications in engineering and industry. The problem in question is quite general and models successfully a large set of graph theory problems [**?**]: Modified Cutwidth, Interval Thickness, Node Search Game, Edge Search Game, Narrowness, Split Bandwidth, Graph Pathwidth, Edge Separation and Vertex Separation – and scheduling problems [**?**]: Minimization of Open Stacks Problem (MOSP) and Programmable Logic Array Folding.

In this work we propose a new heuristic for the Gate Matrix Layout Problem and report extensive computational experiment using instances and results available in the literature.

1.1 Brief Literature Review

Heuristics have been designed for the GMLP since the late eighties. Wing *et al.* [**?**] modeled two real circuits using interval graphs, whose results were later improved by Hwang *et al.* [**?**] using the linear-logarithmic modified *min-net-cut* algorithm. The same algorithm was used by Chen and Hou [**?**] to improve again the previous results. Two algorithms, GM-Plan [**?**] and GM-Learn [**?**], both based on artificial intelligence techniques, matched or determined the best solution values for all instances available until then.

In the next decade, the GMLP was predominantly addressed with metaheuristics and hybrid methods. Shahookar *et al.* [**?**] proposed an implementation of Genetic Algorithm and Beam Search, named *Genetic Beam Search*. Linhares [**?**] introduced a *Predatory Search* method, which matched some of the best results in the literature obtained by GM-Plan and GM-Learn, twelve of them proved optimal, and also improved the solution values for other four instances. The *Microcanonical Optimization*, a procedure derived from statistical mechanics, was introduced by Linhares *et al.* [**?**]. This method was able to match or improve the best results previously obtained by Chen and Hu [**?**] and Hu and Chen [**?**], but not the results obtained by the *Predatory Search* method. Additionally, the *Microcanonical Optimization* improved other results for one dimensional circuits (logical arrays). *Constructive Genetic Algorithms* were employed by Oliveira and Lorena [**?**] to match all the best results obtained by the *Microcanonical Optimization* – according to the authors, the proposed genetic algorithm is more robust. A multi-population evolutionary approach, with a *Memetic Algorithm* and a *Genetic Algorithm* as search engines, was proposed by Mendes and Linhares [**?**]. This work introduced the concept of "critical columns", allowing a substantial reduction of the search space. The method matched the results obtained by the *Microcanonical Optimization*, and outperformed GM-Plan, GM-Learn and implementations of *Simulated Annealing*, *Microcanonical Annealing* and a variety of constructive heuristics.

Exact methods for the GMLP are found in the literature much less often. A dynamic programming formulation is provided in [**?**], however, experiments were not conducted. Recently, De Giovanni *et al.* [**?**] consider the minimization of tracks number rather as a constraint with the objective of minimizing the cost of the connections in a circuit, expressed by the total wire length used. A genetic algorithm and a branch-and-cut procedure are presented and tested over three hundred instances from the literature. Specifically, the genetic algorithm was used to determine a threshold on the number of required tracks, and the branch-and-cut procedure was used to minimize the wire length.

2 A Two-Phase Heuristic

The proposed heuristic has two phases: a constructive phase, when an instance is represented as a graph and an initial solution is constructed by means of a search, and an improvement phase, when a local search is applied to the initial solution.

The heuristic also tries to reduce the problem size using a preprocessing technique that removes "dominated" gates [**?**]: A dominated gate g_1 is one whose set of nets is a subset of another gate g_2 set of nets, e.g., a gate g_2 contained in nets 1, 4, 8, 9 dominates a gate g_1 contained in nets 1 and 9 only. If g_1 is sequenced in the solution right after g_2, there will be no increase in the number of necessary tracks, as g_1 is not included in any net different of those that include g_1. To identify the dominated gates, it is necessary to compare the set of nets that includes all pairs of gates, which requires $O(n^2 g)$ complexity, where n represents the number of nets and g represents the number of gates.

2.1 Constructive Phase

The first phase models the problem as a search in a graph that represent the nets. Based on a Breadth-First Search (BFS) the gates sequencing is determined.

Graph Representation and Search

Yanasse [**?**] introduced the representation of graphs for a problem of equivalent formulation that can be applied to the GMLP: a node corresponds to a net and an edge connects two nets that share gates in common. Therefore, each net induces a clique in the graph.

The BFS [**?**] has been previously applied successfully to problems related to the GMLP, generating very good suboptimal solutions for the MOSP [**?**]. Other benefit from the BFS is that it naturally detects graph components. Under this circumstance, the problem can also be decomposed into separate and distinct subproblems.

This particular implementation of the BFS takes as input a graph G and a node v with the minimum degree (ties are

broken in lexicon-decreasing order of the node label) and explores the neighborhood of each node in ascending degree order, returning at the end a list that contains the nodes sequence of exploration. If the graph has more than one component, then it has distinct subproblems. Therefore, the given procedure is applied separately to each one of them.

The choice of exploring minimum degree connected nodes is a greedy criterion. Its rationale stems from the expectation that nodes in a clique and denser graph regions are sequenced next to each other, that is, nets that share similar gates may be explored in succession.

Gates Sequencing

The BFS only returns a list of nodes from the graph without giving the permutation of gates that is necessary to solve the problem. This permutation of gates is obtained from the result of the BFS by traversing the nodes from the last to the first node, including in the solution the gates that are contained in the node (net) being considered. This procedure has complexity bounded by $O(ng)$. where n denotes the number of nets and g denotes the number of gates.

Becceneri *et al.* [**?**] presents another way to obtain the permutation of gates: the list generated by the BFS is traversed, indicating the current *active* nets (that is, at least one gate contained in the net is already inserted in the solution), and when all nets that contain a specific gate are active, then the gate is inserted in the solution. Similarly to the previous method, the complexity is bounded by $O(ng)$.

As both methods have an acceptable complexity, both are used to determine the gates permutation from the BFS result.

2.2 Improvement Phase

After an initial solution is generated, as described above, two local search procedures are applied ir order to improve its quality. The idea is to change the relative position of gates in the obtained permutation such that nets are compacted regarding the wire length used. Note that minimizing the number of tracks is not equivalent to minimizing the nets length, as pointed by Linhares and Yanasse [**?**].

Both procedures try to compact a net by eliminating the fill-ins from the matrix. To do so, each line of the matrix is individually analyzed and the gates are shifted to the left (first procedure) or to the right (second procedure) if there is no increase in the number of required tracks. As the BFS has a low running time, it is possible to apply systematically the local search. The proposed local search is adaptive and can analyze up to 70% of the gates for each net. This limit was defined empirically, in a search for balance between running time and solution quality. However, they may be extrapolated for small instances, e.g., 10 nets and gates.

The local search requires an examination of the gates within every net, thus, considering the matrix representation their complexity is bounded by $O(n^2g^2)$, where n denotes the number of nets and g denotes the number of gates. The proposed heuristic has, therefore, the same complexity bound. Note that in this calculation the maximum amount of required wire is considered to be g, although this is not the case of any instance considered in the experiments and thus, the worst case scenario is not often, as described in the next section.

3 Computational Experiments

The computational experiments were carried out on an Intel i5 Quad Core 3.2 GHz processor with 16 GB RAM using Ubuntu 12.4.1. The proposed heuristic code was written in C++, compiled with g++ 4.4.1 and the -O3 optimization option. Four data sets were solved in the computational experiments, and two different methods from the literature are used in the comparisons, where applicable. The running times are not compared in these experiments, as the methods were run on different architectures and no fair basis is available.

3.1 Real World VLSI Instances

The first set contains 25 real world VLSI instances introduced by [**?**]. Table 1 presents the results (expressed in number of tracks) and running times (expressed in milliseconds) of the proposed heuristic for these instances. The best results for this set, obtained by the *Predatory Search* [**?**] are also presented as reference values.

Table 1. Results for real world VLSI instances.

	Predatory Search	Proposed Heuristic	
Instance	Result	Result	Time (ms)
v4000	5	5	3.93
v4050	5	6	2.60
v4090	10	10	8.96
v4470	10	9	40.47
vc1	9	9	3.98
vl	3	3	0.24
vw1	4	4	0.20
vw2	5	5	0.58
w1	4	4	2.76
w2	14	14	50.61
w3	21	19	218.90
w4	32	28	774.80
wan	6	6	0.61
wli	4	4	1.00
wsn	8	8	3.73
x0	11	11	36.00
x1	5	5	0.53
x2	6	6	1.05
x3	7	7	2.53
x4	2	2	0.26
x5	2	2	0.79
x6	2	2	3.15
x7	4	4	0.87
x8	4	4	2.83
x9	4	4	11.04

In a very small amount of time, the proposed heuristic was able to match all but one the best results determined by the *Predatory Search* [**?**], and improve three of them. However, the best solutions for instances *v4470* (9), *w3* (18) and *w4* (27) were previously determined by the *Constructive Genetic Algorithm* reported by Oliveira and Lorena [**?**], and the proposed heuristic found worse results for the latter two, differing by one additional track.

3.2 Small Real World MOSP Instances

This set of instances, provided by the Sheet Cutting and Process Optimization for Furniture Industry (SCOOP) consortium and available at `http://www.scoop-project.net`, contains 187 real world MOSP instances from two enterprises. However, most of these instances are too small (e.g., 2 rows and columns). From this set, 24 instances with dimensions ranging from 10 rows and columns to 202 rows and 141 columns were select for use in the experiments. Table 2 presents the results using the same convention as the previous table, except for the two first lines, where "Wood. A (4)" indicates a group of 4 instances and "Wood. B (11)" indicates a group of 11 instances, therefore, the mean values are displayed for these lines. The reference values are those obtained by the Genetic Algorithm (GA) proposed by De Giovanni *et al.* [**?**].

Table 2. Results for small real world MOSP instances.

	GA	Proposed Heuristic	
Instance	Result	Result	Time (ms)
Wood. A (4)	5.75	4.75	3.00
Wood. B (11)	5.73	5.81	4.16
A_FA_1	12	13	103.66
A_FA_11	11	11	57.32
A_FA_12	9	9	29.10
A_FA_13	18	18	197.19
A_FA_15	9	10	21.53
A_FA_2	11	11	31.00
A_FA_6	13	13	64.03
A_FA_8	12	12	65.82
B_REVAL_145	7	7	56.96

Four of the best previous results for the instances of group *Wood. A* were improved by the proposed heuristic. The best known results for 7 individual instances were matched, and for the other group of instances and for two of the individual instances, worse results were generated. Again, the execution times are low, being the maximum less than a half second.

3.3 Small Artificial MOSP Instances

The results for a subset of 45 MOSP instances, proposed for the 2005 Constraint Modeling Challenge [**?**] is presented in Table 3. This subset was selected considering the dimensions of the instances and the absence of special structures that may facilitate the solution, as analyzed by Yanasse and Senne [**?**].

Again, the two first lines indicate grouped instances being the mean values displayed and the reference values are those obtained by the Genetic Algorithm (GA) proposed by De Giovanni *et al.* [**?**], where applicable – the authors of the Genetic Algorithm did not report the individual solutions for all instances.

For the first group of instances all results were matched, while for the second the results differed by a fraction. Considering the individual instances, three results were matched and other five worse results were obtained. Nonetheless, it is clear that the proposed heuristic faced difficulties in solving this set of instances, as the execution times grew up to 2.55 minutes for some individual instances. Despite the increase in these values, they still are considered low and desirable.

Table 3. Results for small artificial MOSP instances.

	GA	Proposed Heuristic	
Instance	Result	Result	Time (ms)
Shaw (25)	13.68	13.68	30.70
Wilson (12)	19	19.50	2475.21
NWRS1	–[1]	3	50.20
NWRS2	–	4	52.13
NWRS3	–	7	1.52
NWRS4	–	7	2.47
NWRS5	–	12	9.81
NWRS6	–	12	17.25
NWRS7	10	10	117.10
NWRS8	16	16	223.08
GP1	–	45	8431.32
GP2	–	41	8347.87
GP3	–	41	7371.09
GP4	–	33	5305.29
GP5	95	96	153276.28
GP6	75	75	127602.70
GP7	75	78	153019.75
GP8	60	67	81140.10
SP1	–	9	11.06
SP2	–	20	102.48
SP3	34	36	567.96
SP4	53	56	1532.99

[1] Individual results not available.

3.4 Large Artificial MOSP Instances

A third set of 150 large artificial MOSP instances is also considered, available at `http://www.decom.ufop.br/marco/pesquisa/problem-instances/`. These instances were randomly generated, have none of the special structures pointed by [**?**] that would facilitate the solution and the dimensions vary between 150 rows and columns to 200 rows and columns. In Table 4, "Random-r-c-d" identifies each collection of 10 instances, where r means the number of rows, c stands for the number of columns and d is the number of nets per gate, that is, the fixed demand for each gate. The reference mean values are the proved optimal solutions or the best lower bound available (provided in [**?**]), as none of the previous methods have been tested using these instances.

The overall percentage gap (computed as *100(heuristic solution value reference value)/reference value*) is 4.62%. Considering these are larger instances with no special structures, this is a considerable mark. Surprisingly, the running times were lesser than those obtained for some smaller instances, as in the previous set: the average running time remained under 43 seconds even for the larger instances.

4 Conclusions and Future Work

The Gate Matrix Layout Problem (GMLP) is an industrial NP-hard problem with application to microelectronics design that also models successfully a variety of problems from

Table 4. Results for large artificial MOSP instances.

	Optimal	Proposed Heuristic	
Instance	Solution	Result	Time (ms)
Random_150_150_2	25.90*	30.00	789.06
Random_150_150_4	61.60	65.70	2655.74
Random_150_150_6	93.20*	97.30	6664.61
Random_150_150_8	111.70*	115.40	12610.57
Random_150_150_10	123.90*	127.20	20158.10
Random_175_175_2	30.30	35.11	1232.49
Random_175_175_4	73.70	78.90	4106.86
Random_175_175_6	107.70	112.80	10225.45
Random_175_175_8	128.30*	132.30	18666.04
Random_175_175_10	143.50*	147.10	31320.01
Random_200_200_2	36.00	41.20	1774.56
Random_200_200_4	84.30	88.90	5430.38
Random_200_200_6	121.50	126.00	13430.33
Random_200_200_8	147.10	152.80	25742.51
Random_200_200_10	162.80*	167.80	42548.39

*Proved optimal.

graph theory and scheduling problems. The objective of the problem is to find a layout of components in such a way that the area required to implement the circuit is minimized. Thus, these circuits can be produced cheaper and faster.

This work proposed a two phase heuristic. The first phase models the problem as a graph with specific structure and uses the well known Breadth-First Search to get an initial solution. In order to improve the generated solution, two procedures of local search are applied to compact each set of components of the circuit.

Comprehensive computational experiments involved 256 instances from four different sets, including real world instances. The proposed heuristic was able to match a good part of the best results available, to improve some of them and to obtain a gap of 4.62% when compared to lower bounds and optimal solutions when available.

Future work will be concentrated in fine tuning the used local search, experimenting new local search techniques and also developing an exact method component to help guide the heuristic search, turning the method into a *matheuristic*.

The suggested heuristic can be incorporated into exact methods because it gives a reasonable upper bound to the problem, or it can be used directly as a feasible option to quickly obtain good solutions to the GMLP.

5 Acknowledgments

This research was supported by CNPq (grant 441565/2014-0) and CAPES.

References

[Becceneri *et al.*, 2004] José Carlos Becceneri, Horacio Hideki Yanasse, and Nei Yoshihiro Soma. 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, 2004.

[Carvalho and Soma, 2014] Marco Antonio Moreira de Carvalho and Nei Yoshihiro Soma. A breadth-first search applied to the minimization of open stacks. *Journal of the Operational Research Society*, 2014. (to appear).

[Chen and Hou, 1988] C.Y.R. Chen and C.Y. Hou. 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, Nov 1988.

[Chen and Hu, 1990] Sao-Jie Chen and Yu Hen Hu. Gmlearn: an iterative learning algorithm for cmos gate matrix layout. *Computers and Digital Techniques, IEE Proceedings E*, 137(4):301–309, Jul 1990.

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

[Cormen *et al.*, 2001] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. MIT Press, Cambridge, MA, 2nd edition, 2001.

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

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

[Deo *et al.*, 1987] N. Deo, M.S. Krishnamoorthy, and M.A. Langston. 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, January 1987.

[Ferreira and Song, 1992] A. Ferreira and S.W. Song. Achieving optimality for gate matrix layout and PLA folding : a graph theoretic approach. *Integration: the VLSI journal*, 14:173–195, 1992.

[Hu and Chen, 1990] Yu Hen Hu and Sao-Jie Chen. 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, Aug 1990.

[Hwang *et al.*, 1987] D.K. Hwang, W.K. Fuchs, and Sung Mo Kang. An efficient approach to gate matrix layout. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 6(5):802–809, September 1987.

[Linhares and Yanasse, 2002] Alexandre Linhares and Horacio Hideki Yanasse. Connections between cutting-pattern sequencing, VLSI design, and flexible machines. *Comput. Oper. Res.*, 29(12):1759–1772, 2002.

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

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

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

[Möhring, 1990] R.H. Möhring. *Graph problems related to gate matrix layout and PLA folding*, volume 7 of *Computing supplementum*, pages 17–51. Springer-Verlag, Berlin, 1990.

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

[Smith and Gent, 2005] Barbara Smith and Ian Gent. 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 *et al.*, 1985] O. Wing, Shuo Huang, and Rui Wang. Gate matrix layout. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 4(3):220–231, July 1985.

[Yanasse and Senne, 2010] Horacio Hideki Yanasse and Edson Luiz Frana Senne. 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, Jun. 2010.

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